UNIVERSIDADE FEDERAL DO PAMPA

Jonnathan Riquelmo Lopes Frescura

Uma Linguagem Específica de Domínio para a Representação de Modelos Conceituais de Bancos de Dados Relacionais

Alegrete
2019

Jonnathan Riquelmo Lopes Frescura

Uma Linguagem Específica de Domínio para a Representação de Modelos Conceituais de Bancos de Dados Relacionais

Projeto de Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Dr. Maicon Bernardino da Silveira

Coorientador: Prof. Dr. Fábio Paulo Basso

Alegrete
2019

Jonnathan Riquelmo Lopes Frescura

Uma Linguagem Específica de Domínio para a Representação de Modelos Conceituais de Bancos de Dados Relacionais

Projeto de Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Projeto de Trabalho de Conclusão de Curso defendido e aprovado em ..... de ..... de .....
Banca examinadora:

Prof. Dr. Maicon Bernardino da Silveira
Orientador
UNIPAMPA

Prof. Dr. Fábio Paulo Basso
Coorientador
UNIPAMPA

Prof. Dr. Sérgio Luis Sardi Mergen
UFSM

Prof. Dr. Elder de Macedo Rodrigues
UNIPAMPA

RESUMO

Com o avanço da tecnologia os bancos de dados passaram a ser elementos vitais na sociedade contemporânea. Os bancos de dados são conjuntos de dados armazenados para retratar algum sentido sobre um domínio específico. As informações armazenadas são consideradas bens de grande relevância nas organizações modernas. Dessa forma o uso eficaz de bancos de dados é de suma importância para a manutenção e o prosseguimento correto das suas atividades. Posto isto, a capacitação nessa área para profissionais oriundos da academia deve ser constante, sendo esse um ponto fundamental com o qual as instituições de ensino superior devem ter especial atenção. Contudo, a variedade de tecnologias de sistemas de banco de dados que se tornaram disponíveis nos últimos anos, sendo a grande maioria focada em abordagens gráficas, dificulta a escolha de ferramentas para modelagem de entidade-relacionamento (ER) na indústria e, consequentemente, no meio acadêmico. Objetivando contribuir com uma alternativa *open source* relevante, este Projeto de Trabalho de Conclusão de Curso propõe uma Linguagem Específica de Domínio (*Domain Specific Language* - DSL) textual para apoiar o processo de ensino-aprendizagem da modelagem conceitual de banco de dados. O uso de DSLs fornece meios de especificar e modelar domínios de forma mais rápida e produtiva, pois são linguagens com expressividade limitada a domínios particulares, diferenciando-se assim das linguagens de propósito geral. Nesse sentido, foi executado uma investigação do estado da arte e da prática em projeto e modelagem de banco de dados utilizando DSLs. Um levantamento de inovações recentes foi realizado por meio de um mapeamento sistemático complementado por uma pesquisa na literatura cinza. Esse trabalho abrange um conjunto final de 10 estudos primários focados em DSLs e identifica 55 ferramentas já aplicadas na indústria e academia para modelagem ER em nível conceitual, lógico e físico. Em seguida, houve a seleção do *framework* Xtext para apoiar o desenvolvimento inicial da linguagem de modelagem. Por meio disso foi possível inferir requisitos necessários, decisões de projeto e então realizar a definição preliminar de uma gramática. Posteriormente, ocorreu a implementação de um protótipo funcional e a integração da DSL em um RCP (*Rich Client Platform*) Eclipse. Dessa forma, houve o teste prévio do projeto da proposta. Nele, o processo de modelagem com a nova linguagem criada ganhou recursos nativos como formatação, validação com base nas restrições descritas na gramática e *syntax highlighting*. O *plugin* pôde ser testado devido a integração nativa fornecida pelo Xtext com o EMF (*Eclipse Modeling Framework*), um conjunto de recursos do Eclipse para representar modelos e gerar código equivalente.

Palavras-chave: Banco de Dados. Projeto e Modelagem de Banco de Dados. Modelagem Conceitual. Linguagem Específica de Domínio.

ABSTRACT

With the advance of technology, databases have become vital elements in contemporary society. Databases are stored data sets to describe some meaning about a specific domain. The information stored is considered to be of great relevance in modern organizations. In this way, the effective use of databases have great importance for the maintenance and correct progress of their activities. That said, the formation in this area for professionals coming from academy must be constant, which is a fundamental point with which higher education institutions should pay special attention. However, the variety of database systems technologies that have become available in recent years, most of which are focused on graphical approaches, make it difficult to choose entity-relationship (ER) modeling tools in industry and, consequently, in the academy. In order to contribute with a relevant open source alternative, this Course Conclusion Work Project proposes a Textual Domain Specific Language (DSL) to support the teaching-learning process of conceptual database modeling. The use of DSLs provides means to specify and model domains more quickly and productively, since they are expressive languages limited to particular domains, thus differentiating themselves from general-purpose languages. In this sense, an investigation of the state of the art and the practice in database design and modeling using DSLs was performed. A survey of recent innovations was carried out through a systematic mapping complemented by a survey in the gray literature. This work covers a final set of 10 primary studies focused on DSLs and identifies 55 tools already applied in industry and academy for ER modeling at conceptual, logical and physical level. Then, there was the selection of the Xtext framework to support the initial development of the modeling language. Through this, it was possible to infer necessary requirements, design decisions and then make out the preliminary definition of a grammar. Later, an implementation of a functional prototype and a DSL integration in an Eclipse Rich Client Platform (RCP) occurred. In this way, there was the preliminary test of the project for this proposal. In it, the modeling process with the newly created language gained native features such as formatting, validation based on the constraints described in grammar and syntax highlighting. The plugin could be tested due to the native integration provided by Xtext with the EMF (Eclipse Modeling Framework), a set of Eclipse features to represent models and generate equivalent code.

Key-words: Database. Database Design and Modeling. Conceptual Modeling. Domain Specific Language.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AST *Abstract Syntax Tree*

BD Banco de Dados

BNF *Backus-Naur Form*

CE Critério de Exclusão

CI Critério de Inclusão

CIM *Computation-Independent Model*

CQ Critério de Qualidade

CSS *Cascading Style Sheets*

DDL *Data Definition Language*

DER Diagrama Entidade-Relacionamento

DML *Data Manipulation Language*

DOM *Document Object Graph*

DQL *Data Query Language*

DSL *Domain Specific Language*

DTL *Data Transaction Language*

EER *Enhanced Entity–Relationship*

EMF *Eclipse Modeling Framework*

ER Abordagem Entidade-Relacionamento

ES Engenharia de Software

GPL *General-Purpose Programming Language*

IDE *Integrated Development Environment*

IHC Interação Humano-Computador

LW *Language Workbench*

MCD Modelo Conceitual de Dados

MDA *Model-Driven Architecture*

MDD *Model-Driven Development*

MDE *Model-Driven Engineering*

MDSE *Model-Driven Software Engineering*

MFD Modelo Físico de Dados

MLD Modelo Lógico de Dados

MLM *Multivocal Literature Mapping*

OMG *Object Management Group*

PIM *Plataform-Independent Model*

CIM *Plataform-Specific Model*

RCP *Rich Client Platform*

SGBD Sistema de Gerenciamento de Banco de Dados

SLM *Systematic Literature Mapping*

SQL *Structured Query Language*

SysML *Systems Modeling Language*

TCC Trabalho de Conclusão de Curso

UML *Unified Modeling Language*

VHDL *VHSIC Hardware Description Language*

XML *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

É praticamente impossível administrar o mundo moderno sem software. As infraestruturas de empresas nacionais e multinacionais são controladas por sistemas baseados em computadores e a maioria dos produtos elétricos inclui algum computador ou software de controle. A fabricação e distribuição industrial é totalmente informatizada, assim como o sistema financeiro. Entretenimento, incluindo a indústria criativa formada por jogos, música e cinema tem intensiva participação de software em suas atividades. Portanto, a Engenharia de Software (ES) é essencial para o funcionamento das sociedades nacionais e internacionais (SOMMERVILLE, 2011).

Contudo, a ES é inviável sem a persistência e manipulação de dados. Nos primórdios do uso de computadores, a persistência de dados se dava em forma de arquivos de texto simples, inspirado inicialmente nas raízes de uma invenção muito antiga, denominada máquina de tabulação, criada por volta de 1860. Todavia em pouco tempo, e em razão da evolução das tecnologias, o uso de arquivos de texto começaram a se mostrar ineficientes (SILBERSCHATZ; KORTH; SUDARSHAN, 1999).

Uma alternativa ao problema relacionado aos arquivos de texto, surgida há mais de 40 anos atrás e ainda utilizada amplamente nos dias atuais, foi justamente a automatização do conceito das máquinas de tabulação. Este feito foi realizado pelo cientista da computação Charles Bachman, pesquisador ligado a indústria e ganhador do prêmio ACM Alan Turing de 1973 pela sua fundamental contribuição para a área de Banco de Dados (BD) (KRISHNA, 1992).

Devido ao alto volume de informação que é produzido e manipulado, os BDs fundamentados inicialmente por Bachman são essenciais na sociedade contemporânea. Segundo Date e Warden (1990), uma base de dados é uma coleção de dados operacionais armazenados, usados pelos sistemas de aplicação de uma determinada organização. Para Elmasri e Navathe (2011), um BD pode ser definido como uma abstração do mundo real, também chamado de minimundo, uma vez que representa aspectos que, em conjunto, carregam um significado implícito.

Analisando de forma mais objetiva, as bases de dados podem ser consideradas os ativos organizacionais mais importantes atualmente. Isso se deve ao fato de que não armazenam apenas informações triviais mas, por exemplo, também dados de faturamento, pesquisas de mercado e outros aspectos que auxiliam na tomada de decisão. Contudo a sua importância não está focada apenas nas organizações, sendo também possível atestar que o uso de BD pode representar um papel crítico na vida dos usuários finais quando estes são analisados individualmente.

Com este cenário estabelecido, é notável que existe um dever crescente da academia em fornecer um bom nível de preparo para os futuros profissionais que vão ingressar em uma indústria cada vez mais exigente. Com frequência as instituições de ensino superior abordam a área de BD com disciplinas específicas e também em componentes curriculares

que são convergentes nos currículos de seus cursos. Neste contexto, a UNIPAMPA possui cursos que envolvem desenvolvimento de software, tais como Engenharia de Software e Ciência da Computação.

1.1 Motivação

Segundo Salgado e Medeiros (1995), o ensino na área de BD é parte essencial da formação de profissionais de computação. O foco no ensino em BD geralmente é dividido em quatro etapas: projeto e modelagem, sistemas de gerência de bases de dados, estudos comparativos entre estes sistemas e o desenvolvimento de aplicações.

Tendo como premissa que existe uma crescente busca por instrumentos que apoiem o processo de ensino-aprendizagem na academia, este estudo tem foco na primeira etapa. O ensino de projeto e modelagem de BD em geral é conduzido com a apresentação de tópicos essenciais e a posterior introdução ao uso de ferramentas de modelagem que utilizam abordagens geralmente gráficas. Este estudo tem como motivação oferecer um produto de software que dê apoio a esta fase. Este produto fará uso da abordagem textual, tendo em vista que possua uma gramática de fácil uso e compreensão.

1.2 Objetivos

O objetivo geral deste trabalho é propor a especificação e realizar a implementação de uma Linguagem Específica de Domínio, do inglês *Domain Specific Language* (DSL), para o projeto e modelagem de BDs relacionais. O foco da modelagem é em nível conceitual, mas deseja-se que a solução possa também realizar a transformação de modelos conceituais para os modelos lógico e físico.

Para atingir o objetivo geral proposto, é fundamental que exista a divisão do problema nos seguintes objetivos específicos que precisam ser atingidos:

- Pesquisar a literatura visando encontrar propostas que façam proposições que forneçam apoio à modelagem de bases de dados;
- Investigar tecnologias que auxiliem no processo de criação de linguagens específicas de domínio;
- Compreender quais são os requisitos necessários para a criação da linguagem;
- Desenvolver uma gramática para modelagem conceitual de BDs relacionais;
- Integrar a linguagem proposta em uma ferramenta *open source*;
- Implementar a transformação do modelo conceitual para o modelo lógico;
- Criar geradores para tecnologias específicas de BDs, representando assim o modelo físico;

- Realizar a avaliação da solução proposta;
- Contribuir com uma ferramenta que auxilie no processo de ensino de projeto e modelagem de BDs.

1.3 Questão de Pesquisa

Visando a condução do restante deste projeto de Trabalho de Conclusão de Curso (TCC) foi definida uma Questão de Pesquisa (QP) central, descrita a seguir:

QP: Uma DSL textual pode auxiliar, em nível conceitual de modelagem, o ensino de projeto e modelagem de banco de dados relacionais?

1.4 Justificativa

Desde os primórdios os desenvolvedores utilizam texto para especificar produtos de software. As linguagens de programação aumentam o nível de abstração de maneira similar aos modelos. Logo, por consequência lógica, isso resulta em linguagens de modelagem textual.

Uma linguagem de modelagem textual é geralmente processada por mecanismos que transformam as informações expressas em formato textual para modelos. Esses mecanismos baseiam sua execução na estrutura sintática de uma linguagem de modelagem textual, que é formalizada em uma gramática. Uma gramática define palavras-chave de uma linguagem, o aninhamento de seus elementos e também a notação de suas propriedades. Dito isto, pode-se inferir que os modelos textuais podem trazer diversos benefícios:

- **Transmitir muitos detalhes**: Quando se trata de elementos com inúmeras propriedades, a abordagem textual geralmente se destaca em relação aos gráficos. O mesmo pode ser dito quanto a estruturas formadas por um grande número de partes muito pequenas, como operações matemáticas ou sequências de instruções;
- **Coesão**: Um modelo textual geralmente especifica os elementos inteiramente em um só local. Se por um lado isso pode ser uma desvantagem para uma exibição em alto nível, em contrapartida pode facilitar a localização de definições de propriedades em baixo nível. Na proposta deste trabalho toda a modelagem conceitual é realizada em apenas um arquivo;
- **Edição rápida**: Durante a criação e edição de modelos textuais não é necessário a recorrente alternância entre teclado e *mouse*. Logo, é provável que se gaste menos tempo formatando modelos textuais do que, por exemplo, refinando a posição, as ligações ou mesmo as bordas de elementos em diagramas;
- **Editores genéricos**: Não existe necessariamente a exigência de uma ferramenta específica para criar ou modificar modelos textuais, como é o caso de linguagens

específicas de domínio com esta abordagem. Para alterações simples é possível o uso de qualquer editor de texto genérico. Entretanto, para tarefas maiores é melhor se ter algum suporte para a linguagem de modelagem. Na proposta desse trabalho está incluso a integração da linguagem definida com um editor Eclipse, fornecendo assim um alto nível de auxílio para a escrita.

1.5 Organização

Este trabalho está organizado da seguinte forma. No Capítulo 2 é apresentado a metodologia de pesquisa adotada. No Capítulo 3 é realizada a fundamentação teórica. Um mapeamento multivocal de literatura tem o protocolo e condução descritos no Capítulo 4. A seguir, ocorre a apresentação da proposta no Capítulo 5. Finalmente, as considerações preliminares são discutidas no Capítulo 6, em que ainda são apontados os trabalhos futuros e o cronograma de execução para continuação da pesquisa.

2 METODOLOGIA DA PESQUISA

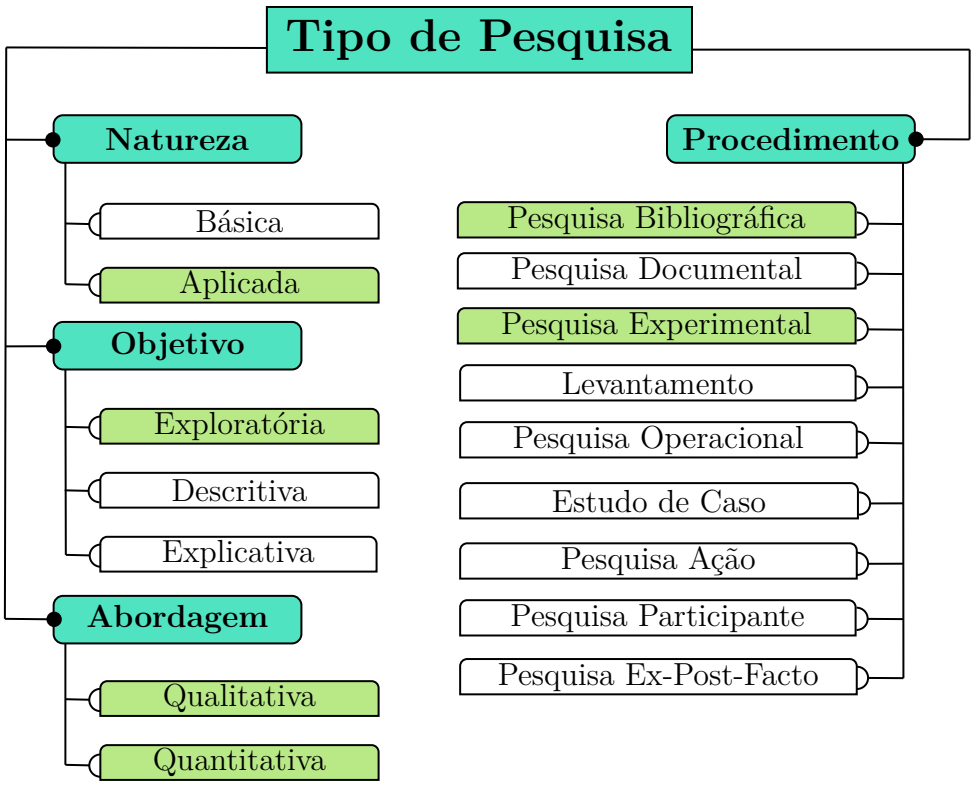
Este capítulo aborda a metodologia de pesquisa adotada para a execução deste estudo. Na Seção 2.1 são elencadas as classificações dos tipos de pesquisa. Os desenhos de pesquisa do TCC I e II utilizados para a execução da pesquisa são apresentados na Seção 2.2 e, em seguida, na Seção 2.3 as lições do capítulo são pontuadas.

2.1 Classificação de Pesquisa

Segundo Wazlawick (2017), as monografias geralmente possuem um capítulo para a elucidação da metodologia. Contudo, analisando-se semanticamente o termo, a metodologia seria o estudo dos métodos. Apesar do uso corrente, linguisticamente seria mais correto afirmar que um trabalho científico individualmente tem um método de pesquisa, ou desenho de pesquisa, utilizando abordagens, técnicas e procedimentos diversos combinados.

Em geral, para que um estudo possa ter maior confiança no que diz respeito ao rigor científico, é imprescindível que sejam identificadas as atividades e técnicas necessárias que possibilitem a se chegar no objetivo proposto (PEFFERS et al., 2007). A Figura 1 mostra a classificação desse estudo mediante sua natureza, abordagem, objetivos e procedimentos.

Figura 1 – Classificação da Pesquisa.



Fonte: Adaptado de Prodanov e Freitas (2013).

Para Prodanov e Freitas (2013), nenhum tipo de pesquisa é autossuficiente, sendo então necessário a mescla de diferentes tipos, tendo um ou outro ponto mais acentuado, para a obtenção de resultados satisfatórios. Os métodos escolhidos determinam os procedimentos que devem ser utilizados, tanto na coleta de dados e informações quanto na análise.

No que se refere a natureza uma pesquisa pode ser básica ou aplicada. Este trabalho objetiva desenvolver uma DSL, gerando uma ferramenta prática para solucionar algo específico e, sendo assim, pode ser categorizada como uma pesquisa de natureza aplicada.

Do ponto de vista dos seus objetivos, uma pesquisa pode ser classificada como exploratória, descritiva ou explicativa. Este estudo se encontra na fase preliminar e tem como finalidade oferecer mais informações sobre o assunto que investiga, logo, é uma pesquisa exploratória em sua fase de concepção.

A abordagem do problema pode classificar a pesquisa em quantitativa ou qualitativa. Este trabalho aplicará conceitos de ambas as categorias. É previsto que seja realizada uma avaliação preliminar da gramática da DSL, onde deverá ocorrer a interpretação dos acontecimentos e a atribuição de significados aos resultados sem requerer necessariamente o uso de métodos e técnicas estatísticas, caracterizando assim essa atividade como uma pesquisa qualitativa. Por outro lado, na análise dos resultados da avaliação experimental da solução construída entende-se que será necessário considerar todos os elementos passíveis de serem quantificados, o que significará explicar em números as opiniões e dados para classificar os resultados. Desta forma, será fundamental o uso de recursos e de técnicas estatísticas, definindo assim essa atividade como uma pesquisa quantitativa.

E por fim, com relação a seu procedimentos técnicos, uma pesquisa pode ser categorizada como pesquisa bibliográfica, documental, experimental, levantamento, também chamada de *survey*, pesquisa operacional, estudo de caso, pesquisa *ex-post-facto*, pesquisa-ação e pesquisa participante. Este trabalho realiza uma pesquisa bibliográfica nas atividades executadas para o levantamento de sua base teórica e, além disso, também pretende executar uma pesquisa experimental na etapa de avaliação da ferramenta produzida.

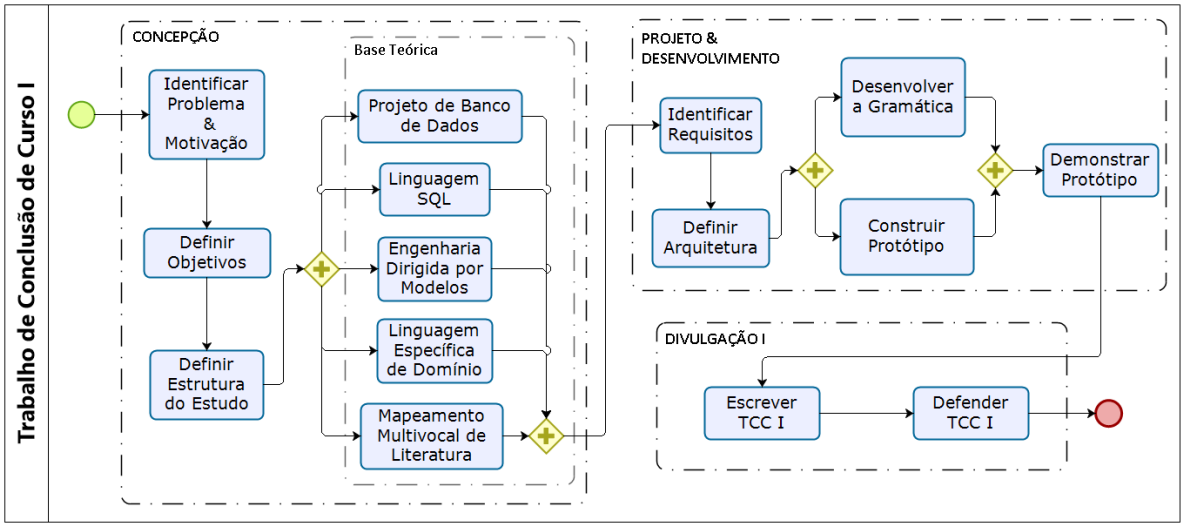
2.2 Desenho de Pesquisa

Para a condução deste estudo foi definido o desenho de pesquisa. Nesta atividade, três fases foram determinantes: a fase decisória, a fase construtiva e a fase redacional.

A fase decisória se refere a escolha do tema e a delimitação dos problemas de pesquisa. A fase construtiva foi onde ocorreu o planejamento das atividades que devem ser realizadas. A fase redacional se refere à escrita do projeto de trabalho e a futura análise de dados que serão colhidos e analisados no andamento do estudo. O desenho de pesquisa foi dividido em duas fases, sendo o TCC I apresentado na Figura 2 e o TCC II

na Figura 3.

Figura 2 – Desenho de pesquisa.



Fonte: O autor.

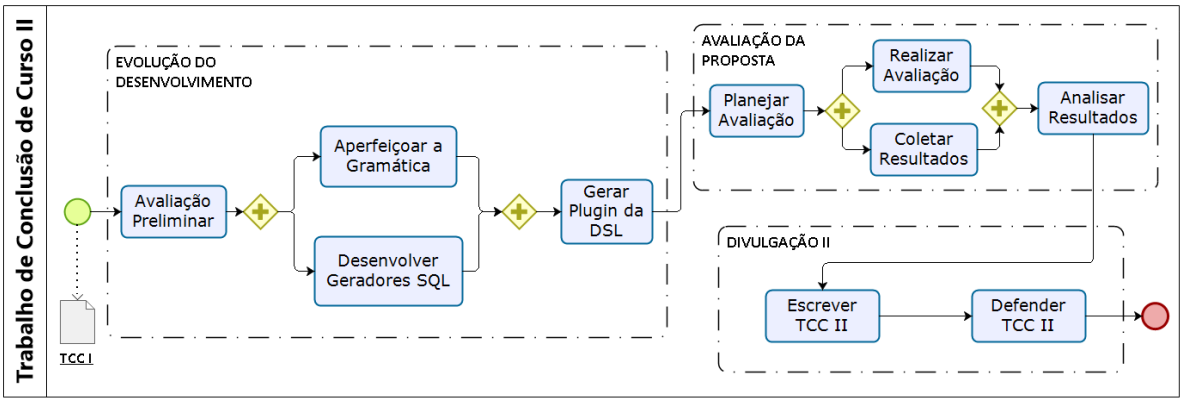
O projeto foi desenvolvido no TCC I, possuindo atividades bem definidas. Primeiramente aconteceu o processo de concepção, identificando o problema e a motivação que guiariam todo o estudo. Após, foram definidos os objetivos que este trabalho procura atingir. Então, houve a elaboração da estrutura geral do estudo em que percebeu-se que era necessário criar uma base teórica bem fundamentada. Desta forma, foram caracterizadas as principais áreas que deveriam ser investigadas para o entendimento do domínio do problema. Em paralelo, foi realizado um mapeamento multivocal de literatura com o objetivo de encontrar trabalhos similares a esta proposta.

Com a etapa teórica de concepção realizada, partiu-se então para o projeto inicial da proposta de DSL, visando identificar seus requisitos essenciais e sua arquitetura básica. Feito isto, deu-se início ao desenvolvimento da descrição de gramática da linguagem e sua implementação de fato. Ao fim, um protótipo da ferramenta foi construída. A demonstração de uso do protótipo está descrita no Capítulo Capítulo 5. Como última etapa do TCC I, foi realizada a especificação do projeto para a apresentação da proposta.

O TCC II tem como entrada os resultados do TCC I e inicia-se com uma avaliação preliminar da gramática criada para a DSL. Após, ocorre a evolução do desenvolvimento, com o aperfeiçoamento da gramática e a construção dos geradores para instruções de Linguagem de Consulta Estruturada, do inglês *Structured Query Language* (SQL). Com essas atividades realizadas, um *plugin* será gerado e incorporado em um *Rich Client Platform* (RCP).

Com o desenvolvimento finalizado, é desejado que se execute uma avaliação da ferramenta. Essa fase vai envolver o planejamento da avaliação para sua realização e

Figura 3 – Desenho de pesquisa (continuação).



Fonte: O autor.

coleta de dados. Finalmente, será realizada a análise dos resultados, a escrita do TCC II e sua posterior defesa. A Tabela 1 apresenta o resumo geral deste estudo.

Tabela 1 – Síntese do TCC.

Assunto	Banco de Dados
Tópico	Projeto e Modelagem de Banco de Dados
Questão de Pesquisa	Uma DSL textual pode auxiliar, em nível conceitual de modelagem, o ensino de projeto e modelagem de banco de dados relacionais?
Objetivo Principal	Propor a especificação e realizar a implementação de uma DSL para o projeto e modelagem de banco de dados relacionais.

Fonte: O autor.

2.3 Lições do Capítulo

Este capítulo forneceu uma ideia do que é a metodologia de um modo geral e de que forma a pesquisa deste trabalho pode ser classificada. Além disso, foi apresentado o desenho estabelecido para a pesquisa, sendo que o que aborda o TCC I fornece o entendimento de quais processos foram executados até o momento, e o desenho de TCC II aponta o que é previsto para a sua continuação.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentado de forma abrangente os domínios abordados para a realização e compreensão deste trabalho. A Seção 3.1 expõe o que é um projeto de BDs e suas fases de modelagem para a construção de esquemas de BDs. A Seção 3.2 elucida a aplicação da SQL, e exemplifica seu uso com alguns exemplos de Sistemas de Gerenciamento de Bancos de Dados (SGBDs). A Seção 3.3 aborda a Engenharia Dirigida por Modelos, do inglês *Model-Driven Engineering* (MDE), e suas aplicações com as DSLs. Na Seção 3.4 é discutido os trabalhos relacionados e, por fim, na Seção 3.5 é apresentado algumas lições aprendidas e pontuados os principais tópicos discutidos.

3.1 Projeto de Banco de Dados

Segundo Date (2004) um BD é fundamentalmente um sistema computacional para a manutenção de registros. Sistemas desse tipo tem a finalidade de armazenar dados de forma persistente, bem como permitir que usuários definam, busquem e atualizem esses dados para gerar informações pertinentes quando necessário. Um BD pode ser representado por um modelo de dados, expressado em diferentes níveis e com diferentes técnicas.

Normalmente durante o ciclo de desenvolvimento de software os modelos de dados passam por níveis distintos de transformações. Inicialmente não existia um padrão ou recomendação difundida amplamente na indústria, ou mesmo na academia, para o processo de modelagem de dados. A estratégia para a utilização de diferentes níveis de projeto e representação tem suas origens com o grupo de estudos em SGBDs intitulado *ANSI/X3/SPARK*, ainda na década de 1970 (ANSI, 1975).

Na abordagem proposta, o padrão de definição e especificação de parâmetros e elementos que compreendiam um BD levavam em consideração aspectos conceituais, lógicos e físicos. Esses aspectos eram chamados genericamente de esquemas (do inglês, *schemas*). Esses esquemas na realidade eram fragmentos que serviam, quando em conjunto, para todo o mapeamento da estrutura de um BD. Esses mesmos conceitos continuam em aplicação até os dias de hoje na implementação de BDs em SGBDs.

De acordo com Cougo (2013), as dificuldades existentes antes do estabelecimento da arquitetura de três níveis estava essencialmente em um ponto. Um mesmo modelo de dados concebido para uma aplicação necessitava de diferentes implementações quando aplicados aos SGBDs primitivos da época anterior a proposta de três níveis, incluindo modificações significativas no próprio modelo original. Isso ocasionava como resultado esquemas bastante particulares e reflexos significativos no modelo de dados final.

Tendo essa realidade como fato, um mesmo modelo de dados gerado para uma única aplicação poderia necessitar de um grande número de diferentes esquemas para abranger as variações de modos de implementação e de visões externas a serem disponibilizadas aos usuários. As dificuldades provenientes da administração e manutenção de

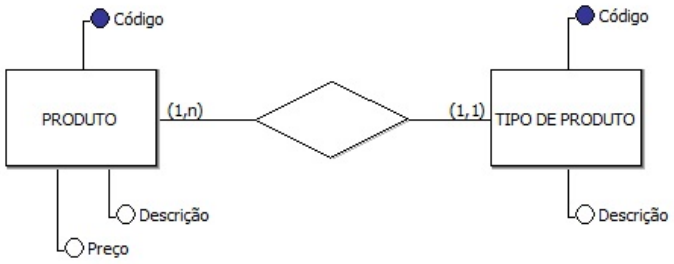
toda essa variedade de modelos levaram o grupo *ANSI/X3/SPARK* a propor o padrão que tem como ideia central a definição de níveis de esquemas relacionados a um modelo de dados (COUGO, 2013). Esse padrão acabou por influenciar a proposta de modelagem conceitual de dados concebida por Chen (1976). Sendo assim, os BDs relacionais até os dias atuais continuam levando em consideração estes conceitos.

A construção de um BD é baseada em um modelo de BD, o qual é uma descrição detalhada dos tipos de informações que devem ser armazenadas. O projeto de BD acontece em três fases distintas de modelagem, onde são gerados o Modelo Conceitual de Dados (MCD), Modelo Lógico de Dados (MLD) e o Modelo Físico de Dados (MFD) (HEUSER, 2009). Para a elaboração de modelos de dados deve ser usado uma linguagem de modelagem de dados. Existem linguagens gráficas e textuais capazes de descrever os modelos em diferentes níveis de detalhamento e abstração.

3.1.1 Modelo Conceitual de Dados

O MCD é a descrição do BD de forma independente da implementação utilizada em um SGBD. Este modelo lista quais dados podem ocorrer no BD, mas não registra como estes dados estão armazenados no nível de SGBD. A técnica mais difundida de MCD é a Abordagem Entidade-Relacionamento (ER) de Chen (1976). Esta abordagem foi tão bem aceita que passou a ser considerada uma referência definitiva para a modelagem de BDs relacionais. É composta basicamente por um método de diagramação e de conceitos que devem ser respeitados. Sendo assim, com esta abordagem os MCDs são representados com Diagramas Entidade-Relacionamento (DERs), como pode ser visto na Figura 4.

Figura 4 – Fragmento de DER.



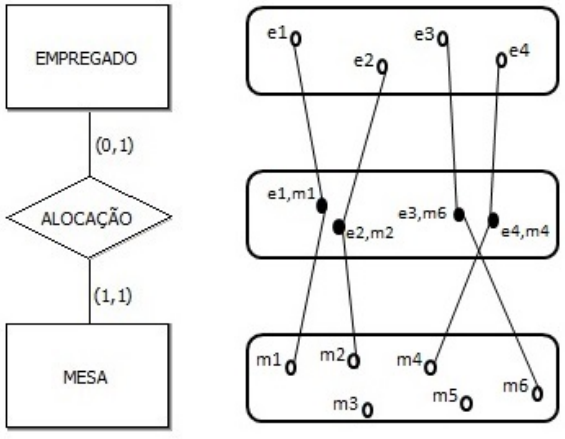
Fonte: Heuser (2009).

Na abordagem ER o conceito principal é o de **entidade**, o qual é uma representação de um conjunto de objetos do domínio modelado. As entidades são simbolizadas por retângulos. Contudo, nesta abordagem ainda existem também outros conceitos que são essenciais e devem ser analisados.

Os **relacionamentos** representam a associação entre os objetos e são sinalizados por losangos. A **cardinalidade** de relacionamentos registra o número de ocorrências

com que as entidades podem se associar. Existem duas cardinalidades que devem ser atribuídas: a mínima e a máxima. Os **atributos** são características representadas por pequenos círculos conectados as entidades. Na esquerda da Figura 5 é ilustrado um exemplo de DER simples e, na direita, uma representação hipotética do conjunto de ocorrências que podem acontecer entre suas entidades.

Figura 5 – Relacionamentos e cardinalidades.



Fonte: Heuser (2009).

3.1.2 Modelo Lógico de Dados

Um MLD é definido como um modelo que possui a representação dos objetos, relacionamentos e características de acordo com regras de implementação. Isso significa que esse modelo tem um nível de abstração do ponto de vista do usuário de um SGBD. Ainda assim, o modelo lógico é independente do tipo de SGBD em que é implementado. Na Figura 6 é apresentado um MLD gráfico baseado no modelo da Figura 4.

Figura 6 – Exemplo de MLD.

Produto

CodProd	DescrProd	PrecoProd	CodTipoProd
1	PC Desktop X	2.500,00	1
2	PC Desktop Y	3.500,00	1
3	Impressora X	500,00	2
4	Impressora Z	1.500,00	2

n 1

TipoDeProduto

CodTipoProd	DescrTipoProd
1	Computador
2	Impressora

Fonte: Heuser (2009).

Esse tipo de modelo deve necessariamente respeitar conceitos tais como chaves de acesso, controle de chaves duplicadas, normalização, integridade referencial, controle de redundância de dados, entre outros. Este modelo é intrinsecamente relacionado a fase

de projeto (COUGO, 2013). É importante salientar que essa é uma forma direcionada a um aspecto gráfico, porém existem meios alternativos de se representar estruturalmente o mesmo modelo de forma textual (MARTELLI; FILHO; CABRAL, 2018). Isso é mostrado na Figura 7 que apresenta a definição da mesma estrutura descrita na Figura 6.

Figura 7 – Estrutura de um modelo lógico descrita de forma textual.

```
TipoDeProduto (CodTipoProd, DescrTipoProd)
Produto (CodProd, DescrProd, PrecoProd, CodTipoProd)
CodTipoProd referencia TipoDeProduto
```

Fonte: Heuser (2009).

A obtenção de um MLD se dá mediante a aplicação de regras de derivação sobre um MCD já construído. Entretanto, não é raro que desenvolvedores e analistas experientes comecem diretamente pelo processo de modelagem lógica, ignorando a modelagem conceitual. Isso ocorre pois esse modelo não se preocupa somente com a representação dos objetos observados no domínio analisado, mas também com outros elementos como chaves, métodos de acesso, formatos de campo, etc. Isso implica, em uma última observação, que do ponto de vista formal da definição da abordagem ER, esse modelo não se enquadra fielmente como um modelo ER (WEST, 2011).

3.1.3 Modelo Físico de Dados

Um MFD caracteriza-se como um modelo em que a representação dos objetos de BD já estão em um nível físico de implementação das ocorrências, ou instâncias, das entidades de relacionamentos. Cada SGBD pode definir diferentes modos de implementação física das características e recursos indispensáveis para o armazenamento e manipulação das estruturas de dados (COUGO, 2013).

Em geral os modelos físicos apresentam dois aspectos bem representados. Primeiramente, existem as ocorrências ou instâncias, seus relacionamentos e a disposição básica dos elementos. O outro aspecto diz respeito a alocação nos diversos níveis de agrupamentos possíveis, como as tabelas, linhas (registros), colunas (campos) e blocos (WEST, 2011). Em suma é uma sequência lógica de instruções em SQL, executadas em um SGBD.

3.2 Linguagem SQL

Para a manipulação dos dados a SQL é a linguagem padrão utilizada por sistemas de BDs relacionais disponíveis no mercado. A SQL teve sua gênese originalmente nos laboratórios da IBM Research, na década de 1970, com o nome inicial de SEQUEL (CHAMBERLIN; BOYCE, 1974). A SQL é uma linguagem com a versão estável mais recente lançada em 2016 e denominada `SQL:2016`, possuindo um total de mais de 2000

páginas de especificação[1,2,3,4]. Entretanto, é importante salientar que ao mesmo tempo em que a maioria dos produtos do mercado trabalham com a SQL, estas soluções também deixam de oferecer suporte a determinados aspectos ou ainda os implementa de uma forma diferente da especificação oficial.

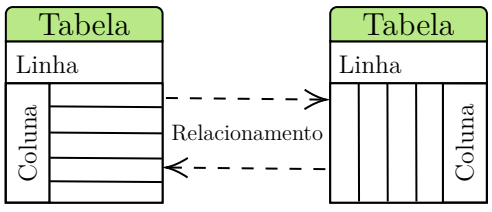
A SQL é uma única linguagem, mas comumente é categorizada conforme a funcionalidade das suas instruções. A primeira categoria é chamada Linguagem de Definição de Dados, do inglês *Data Definition Language* (DDL). Entre os comandos dessa categoria estão o `CREATE`, `ALTER`, `DROP` e `TRUNCATE`. A segunda categoria é nomeada de Linguagem de Definição de Dados, do inglês *Data Manipulation Language* (DML). Entre os comandos categorizados como DML estão o `INSERT`, `UPDATE` e `DELETE`. A terceira categoria é chamada de Linguagem de Consulta de Dados, do inglês *Data Query Language* (DQL), a qual possui apenas o comando `SELECT`. A quarta categoria é denominada Linguagem de Transação de Dados, do inglês *Data Transaction Language* (DTL), a qual detém comandos como `COMMIT` e `ROLLBACK`.

A SQL ainda utiliza uma série de cláusulas (*e.g.* `FROM`, `WHERE`, `GROUP BY`, `ORDER BY`, `HAVING`, `DISTINCT`, `UNION`), operadores lógicos (*e.g.* `AND`, `OR`, `NOT`), operadores relacionais (*e.g.* `<`, `>`, `<=`, `>=`, `=`, `<>`) e funções de agregação (*e.g.* `AVG`, `SUM`, `COUNT`, `MAX`, `MIN`). A aplicação destes termos e palavras reservadas, associado a características inspiradas na álgebra relacional, fundamenta a base da SQL utilizada pelos SGBDs.

3.2.1 Bancos de Dados Relacionais

Os SGBDs relacionais suportam o modelo de dados relacional. O modelo de dados relacional foi proposto por Codd (1970) e tem como premissa a modelagem orientada a tabelas. Um ponto importante a ser citado é o fato de praticamente todos os SGBDs relacionais do mercado utilizarem SQL para a criação e manipulação dos dados.

Figura 8 – Modelo de dados relacional.



Fonte: O autor.

A Figura 8 mostra o esquema estrutural de um modelo relacional. Nele uma tabela,

[1] https://iso.org/standard/63555.html
[2] https://standards.iso.org/ittf/PubliclyAvailableStandards/c065143_ISO_IEC_TR_19075-5_2016.zip
[3] https://standards.iso.org/ittf/PubliclyAvailableStandards/c067367_ISO_IEC_TR_19075-6_2017.zip
[4] https://standards.iso.org/ittf/PubliclyAvailableStandards/c069776_ISO_IEC_TR_19075-7_2017.zip

também chamada de entidade, é definida por um nome e um número fixo de atributos com seus tipos de dados indicados. Cada tabela deve ter um ou mais atributos identificadores, chamado de chaves, o qual visa auxiliar na integridade referencial dos dados. Um registro, também chamado de ocorrência ou tupla, corresponde a uma linha na tabela e consiste nos valores de cada atributo. Uma relação, portanto, consiste em um conjunto de registros associados entre tabelas (RAMAKRISHNAN; GEHRKE, 2003).

3.2.1.1 MySQL

O MySQL é um SGBD relacional que foi criado por uma empresa sueca chamada MySQL AB, e atualmente é desenvolvido e mantido pela Oracle. O desenvolvimento original do MySQL começou em 1994, mas a primeira versão do MySQL foi lançada apenas em maio de 1995. Foi inicialmente criada para uso pessoal do SGBD mSQL baseado na linguagem de baixo nível. O MySQL é usado por muitos aplicativos da *Web* orientados a BD como o Drupal, o Joomla, o phpBB e o WordPress.

Entre suas funcionalidades mais relevantes estão o suporte multiplataforma, suporte SSL, *stored procedures*, *triggers*, *views* atualizáveis, *subselects*, entre outras. Atualmente, o MySQL está na versão 8.0, funciona sobre plataformas Windows, Linux, Solaris, macOS e FreeBSD. Existe a versão paga *Enterprise Server* e a versão de código aberto MySQL *Community Server*, gratuita e com licença GPL. É considerado o 2º SGBD mais popular [5] entre as opções do mercado pelo portal DB-Engines. Segundo a avaliação publicada em 2019 pela empresa de consultoria Gartner *Group*[6], o MySQL possui uma avaliação de 4,5 de 5 pelo mercado.

3.2.1.2 Microsoft SQL Server

O Microsoft SQL Server é um SGBD relacional desenvolvido e mantido pela Microsoft. Teve uma versão de teste foi criada em parceria com a Sybase em 1988, mas sua primeira versão para uso comercial foi lançada em abril de 1989. Desde de seu lançamento este SGBD sofreu inúmeras melhorias. Atualmente possui diversas versões disponibilizadas no mercado, como a *Enterprise*, a *Standard*, *Web*, *Business Intelligence* e *Workgroup*.

Também possui uma versão chamada *Express*, uma edição gratuita e reduzida. Essa versão inclui o mecanismo de BD principal e, embora não haja limitações quanto ao número de BDs ou usuários com suporte, ele é limitado ao uso de um processador, 1 GB de memória e 10 GB de arquivos de BD. O Microsoft SQL Server pode funcionar sobre plataformas Linux, Microsoft Windows Server e Microsoft Windows. Atualmente está na versão SQL Server 2017 e é considerado o 3º SGBD mais popular [7] no mercado

[5] https://db-engines.com/en/system/MySQL
[6] https://gartner.com/reviews/market/operational-dbms
[7] https://db-engines.com/en/system/Microsoft+SQL+Server

pelo portal DB-Engines. Segundo a avaliação publicada pela Gartner Group, o Microsoft SQL Server possui uma avaliação de 4,4 de 5 pelo mercado em 2019.

3.2.1.3 PostgreSQL

O PostgreSQL é um SGBD que incorpora o modelo relacional para seus esquemas de dados e suporta a linguagem de consulta padrão SQL. Surgiu dentro do projeto do Ingres, outro SGBD, na universidade da Califórnia. Lançado em 1996 e mantido atualmente pelo PostgreSQL Global Development Group, é considerado pelo mercado um SGBD estável, abrangente e possuidor de boas características de desempenho. É executado em praticamente qualquer plataforma Linux, macOS e Microsoft Windows. O grande diferencial do PostgreSQL para ser um dos SGBDs de maior sucesso é o fato de ser gratuito e de código aberto por meio da flexível licença BSD.

Entre algumas características suportadas pelo PostgreSQL estão transações, *sub-selects*, *views*, integridade referencial de chaves estrangeiras, bloqueios sofisticados, tipos definidos pelo usuário, herança, regras variadas, *triggers*, funções, procedimentos armazenáveis, entre outras. Atualmente está na versão 11.3 e é considerado o 4º mais popular [8] entre os SGBDs existentes pelo portal DB-Engines. Segundo a avaliação publicada pela Gartner Group, o Microsoft SQL Server possui uma avaliação de 4,5 de 5 pelo mercado em 2019.

3.3 *Model-Driven Engineering*

Conceitualmente um modelo é uma representação, protótipo ou exemplo que se tem por objetivo reproduzir ou imitar de alguma forma. A construção de modelos são pontos centrais e importantes em diferentes áreas científicas. Na matemática, física e química, por exemplo, o emprego de modelos é tido como vital para a investigação teórica e prática em diferentes campos de estudo (BAILER-JONES, 2009).

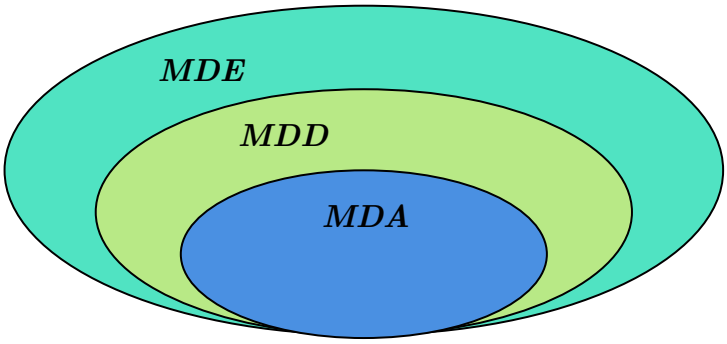
Em uma análise mais profunda, Brambilla, Cabot e Wimmer (2017) discutem que, considerando-se a premissa de que um observador e suas observações alteram a própria realidade, é possível se concluir que tudo na percepção de um indivíduo é um modelo, já que absolutamente nada pode ser processado pela mente humana sem ser modelado. Em resumo, a criação de modelos é uma tarefa de abstração de domínios e conceitos do mundo real. Sendo assim, não é de surpreender que os modelos tenham se tornado cruciais e amplamente adotados também em áreas técnicas como mecânica, engenharia civil e, por fim, na ciência da computação, engenharia da computação e ES.

A MDE, também chamada de *Model-Driven Software Engineering* (MDSE), é uma abordagem da ES para desenvolvimento de *software* que tem essencialmente modelos como saídas principais de algum processo. Essa abordagem resulta em programas ou atividades

[8] https://db-engines.com/en/system/PostgreSQL

de computador executados em *hardware* ou *software* que são gerados automaticamente a partir de modelos (SOMMERVILLE, 2011). Conceitualmente, a MDE fornece apoio a outros conceitos, como a *Model-Driven Development* (MDD) e a *Model-Driven Architecture* (MDA). Na Figura 9 a relação entre estes conceitos é ilustrada.

Figura 9 – Relação de MDE, MDD e MDA.



Fonte: Adaptado de Ameller (2009).

O MDD é um paradigma de desenvolvimento que usa modelos como o principal artefato do processo de desenvolvimento. Normalmente na MDD a implementação é gerada de forma semiautomática a partir dos modelos. Apesar de serem vistas como a mesma coisa, o conceito da MDE tem origem na MDA, proposta em 2001 pelo *Object Management Group* (OMG). Com diferenças sutis, Sommerville (2011) afirma que a MDA concentra-se nos estágios de projeto e implementação do processo de desenvolvimento de software, sendo muito similar ao MDD, porém implementando diretrizes específicas da OMG. Desta forma, conclui-se que a MDA é um subconjunto da MDD. Por outro lado, a MDE pode abordar muitos outros tópicos do processo de ES, entre eles a engenharia de requisitos baseada em modelos, processos de software para desenvolvimento baseado em modelos, ou ainda, testes baseados em modelos.

A MDE, como uma metodologia, auxilia a aplicação das vantagens da modelagem nas atividades de ES. Para (BRAMBILLA; CABOT; WIMMER, 2017) essa abordagem leva em consideração quatro aspectos fundamentais, listados a seguir.

1. **Conceitos:** os componentes que constroem a metodologia, abrangendo desde artefatos de linguagem até atores, e assim por diante;

2. **Notações:** A maneira como os conceitos são representados, ou seja, as linguagens usadas na metodologia;

3. **Processos e Regras:** As atividades que levam à elaboração do produto final, as regras para sua administração e controle, e as afirmações sobre as propriedades desejadas (correção, consistência, etc) dos produtos ou do próprio processo;

4. **Ferramentas:** Aplicações que facilitam a execução de atividades ou seu controle, abrangendo o processo de produção e apoiando o desenvolvedor no uso das notações.

A motivação por trás da MDE é a ideia de se aumentar o nível de abstração do processo de desenvolvimento em geral, para então assim capturar sistemas ou processos como uma coleção de modelos reutilizáveis. Logo, ela visa reduzir a dificuldade associada ao desenvolvimento de sistemas de software, em geral mais complexos, por meio do uso de técnicas de modelagem que suportam a separação de interesses e geração automatizada de artefatos de sistemas a partir de modelos (KLEPPE et al., 2003).

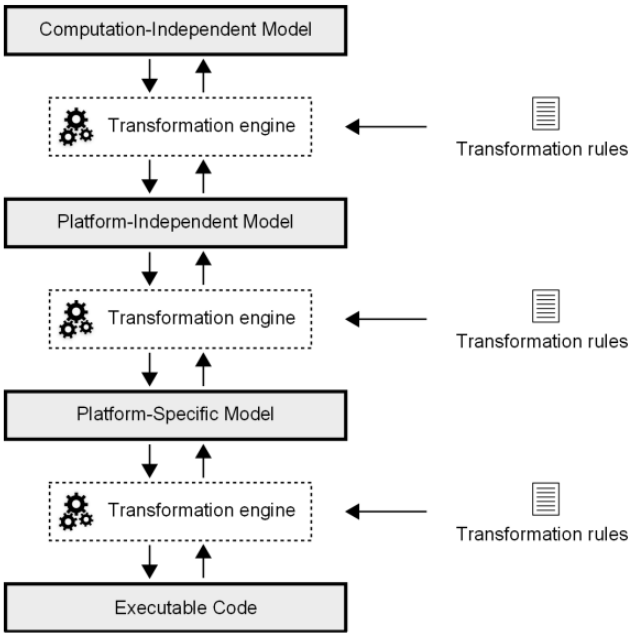
De uma forma objetiva, a abordagem MDA, ou ainda a MDD, é a forma de se realizar a MDE. Essa abordagem define três camadas que devem ser usadas como pilares para todo o processo, listados a seguir. A relação conceitual entre esses níveis, com o o uso de mecanismos de transformação e regras de transformação, é exemplificado na Figura 10.

1. ***Computation-Independent Models* (CIMs):** descrevem objetos de negócio e as atividades independentemente de sistemas de suporte;

2. ***Plataform-Independent Models* (PIMs):** descrevem como os processos de negócio são suportados por sistemas, vistos como caixas-pretas funcionais, ou seja, desconsiderando as restrições associadas as tecnologias candidatas;

3. ***Plataform-Specific Models* (CIMs):** descrevem os componentes do sistema conforme implementados por tecnologias específicas.

Figura 10 – Níveis de abstração do MDA.



Fonte: Frantz (2012).

A separação de interesses da MDA baseia-se, por exemplo, na exploração de diferentes DSLs, cada uma fornecendo construções baseadas em abstrações que são específicas do domínio de um sistema. Por conta disto, as DSLs podem desempenhar um papel de destaque na MDE (SCHMIDT, 2006).

3.3.1 *Domain-Specific Language*

Para Deursen, Klint e Visser (2000) uma DSL é uma linguagem de programação ou linguagem de especificação executável que oferece, por meio de notações e abstrações apropriadas, poder expressivo focado e, geralmente, restrito a um domínio de problema específico. Assim como outras linguagens, as DSLs devem apresentar um conjunto de sentenças bem definidas por uma sintaxe e semântica própria. Para Fowler (2010) uma DSL é definida como uma linguagem de programação de computadores com expressividade limitada e focada em um domínio particular. Entre exemplos conhecidos de DSLs estão:

- SQL, para bancos de dados;
- *Cascading Style Sheets* (CSS), para *layout* de páginas *Web*;
- *eXtensible Markup Language* (XML), para codificação de dados;
- *Unified Modeling Language* (UML), para projeto de software;
- *Systems Modeling Language* (SysML), para modelagem de sistemas;
- *VHSIC Hardware Description Language* (VHDL), para projeto de hardware;
- LaTeX, para tipografia de documentos.

Segundo Faveri (2013), apesar do termo DSL poder intuitivamente remeter para um campo de estudos recente, de fato isso não é uma realidade. Por exemplo, a APT é uma DSL para programação de máquinas controladas numericamente que foi desenvolvida por dois anos a partir de 1957 (ROSS, 1981), enquanto o formalismo de especificação de sintaxe *Backus-Naur Form* (BNF), o mais usado para notação das linguagens de programação nos dias de hoje, remonta o final da década de 1950 (BACKUS, 1959).

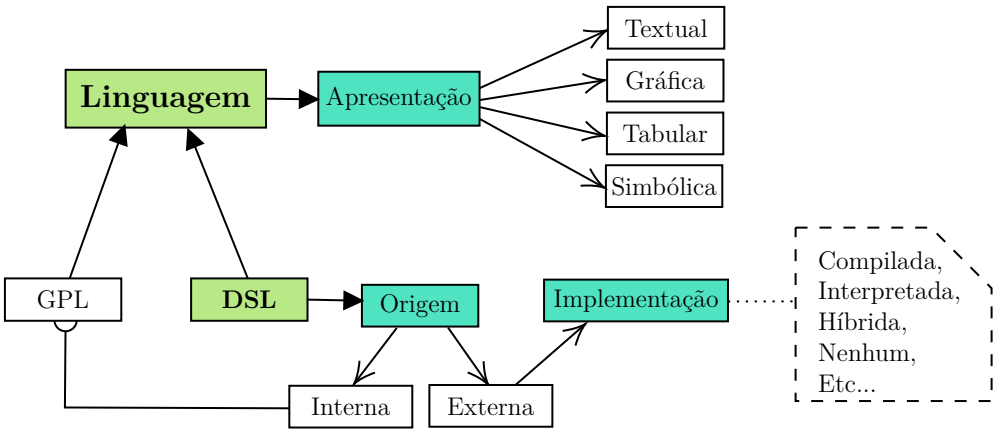
Por conta desse fato é possível encontrar na literatura muitos estudos que abordam conceitualmente DSLs, porém com diferentes terminologias. Entre estas, pode-se citar: *Languages for specialized application* (SAMMET, 1972); *Special-purpose languages* (WEXELBLAT, 1981); *Application Languages* (MARTIN, 1982); *Task-specific programming languages* (NARDI, 1993); *Specialized languages* (BERGIN JR.; GIBSON JR., 1996).

A aplicação de DSLs permite que softwares sejam desenvolvidos de forma mais rápida e eficaz. A maior vantagem observada no uso de DSLs é que o conhecimento necessário para a sua aplicabilidade é abstraído para outro nível. Desta forma, especialistas do domínio podem entender, validar e modificar o código, adaptando o modelo

as suas necessidades, tornando o impacto das mudanças mais fácil de ser compreendido. Ainda existe um aumento significativo na produtividade, confiabilidade, facilidade de uso e flexibilidade (DEURSEN; KLINT; VISSER, 2000).

Segundo Mernik, Heering e Sloane (2005) as DSLs podem ser classificadas sob três dimensões diferentes: **origem**, **aparência** e **implementação**. As dimensões de classificação de DSL são exibidas na Figura 11. Em relação a origem de uma DSL, as opções existentes são as DSLs **internas** e **externas**.

Figura 11 – Dimensões de uma DSL.



Fonte: Adaptado de Faveri (2013).

Uma DSL **interna** é projetada a partir das regras sintáticas e semânticas da gramática de uma linguagem já existente, podendo ser essa uma linguagem de propósito geral, do inglês *General-Purpose Programming Language* (GPL), ou outra DSL. Sendo assim, para seu funcionamento correto uma DSL interna acaba transferindo todas as atividades de verificação léxica, sintática, semântica e de transformação de código ao compilador da linguagem hospedeira.

Uma DSL **externa** é uma linguagem com sintaxe distinta e que depende de uma infraestrutura própria para a análise léxica, sintática, semântica, interpretação, compilação, otimização e geração de código. Se comparada a uma GPL, uma DSL externa possui especificidades similares, porém seus recursos são restritos ao domínio de aplicação para o qual a linguagem é projetada.

No que diz respeito a dimensão de **aparência**, uma DSL pode ser classificada como **textual**, **gráfica**, **tabular** e **simbólica**. Quando no formato textual as DSLs permitem que o domínio seja expressado com caracteres, os quais são então combinados gerando palavras, expressões, sentenças e instruções que seguem as regras gramaticais previamente estabelecidas na linguagem. As DSLs não textuais seguem a mesma lógica, mas utilizando-se de modelos gráficos para permitir que o usuário possa expressar conhecimento de domínio com um maior nível de compreensão e empregando para tal o uso de

símbolos, tabelas, figuras e conectores.

E finalmente, no que se refere a dimensão de **implementação**, as DSLs podem ser classificadas tendo em vista a perspectiva de sua execução. Essas classificações formam quatro grupos: (i) DSLs de execução bem definidas (*e.g.* Excel Macro Language); (ii) DSLs que servem de entrada para geradores de aplicação; (iii) DSLs não executáveis mas úteis como entrada de geradores de aplicação; (iv) DSLs não projetadas para serem executadas.

Em geral o principal aspecto levado em consideração para a construção de uma DSLs deve ser a sua **origem** pois cada abordagem apresenta vantagens e desvantagens específicas que são inerentes a cada tipo (FOWLER, 2010). Apesar das DSLs externas poderem ter um esforço associado a sua construção muitas vezes maior do que o de uma DSL interna, atualmente existem ferramentas que dão grande suporte a construção de DSLs. Estas ferramentas são conhecidas como *Language Workbenches* (LWs) e aplicam conceitos de programação orientada a linguagens, fornecendo um nível de abstração maior no que diz respeito as questões complexas de infraestrutura (FOWLER, 2005).

3.3.2 *Language Workbenches*

O desenvolvimento de uma DSL não é tarefa trivial, pois como são linguagens de programação possuem uma sintaxe que é, por consequência lógica, definida por uma gramática. Desta forma, se faz necessária a utilização de ferramentas que suportem a definição dos conceitos para a nova linguagem (FOWLER, 2005).

Os LWs são ferramentas que fornecem mecanismos de infraestrutura para a implementação de linguagens de programação, tornando assim a criação de linguagens mais acessível (Wachsmuth; Konat; Visser, 2014). Entre os mecanismos fornecidos nesses ambientes está a formatação automática, validação com base nas restrições descritas na gramática, *syntax highlighting*[9] e *syntax completion*[10]. A seguir são citados três dos mais conhecidos LWs da atualidade:

- **Xtext:** lançado em 2006, o Xtext é um *framework* de código aberto para o desenvolvimento linguagens de programação textuais, com integração com o ambiente de desenvolvimento integrado, do inglês *Integrated Development Environment* (IDE), Eclipse. Para especificar uma linguagem, o desenvolvedor descreve uma gramática no Xtext. Essa gramática descreve como um modelo *Ecore* deve ser derivado de uma notação textual. A partir dessa definição, um gerador de código deriva um analisador ANTLR e as classes para o modelo de objetos. O Xtext também tem um gerador Xtend editável, o que dá a capacidade de se gerar código para qualquer outra gramática. O Xtext inclui recursos inerentes ao IDE Eclipse como *syntax highligh-*

[9] Realce de código-fonte com cor, negrito, etc. Serve para indicar sua estrutura sintática.
[10] Uma função, como em um mecanismo de busca, que fornece uma ou mais opções de palavras reservadas previstas na gramática a partir dos caracteres que um usuário já inseriu.

ting, *code completion*, *static analysis*, *source-code navigation* e outros. Atualmente está na versão 2.17.1.

- **JetBrains MPS:** o JetBrains MPS é um sistema desenvolvido pela JetBrains, empresa da República Tcheca, que usa edição projetiva. Essa abordagem permite aos desenvolvedores uma melhor compreensão, o que a diferencia de outros LWs. Também possui funções comuns de IDEs integrado a seu ambiente de desenvolvimento. Está atualmente na versão 2019.1.1 sob a licença Apache 2.0.

- **MetaEdit+:** o MetaEdit+ é LW proprietário desenvolvido pela companhia finlandesa MetaCase para criar e utilizar DSLs. Possui duas versões nomeadamente *MetaEdit+ Workbench* e *MetaEdit Modeler*. O *Workbench* inclui ferramentas para projetar e usar/testar linguagens de modelagem enquanto o Modeler inclui ferramentas para se utilizar linguagens de modelagem. Normalmente, o *MetaEdit+ Workbench* é usado pelos desenvolvedores que projetam uma DSL do domínio para um projeto. Em seguida, essa linguagem de modelagem é usada para desenvolver produtos finais com o apoio do *MetaEdit+ Modeler*. Atualmente está na versão 5.5 SR1.

3.4 Trabalhos Relacionados

Essa seção descreve os trabalhos de maior representatividade para o objeto deste estudo. Uma vez que a proposta envolve a construção de uma ferramenta que implemente uma DSL textual, e após a pesquisa descrita no Capítulo 4 deste estudo, selecionou-se propostas e ferramentas que mais se aproximam do objetivo final deste trabalho.

O trabalho de Dimitrieski et al. (2015), desenvolvido na universidade de Novi Sad na Sérvia, apresenta uma ferramenta chamada *System Modeling Tool* (MIST). Essa ferramenta utiliza uma DSL chamada EERDSL, uma linguagem com base no modelo aprimorado de entidade-relacionamento, do inglês *Enhanced Entity–Relationship* (EER). A MIST apresenta uma abordagem de modelagem bidirecional (gráfica e textual) de modelagem de BDs. O autor discute que tal decisão tem como motivo o entendimento de que preferência sobre a abordagem de modelagem utilizada pode depender do domínio do problema, do conhecimento e das preferências pessoais de um projetista de BD. Apresenta também uma experiência anterior, onde foi construído uma ferramenta de modelagem com uma abordagem baseada em formulários. A partir dos resultados obtidos nesta experiência, foi concebida a ideia da MIST. O propósito da ferramenta é a aplicação tanto no mercado profissional quanto para o ensino de projeto e modelagem de BD no meio acadêmico. A MIST foi desenvolvida com o auxílio do Xtext para a notação da DSL textual, e inicialmente a Eugene e posteriormente a Sirius para a sua versão gráfica. A MIST ainda oferece suporte à geração de código SQL.

O *dbdiagram.io*[11] é uma ferramenta *Web* gratuita para o desenho de DERs, de-

[11] https://dbdiagram.io/

senvolvida por uma empresa de Singapura, com uma abordagem textual que implementa uma DSL própria. Esta DSL utiliza um modelo muito próximo do lógico. O diferencial da ferramenta é sua rápida curva de aprendizagem e, além disso, a apresentação de uma representação gráfica do que está sendo modelado. A apresentação dos elementos do diagrama pode ser organizada livremente pelo usuário em tempo real. Entretanto é importante se salientar que toda a modelagem de fato é feita de modo textual. A ferramenta ainda oferece a geração automática de código SQL.

Da mesma forma, a *QuickDBD*[12], desenvolvida por uma empresa na Irlanda, é uma ferrramenta *Web* com exatamente o mesmo modo operacional que a *dbdiagram.io*, também implementando uma DSL textual própria para modelagem de BDs. Contudo é uma ferramenta proprietária, ou seja, paga e com o foco declaradamente na indústria. Ambas as ferramentas são muito similares também quanto a geração de representações gráficas da modelagem e apresentam diversos argumentos para sua adoção, como a rápida compreensão de suas DSLs, a perspectiva de realização de trabalhos fluídos, o acesso de qualquer plataforma e o compartilhamento dos modelos com outros usuários.

Finalmente, pode-se citar a ferramenta *Web* gratuita *RelaX (Relational Algebra Calculator)*[13]. Esta ferramenta não foi encontrada no mapeamento, mas indicada por um pesquisador da área de DSLs e BDs. Trata-se de uma ferramenta desenvolvida na universidade de Innsbruck, na Áustria, e voltada ao ensino de álgebra relacional fazendo operações sobre bases de dados relacionais. Tem uma abordagem textual, utilizando uma DSL chamada RelAlg, e apresentando inclusive duas perspectivas de operação: instruções de RelAlg e instruções em SQL. A RelaX utiliza uma abordagem de modelo já em nível físico para operações, como as DDLs de construção e DMLs para as consultas. Apesar de suas funcionalidades, a RelaX não se propõe a ser uma ferramenta de projeto e modelagem de BD, mas de uso restrito ao ensino dentro da academia.

3.5 Lições do Capítulo

Os conceitos mais importantes para este trabalho foram apresentados neste capítulo. No geral foi necessário investigar dois grandes domínios, sendo eles: (i) projeto e modelagem de BD e (ii) MDE. Dentre os temas abordados destaca-se a Seção 3.3.1, a qual apresenta definições importantes para a compreensão do que é de fato uma DSL. A Seção 3.3.2 também merece destaque pois cita alguns LWs, dentre eles o Xtext que acabou por ser a ferramenta selecionada para a construção do protótipo da proposta deste estudo.

[12] https://quickdatabasediagrams.com/
[13] https://dbis-uibk.github.io/relax/

4 MAPEAMENTO MULTIVOCAL DE LITERATURA

Esse capítulo descreve os procedimentos adotados para a investigação da literatura realizada para este estudo. Para este propósito ser alcançado foi realizado o planejamento e a execução de um Mapeamento Multivocal da Literatura, do inglês *Multivocal Literature Mapping* (MLM). Para isto foram aplicadas diretrizes bem estabelecidas conceitualmente por diversos trabalhos (KITCHENHAM; CHARTERS, 2007; PETERSEN et al., 2008; NAKAGAWA et al., 2017).

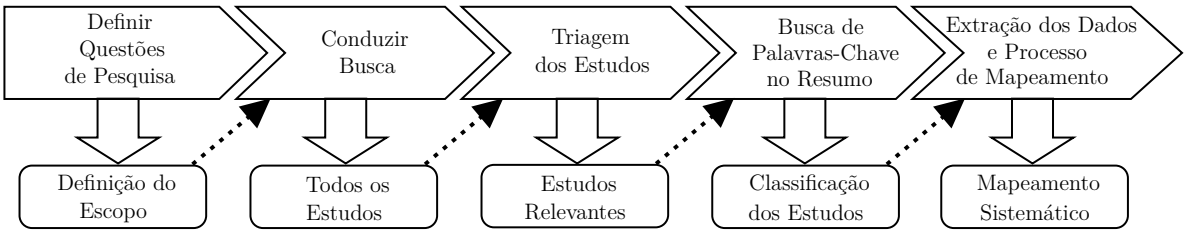
O protocolo seguido é descrito de forma detalhada na Seção 4.1. Na Seção 4.2 é relatado todo o processo de execução do mapeamento multivocal. Os resultados provenientes da execução são analisados e discutidos na Seção 4.3. As ameaças ao MLM são debatidas na Seção 4.4 e, por fim, são pontuadas algumas lições do capítulo na Seção 4.5.

4.1 Protocolo

Um MLM é uma forma de Mapeamento Sistemático de Literatura, do inglês *Systematic Literature Mapping* (SLM), que inclui a literatura cinza. Os MLMs são úteis para pesquisadores e para profissionais pois fornecem visões abrangentes sobre o estado da arte e da prática em uma determinada área. Neste estudo conduziu-se um MLM utilizando o processo de SLM definido por Petersen (PETERSEN et al., 2008) e as diretrizes propostas por Garousi, V. and Felderer, M. and Mäntylä (2019) para pesquisa na literatura cinza.

Um SLM envolve uma busca para determinar que tipos de estudos abordam as questões de pesquisa que se objetiva investigar, além de possibilitar a classificação e extração de dados que possam gerar informações relevantes (Bailey et al., 2007). Com o propósito de ter a obtenção de resultados confiáveis e reproduzíveis em um SLM, é vital que uma série de atividades bem definidas e estruturadas sejam seguidas (NAKAGAWA et al., 2017). Na Figura 12 é apresentado o processo de SLM realizado neste estudo, com base na proposição realizada por Petersen et al. (2008).

Figura 12 – Processo de mapeamento sistemático.



Fonte: Adaptado de Petersen et al. (2008).

4.1.1 Questões de Pesquisa

Levando-se em consideração o objetivo geral deste trabalho, o qual é desenvolver uma DSL para a modelagem conceitual de BDs, foram formulados os seguintes questionamentos que serviram de guia para o restante do MLM:

- **QP1.**: Qual o estado da arte do desenvolvimento de DSLs para transformação de modelos em ER?
- **QP1.1.**: Quais são as metodologias, técnicas e propostas de refinamento (desenho automatizado) baseado em modelos de dados?
- **QP1.2.**: Qual é o ferramental utilizado como apoio ao desenvolvimento dessas DSL?
- **QP1.3.**: Quais são as representações de objetos de BD adotadas ou sugeridas nas DSL propostas?
- **QP2** Quais os métodos de avaliação utilizados nos estudos?
- **QP2.1.**: Quais os pontos positivos e negativos observados na execução dos estudos?
- **QP2.2.**: Quais desafios são apontados após execução dos estudos empíricos?
- **QP3.**: Quais são as ferramentas para modelagem conceitual de BDs?
- **QP3.1.**: Quais são as notações que estas ferramentas usam?
- **QP3.2.**: Quais são os níveis de modelagem de BD (Conceitual, Lógico, Físico) que estas ferramentas suportam?

4.1.2 Fontes de Busca

As bibliotecas digitais são a principal fonte de busca em um SLM (PETERSEN et al., 2008). Para a escolha das fontes de busca desse SLM são considerados três requisitos obrigatórios que as bases devem contemplar: **(I)** possuir mecanismo de pesquisa baseado na *Web*; **(II)** ser capaz de usar palavras-chave durante a pesquisa, e; **(III)** abranger estudos primários da grande área da Ciência da Computação. Na Tabela 2 são listadas as bibliotecas digitais selecionadas para este SLM.

Tabela 2 – Bibliotecas digitais utilizadas.

Fonte	Endereço	Tipo
ACM Digital library	*dl.acm.org*	Híbrida
IEEE Xplore	*ieeexplore.ieee.org*	Base Bibliográfica
ScienceDirect	*sciencedirect.com*	Base Bibliográfica
Scopus	*scopus.com*	Motor de Busca
SpringerLink	*link.springer.com*	Base Bibliográfica

Fonte: Adaptado de Nakagawa et al. (2017).

4.1.3 String de Busca

A elaboração da *string* de busca não é uma tarefa trivial. A identificação de uma combinação de termos que permitam encontrar o maior número de estudos primários relevantes de forma objetiva necessita, na maioria dos casos, de experiência e profundo conhecimento sobre a área de pesquisa abordada.

Tabela 3 – Termos e sinônimos utilizados.

Termos	**Sinônimos**
Domain Specific Language	DSL, Domain-Specific Language, Domain-Specific-Language, DSLM, Domain Specific Modeling Language, Domain-Specific Modeling Language, Domain-Specific-Modeling-Language, Query Language
Entity-Relationship	ER, Enhanced Entity-Relationship, EER, Database

Fonte: O autor.

Para a formação da *string* de busca é fundamental definir um conjunto de palavras referentes ao tema de pesquisa, bem como os sinônimos considerados expressivos. Estes termos devem representar de forma abrangente o tema central do estudo. Para este trabalho foram estabelecidos os termos e sinônimos da Tabela 3, sendo que sua combinação gerou a *string* genérica da Figura 13.

Figura 13 – String genérica de busca.

```
(DSL OR Domain Specific Language OR Domain-Specific Language OR
Domain-Specific-Language OR DSLM OR Domain Specific Modeling Language
OR Domain-Specific Modeling Language OR
Domain-Specific-Modeling-Language OR Query Language) AND (ER OR
Entity-Relationship OR Enhanced Entity-Relationship OR Extended
Entity-Relationship OR Database)
```

Fonte: O autor.

4.1.4 Critérios de Seleção

Segundo Kitchenham e Charters (2007), os Critérios de Inclusão (CIs) indicam por qual ou quais parâmetros um estudo é incluído no SLM, ou seja, considerado relevante. Da mesma forma, os Critérios de Exclusão (CEs) indicam por qual ou quais parâmetros um estudo é excluído, ou seja, considerado não relevante na pesquisa realizada. Para o este trabalho foram determinados os critérios de seleção listados a seguir.

Critérios de Inclusão (CI)

- CI1: Estudo que propõe alguma técnica, método, abordagem ou ferramenta para a representação e transformação de modelos de BD utilizando DSLs.

Critérios de Exclusão (CE)

- CE1: Estudo com menos de 4 páginas;
- CE2: Estudo que não esteja escrito em inglês;
- CE3: Estudo duplicado;
- CE4: Estudo que não fornece acesso completo ao seu conteúdo;
- CE5: Estudo que não atende o CI1.

4.1.5 Avaliação de Qualidade

Foram definidos sete (7) Critérios de Qualidade (CQs) para a avaliação dos estudos primários aprovados após a aplicação dos critérios de seleção. Os CQs visam quantificar a relevância para que seja possível realizar uma comparação entre os estudos selecionados. Foi definido também uma pontuação para ser atribuída a partir das CQs. Para a definição das pontuações foram estabelecidas siglas para representar a pontuação dos CQs.

- **T: Total**, contemplando de forma integral o critério de qualidade avaliado;
- **P: Parcial**, dependendo do peso total do CQ), contemplando parcialmente o critério de qualidade avaliado;
- **N: Negativo**, não contempla de forma nenhuma o critério de qualidade avaliado.

A pontuação máxima possível, avaliados todos os critérios, é dez (10.0) e a mínima zero (0). Cada CQ possui um peso específico ($1 \rightarrow 1.5 \rightarrow 2$) dependendo da sua importância considerada para este estudo. Na Tabela 4 são listadas os CQs e seus respectivos pesos. Estes CQs basearam-se em aspectos que foram consideramos relevantes para o SLM, sendo eles **relato** (QA1, QA4, QA5), **rigor** (QA2, QA3), **credibilidade** (QA2, QA3) e **relevância** (QA1, QA6, QA7).

Para todos os CQs a sigla **N** (Negativo) representa zero (0) e a sigla **T** (Total) representa o conceito máximo ($1 \rightarrow 1.5 \rightarrow 2$). Por outro lado é importante se salientar que apenas os CQs que possuem pontuação um (1) e dois (2) a sigla **P** (Parcial) representa 50%. Nos CQs com peso 1.5 o **P** representa 60% (0.9) da pontuação. Logo após essa definição detalhada dos CQs foi, então, possível dar início a fase de execução do SLM.

4.1.6 Estratégia de Extração de Dados

Definiu-se um formulário de dados para a extração e análise dos dados relevantes contidos nos estudos primários selecionados. A descrição detalhada deste formulário de extração é apresentado na Tabela 5.

Tabela 4 – Critérios de Avaliação de Qualidade.

ID	Descrição	Peso
CQ1.	O estudo apresenta alguma contribuição para a área de modelagem de BD?	1,5
CQ2.	O estudo apresenta metodologias, técnicas ou propostas de refinamento baseado em modelos de dados?	1,5
CQ3.	O estudo apresenta alguma forma de avaliação empírica?	1,5
CQ4.	O estudo apresenta as características do processo de criação da DSL?	1,5
CQ5.	O estudo caracteriza as atividades de transformação do modelo para diferentes diferentes tecnologias de BDs?	2,0
CQ6.	O estudo apresenta pontos positivos e negativos observados em sua execução?	1,0
CQ7.	O estudo aponta desafios decorrentes da sua execução?	1,0

Fonte: O autor.

Tabela 5 – Formulário de Extração de Dados.

Dado	Descrição
Origem da Solução	Organização ou Universidade dos autores do estudo
Ano de Publicação	Ano de publicação do estudo
Objetivo da Solução	Descrição da solução proposta
Ferramenta Citad	A ferramenta(s) citada(s) no estudo
Objetos de BD	Os objetos de BD adotados ou sugeridos pelas DSLs
Avaliação do Estudo	Se houve alguma avaliação, qual?
Ambiente de Avaliação	Academia ou indústria
Desafios	Quais são os desafios/trabalhos futuros apontados no estudo?
Pontos Positivos	Quais são os pontos positivos observados na implementação do estudo?
Pontos Negativos	Quais são os pontos positivos observados na execução do estudo?

Fonte: O autor.

4.1.7 Pesquisa na Literatura Cinza

Para a pesquisa por ferramentas que dessem apoio a modelagem ER na literatura cinza foi utilizado o motor de busca da Google. Para isto, combinações de *keywords* foram definidas a partir da compreensão da *string* de busca genérica e sua adequação para o contexto da busca na *Web*:

- *ERD Moldelling Tool*;
- *ERD Design Tool*;
- *Conceptual Design of Database*;
- *Conceptual Modelling of Database*;
- *Conceptual Modelling*;

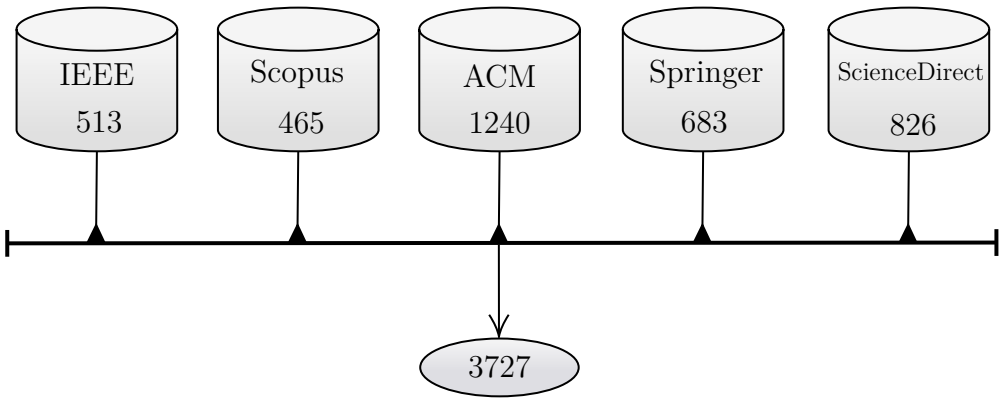
- *Database Modelling Tool*;
- *Database Design Tool*.

A verificação dos resultados foi limitado a 10 (dez) páginas no motor de busca para cada *keyword*. As análise das ferramentas no MLM estava sujeita a um processo de seleção com base em alguns requisitos. Primeiramente, a ferramenta deveria oferecer acesso para alguma forma de uso (incluindo versão de demonstração para ferramentas comerciais), necessitaria dar suporte a algum nível de modelagem de BD e precisaria ter *interface* em inglês ou português. Após, as ferramentas incluídas deveriam ter as notações e níveis de modelagem extraídos, juntamente com outras informações relevantes, e então categorizadas.

4.2 Execução do Mapeamento Multivocal

Para a execução do SLM foi necessário adaptar a sintaxe da *string* genérica para gerar outras versões, buscando assim adequá-la as peculiaridades de parametrização das diferentes bases utilizadas. Em seguida, foi realizada a busca dos estudos nas bases de dados. A Figura 14 mostra os resultados por biblioteca digital, bem como o total de estudos recuperados.

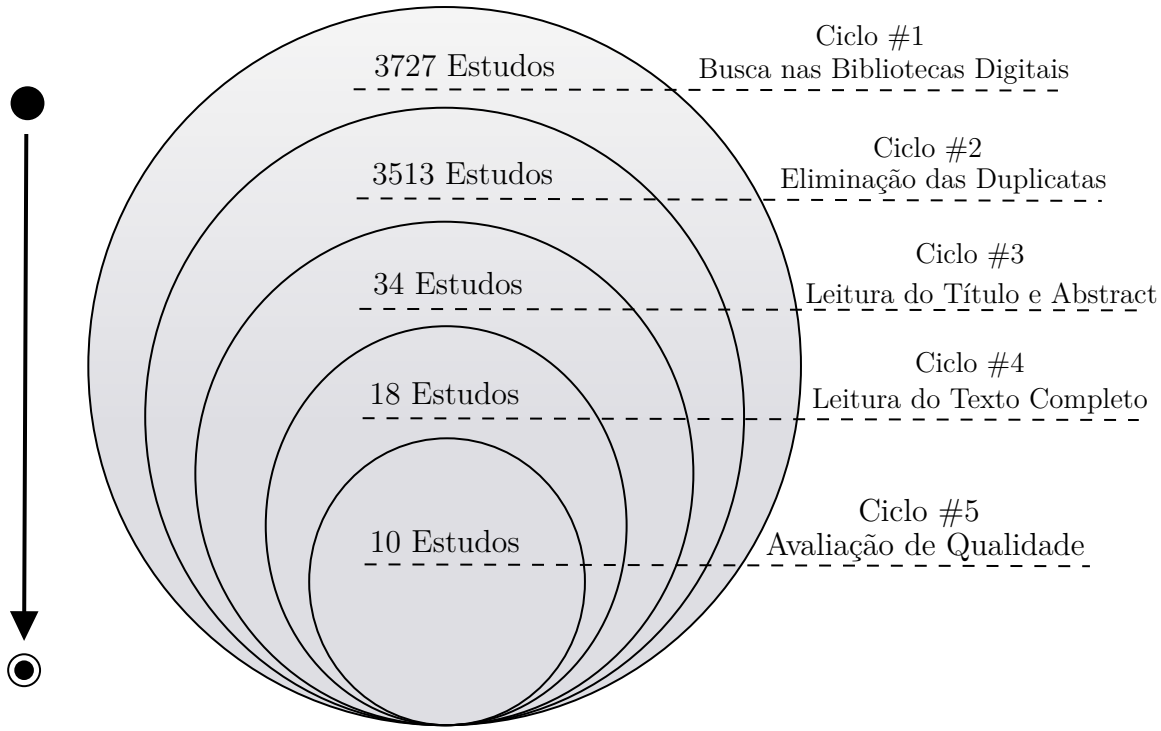
Figura 14 – Estudos primários por biblioteca digital.



Fonte: O autor.

Com o conjunto inicial de 3727 estudos primários identificados, foram definidos cinco (5) ciclos de seleção, apresentados na Figura 15. Nestas iterações houve a exclusão de estudos duplicados (restando 3513), a seleção de estudos baseado em título e *abstract* (restando 34), seleção baseada em texto completo (restando 18) e a seleção baseada na avaliação de qualidade (restando 10). Cada iteração tinha o objetivo de eliminar estudos que estavam fora do escopo da pesquisa ou considerados não relevantes. Na última

Figura 15 – Ciclos de seleção dos estudos primários.



Fonte: O autor.

iteração, 18 estudos tiveram sua qualidade analisada. Foi estabelecido que apenas estudos com pontuação acima de 5 (cinco) seriam aceitos. Assim, após a aplicação dos CQ foram excluídos 8 (oito) trabalhos. O conjunto final de dez (10) estudos aprovados na SLM seguiu para a etapa de extração de dados. Com os trabalhos acadêmicos a menor pontuação foi de 5.3, enquanto o maior alcançou 10. A Tabela 6 resume os resultados obtidos na avaliação da qualidade.

A execução do protocolo de busca na *grey literature* retornou um total de 132 ferramentas. Após sumarização foi realizado a exclusão de duplicatas (restando 67). Durante esse processo houve doze (12) ferramentas que não puderam ser avaliadas *e.g.* impossibilidade de instalação ou incompatibilidade com o ambiente utilizado. Ao fim, foi possível executar testes de uso com um total de 55 ferramentas, e consequentemente, a extração dos dados relevantes ao estudo. Durante o uso das ferramentas foram realizados modelos simples de BDs, em que se procurou observar o suporte aos níveis de modelagem e as notações e linguagens usadas.

4.3 Resultados e Discussão

Os estudos primários foram avaliados com base nos critérios de qualidade definidos na subseção 4.1.5. A menor pontuação foi de 5,3 enquanto o maior alcançou 10. A Tabela 6

resume os resultados obtidos na avaliação da qualidade.

Tabela 6 – Resultados da avaliação de qualidade.

Estudos	**Critérios de Qualidade**							**Pontuação**
Referência	**CQ1**	**CQ2**	**CQ3**	**CQ4**	**CQ5**	**CQ6**	**CQ7**	**Total**
Ayadi, Bouslimi e Akaichi (2016)	P	P	T	P	N	T	P	5,7
Celikovic et al. (2014)	T	T	N	T	P	P	T	7,0
Dimitrieski et al. (2015)	T	T	T	T	T	T	T	10,0
Hammer e Leod (1981)	T	T	N	T	N	P	P	5,5
Jagannathan et al. (1988)	T	T	N	T	P	N	N	5,5
Kersten et al. (2011)	P	T	T	P	N	N	P	5,3
Litwin et al. (1989)	P	T	N	T	N	P	T	5,4
Mazairac e Beetz (2013)	T	T	P	T	N	N	P	5,9
Shipman (1981)	T	T	N	T	P	N	P	6,0
Tian et al. (2006)	T	P	N	T	P	P	T	6,4

Fonte: O autor.

Em relação ao estado da arte (**QP1**) do desenvolvimento de DSL aplicado na transformação de modelos de dados, identificou-se um estudo que apresenta a *System Modeling Tool* (MIST), o qual utiliza uma DSL bidirecional para modelagem conceitual de BDs com a abordagem EER, chamado EERDSL.

Quanto às metodologias, técnicas ou propostas de refinamento (**QP1.1**) baseadas em modelos de dados, apenas os dois estudos mencionados aplicam conceitos para refinamento, sendo que tais conceitos são apoiados na normalização de BDs para auxiliar os desenvolvedores que utilizam sua solução.

No que se refere às tecnologias usadas como suporte para o desenvolvimento de DSLs (**QP1.2**), foram registradas Xtext, Xtend, Sirius e Eugenia (CELIKOVIC et al., 2014; DIMITRIESKI et al., 2015), StarUML (AYADI; BOUSLIMI; AKAICHI, 2016), IfcDoc Tool e ViewEdit Tool (MAZAIRAC; BEETZ, 2013), MonetDB (KERSTEN et al., 2011), Java, JFlex e JCup (TIAN et al., 2006). No entanto, estudos mais antigos eram geralmente especificações de DSLs, não apresentando qualquer forma de implementação ou ferramenta usada (SHIPMAN, 1981; JAGANNATHAN et al., 1988; LITWIN et al., 1989).

As representações de BD adotadas (**QP1.3**) possuem `Tables` e `Functions` em todos os estudos primários analisados. Há também referências explícitas à definição de `Stored Procedures`, `Triggers` e `Views` em outros estudos. A Tabela 7 resume esses dados recuperados de cada um dos estudos primários. Quanto ao uso de DSL para transformação de modelos de dados em casos reais (**QP2**, **RQ2.1**), nenhuma referência foi encontrada nos estudos avaliados.

Sobre os métodos usados para avaliar as DSLs (**QP2.2**) existe apenas um estudo preliminar que apresenta a validação da proposta (DIMITRIESKI et al., 2015) usando 16 participantes, sendo 2 especialistas em Interação Humano-Computador (IHC), 3 especialistas em modelagem de sistemas e 11 estudantes (6 mestrandos na área de BDs e

Tabela 7 – Objetos de banco de dados representados.

Estudo Primário		Objetos de BD				
Referência	**DSL**	**Tables**	**SP**	**Functions**	**Triggers**	**Views**
Ayadi, Bouslimi e Akaichi (2016)	Ayadi's Notation	✓				
Celikovic et al. (2014)	EERDSL v.1	✓		✓	✓	
Dimitrieski et al. (2015)	ERRDSL v.2	✓	✓	✓	✓	✓
Hammer e Leod (1981)	SDM	✓			✓	
Jagannathan et al. (1988)	SDM	✓		✓		
Kersten et al. (2011)	SciSQL	✓		✓		
Litwin et al. (1989)	MSQL	✓	✓	✓	✓	✓
Mazairac e Beetz (2013)	BIMQL	✓	✓	✓		
Shipman (1981)	DAPLEX	✓		✓		
Tian et al. (2006)	NeuroQL	✓		✓		

Legenda: SP = *Stored Procedures*.

Fonte: O autor.

5 doutorandos com experiência em modelagem). Em geral, os outros estudos indicam a falta de uma avaliação de suas proposições como um possível trabalho futuro.

Entre os aspectos positivos e negativos observados (**RQ2.3**) nos estudos, é positivo ser fácil de entender e intuitivo modelagem e independência de plataforma (TIAN et al., 2006; MAZAIRAC; BEETZ, 2013). Pontos negativos foram a falta de geração automática de SQL para sistemas de BD (AYADI; BOUSLIMI; AKAICHI, 2016) ou uma limitação neste item (DIMITRIESKI et al., 2015). Ainda há um registro da falta de implementação real de DSLs até o momento em que o estudo foi realizado, havendo apenas especificações (HAMMER; LEOD, 1981; JAGANNATHAN et al., 1988; TIAN et al., 2006; KERSTEN et al., 2011; AYADI; BOUSLIMI; AKAICHI, 2016). Os principais desafios identificados pelos estudos (**RQ2.4**), em geral, são as avaliações das abordagens, bem como a evolução e/ou simplificação das propostas. Finalmente, na Tabela 8 as DSLs são apresentadas em relação ao seu tipo. No entanto, é importante observar que os estudos que marcam a coluna `bidirecional` (CELIKOVIC et al., 2014; DIMITRIESKI et al., 2015) são versões diferentes da mesma implementação de DSL, enquanto o (HAMMER; LEOD, 1981; JAGANNATHAN et al., 1988) são uma especificação de DSL e implementação com base nesta especificação, respectivamente.

Quanto ao estado da prática (**QP3**) das ferramentas utilizadas na modelagem de BD, foram mapeadas 55 ferramentas. Houve a classificação quanto ao seu tipo, sendo 29 exclusivas de modelagem BD (*Data Modeling*), 13 de modelagem que ainda oferecem conexão com BD e execução de consultas (*Full IDE*), 10 com suporte a diagramação de diversos tipos de modelos (*Diagramming*) e 3 ferramentas projetadas para grandes empresas, podendo diagramar inúmeros tipos de documentos e processos (*Enterprise Modeling*).

Em relação às notações utilizadas nas ferramentas (**QP3.1**) foram identificados mais de 10 (dez) variedades de notações, com destaque a notação *Crow's Foot* com 35 ocorrências e da notação IDEF1X com 23 registros. Esses e outros dados estão listados nas Tabelas 9 e 10.

Finalmente, no que diz respeito aos modelos suportados pelas ferramentas (**QP3.2**)

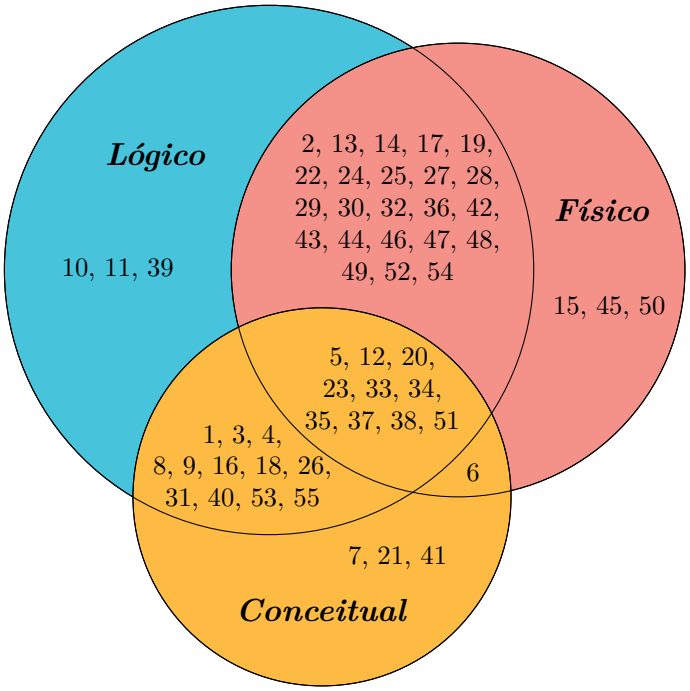
Tabela 8 – Categorização das DSLs propostas.

Estudos Primários		Tipo de DSL		
Referência	**DSL**	**Textual**	**Gráfica**	**Bidirecional**
Ayadi, Bouslimi e Akaichi (2016)	Ayadi's Notation		✓	
Celikovic et al. (2014)	EERDSL v.1			✓
Dimitrieski et al. (2015)	EERDSL v.2			✓
Hammer e Leod (1981)	SDM	✓		
Jagannathan et al. (1988)	SDM	✓		
Kersten et al. (2011)	SciQL	✓		
Litwin et al. (1989)	MSQL	✓		
Mazairac e Beetz (2013)	BIMQL	✓		
Shipman (1981)	DAPLEX	✓		
Tian et al. (2006)	NeuroQL	✓		

Fonte: O autor.

foi constatado que individualmente 26 ferramentas oferecem suporte a modelagem conceitual, 48 ferramentas à modelagem lógica e 37 à modelagem física. O conjunto representando as intersecções do suporte aos modelos quanto às ferramentas é exibido na Figura 16.

Figura 16 – Diagrama de Venn dos modelos suportados nas ferramentas.



Fonte: O autor.

Tabela 9 – Ferramentas para modelagem de bancos de dados.

		Tipo				Modelos			Notações Suportadas							Ambiente		Licença	
#	**Ferramenta**	**DM**	**FIDE**	**DG**	**EM**	**C**	**L**	**F**	**CF**	**IDEF1x**	**CN**	**MN**	**UML**	**BN**	**AN**	**ON**	**D**	**W**	**C**	**G**
1	AnalyseSI	✓				✓	✓			✓		✓					✓			✓
2	Aqua Data Studio ER Modeler		✓				✓	✓	✓	✓							✓		✓	
3	Astah			✓		✓	✓		✓	✓							✓		✓	
4	brModelo			✓		✓	✓			✓	✓					✓			✓	
5	Creately			✓		✓	✓		✓									✓	✓	
6	Database Deployment Manager		✓			✓	✓	✓			✓						✓			✓
7	Database Workbench		✓			✓		✓	✓								✓		✓	
8	DB Designer	✓				✓										✓	✓		✓	
9	DB-Main	✓				✓	✓						✓				✓			✓
10	DBDesigner 4	✓				✓	✓		✓	✓		✓					✓			✓
11	DBDesigner.net	✓					✓									✓		✓	✓	
12	dbdiagram.io	✓					✓									✓		✓		✓
13	dbDiffo	✓				✓	✓	✓	✓						✓			✓		✓
14	dbForge Studio for MySQL		✓				✓	✓	✓	✓							✓		✓	
15	DBSchema		✓				✓	✓	✓	✓				✓			✓		✓	
16	DBVisualizer		✓					✓								✓	✓		✓	
17	DbWrench	✓				✓	✓		✓								✓		✓	
18	DeZign for Databases	✓					✓	✓	✓	✓							✓		✓	
19	Dia			✓		✓	✓		✓	✓	✓		✓	✓	✓		✓			✓
20	dModelAid	✓					✓	✓										✓	✓	
21	Enterprise Architect				✓	✓	✓	✓	✓	✓							✓		✓	
22	ER-Assistant	✓				✓			✓								✓			✓
23	ER/Builder	✓					✓	✓								✓	✓			✓
24	ER/Studio Data Architect	✓				✓	✓	✓	✓	✓							✓		✓	
25	ERD Concepts		✓				✓	✓	✓	✓							✓		✓	
26	ERDesigner NG	✓					✓	✓	✓								✓			✓
27	ERDPlus	✓				✓	✓		✓									✓		✓
28	Erwin Data Modeler	✓					✓	✓	✓	✓							✓		✓	

Legenda: DM (*Data Modeling*) FIDE (*Full IDE*) DG (*Diagramming*) EM (*Enterprise Modeling*) | C (Conceitual) L (Lógico) F (Físico) | CF (*Crow's Foot*) CN (*Chen's Notation*) MN (*Merise Notation*) BN (*Barker's Notation*) AN (*Arrow Notation*) ON (*Other Notation*) | D (*Desktop*) W (Web) C (Comercial) G (*Gratuíta*)

Fonte: O autor.

Tabela 10 – Ferramentas para modelagem de bancos de dados (continuação).

		Tipo				Modelos			Notações Suportadas								Ambiente		Licença	
#	**Ferramenta**	**DM**	**FIDE**	**DG**	**EM**	**C**	**L**	**F**	**CF**	**IDEF1x**	**CN**	**MN**	**UML**	**BN**	**AN**	**ON**	**D**	**W**	**C**	**G**
29	GenMyModel RDS			✓			✓	✓	✓									✓	✓	
30	InfoSphere Data Architect				✓		✓	✓	✓	✓							✓		✓	
31	Jeddict			✓			✓	✓	✓								✓			✓
32	ModelRight	✓				✓	✓			✓							✓		✓	
33	MySQL Workbench		✓				✓	✓	✓	✓			✓			✓	✓			✓
34	Navicat Data Modeler	✓				✓	✓	✓	✓	✓							✓		✓	
35	Navicat Data Modeler	✓				✓	✓	✓	✓	✓			✓				✓		✓	
36	Open ModelSphere	✓				✓	✓	✓	✓		✓	✓					✓			✓
37	Oracle SQL Developer Data Modeler		✓				✓	✓	✓	✓							✓			✓
38	pgModeler	✓				✓	✓	✓								✓	✓			✓
39	PowerDesigner				✓	✓	✓	✓	✓	✓	✓	✓		✓			✓		✓	
40	QuickDBD	✓					✓									✓		✓	✓	
41	RISE			✓		✓	✓		✓								✓			✓
42	Software Ideas Modeler			✓		✓			✓	✓	✓						✓		✓	
43	SQL Database Modeler	✓					✓	✓		✓								✓	✓	
44	SQL Maestro		✓				✓	✓		✓							✓		✓	
45	SQL Power Architect	✓					✓	✓	✓								✓		✓	
46	SQL Server Management Studio		✓					✓								✓	✓			✓
47	SQLDBM	✓					✓	✓		✓								✓		✓
48	SQLyog		✓				✓	✓								✓	✓		✓	
49	Toad Data Modeler	✓					✓	✓	✓	✓							✓		✓	
50	Valentina Studio		✓				✓	✓	✓								✓		✓	
51	Vertabelo	✓						✓	✓									✓	✓	
52	Visual Paradigm			✓		✓	✓	✓	✓								✓		✓	
53	Win A&D			✓			✓	✓	✓								✓		✓	
54	WWW SQL Designer	✓				✓	✓									✓		✓		✓
55	xCase	✓					✓	✓	✓								✓		✓	

Legenda: DM (*Data Modeling*) FIDE (*Full IDE*) DG (*Diagramming*) EM (*Enterprise Modeling*) | C (Conceitual) L (Lógico) F (Físico) | CF (*Crow's Foot*) CN (*Chen's Notation*) MN (*Merise Notation*) BN (*Barker's Notation*) AN (*Arrow Notation*) ON (*Other Notation*) | D (*Desktop*) W (Web) C (Comercial) G (*Gratuíta*)

Fonte: O autor.

4.4 Ameaças à Validade

Ameaças ao resultado do estudo foram identificadas no MLM realizado, e então categorizadas nos seguintes tipos: validade de construto, validade interna, validade externa e validade de conclusão (COOK; CAMPBELL, 1979; WOHLIN et al., 2012).

Validade do Construto: Aborda a possibilidade de que as QPs ou os termos de pesquisa que estruturam a *string* de busca sejam inadequados ou incompletos. Para mitigar essas ameaças, pesquisadores da área de DSL e modelagem de dados foram consultados. Além disso, foi realizada uma pesquisa piloto para avaliar a consistência de nossa *string* de pesquisa. Outra ameaça é a qualidade do material publicado que foi coletado na literatura cinza.

Validade Interna: Algumas possíveis ameaças são o uso de métodos incorretos de busca, o que pode levar a exclusão de estudos relevantes, uma aplicação de estratégia de extração de dados precária, a ocorrência de vieses na seleção ou no conteúdo dos estudos primários. Na tentativa de mitigar esses riscos, um protocolo foi definido com base em modelos de referência já bem estabelecidos na literatura.

Validade Externa: Ameaças externas geralmente abordam se as descobertas de um estudo podem ser generalizadas para outro domínio. Uma razão para esta ameaça seria a ocorrência da seleção estudos de primários contendo informações incompletas. Contudo é provável que por se tratar de uma área de intersecção entre modelagem de dados e DSLs, os resultados não podem ser generalizados para outros tópicos de pesquisa, reduzindo assim esse risco naturalmente.

Validade da Conclusão: Uma possível ameaça é o viés na extração de dados, o que leva a erros de conclusão. Para atenuar esse problema foi realizado uma leitura criteriosa e, assim como os testes de uso das ferramentas, houve a síntese de dados em planilha eletrônica[1] para uma melhor análise.

4.5 Lições do Capítulo

Todos os anos, várias contribuições para a modelagem ER são publicadas. A modelagem de BD é uma área essencial na ES e as DSLs que suportam essa atividade não são encontradas trivialmente na literatura. A fim de acompanhar a evolução e as tendências de vários sistemas de BDs, uma pesquisa de alternativas para o *design* é essencial. Neste capítulo é fornecido uma visão geral sobre as DSLs usadas pela modelagem de ER por meio de um MLM.

Este mapeamento abrangeu 3727 artigos com a intenção de investigar estudos primários que fizeram propostas de DSLs para modelagem de BD. Da mesma forma, foi levantado um conjunto de 132 ferramentas que dão suporte a modelagem de BDs, procurando mapear as notações e modelos suportados pelas mesmas. O protocolo do MLM

[1] http://bit.ly/2Vs0oYN

foi detalhado, assim como sua condução e subsequente análise dos resultados obtidos. Ao final, 10 estudos primários e 55 ferramentas foram selecionadas para serem analisados de forma quantitativa e qualitativa. Como resultado, classificou-se apenas as DSLs atualmente usadas para dar suporte à modelagem de ER e ferramentas que são utilizadas para projetar BDs. Entre os resultados destaca-se que o estudo de Dimitrieski et al. (2015), o qual apresenta uma ferramenta de modelagem bidirecional que aplica sua própria DSL com base na abordagem EER.

Este capítulo forneceu algumas evidências de que, a cada ano, um número significativo de trabalhos apresentando diferentes tipos de notações é publicado. Isso é de certa forma surpreendente, devido ao fato de que as notações de relacionamento entre entidade usadas hoje pela indústria e pela academia, como as de Chen (1976) e Barker (1990), não são propostas recentes. Portanto, conclui-se que a modelagem de ER continua um amplo campo de pesquisa com algumas lacunas a serem exploradas.

5 PROPOSTA DE DSL

Este capítulo apresenta a proposta central deste trabalho. A Seção 5.1 aponta os requisitos levantados para a construção da DSL. A Seção 5.2 descreve as decisões de projeto referente aos requisitos. A arquitetura da implementação da proposta é detalhada na Seção 5.3. A demonstração do protótipo construído ocorre na Seção 5.4 e, por fim, as lições do capítulo são pontuadas na Seção 5.5.

5.1 Requisitos da Linguagem

Esta seção lista os requisitos que foram definidos com base na literatura utilizada neste trabalho, bem como no conhecimento prévio dos pesquisadores envolvidos na condução do estudo. Estes requisitos são relacionados diretamente com as decisões de projeto.

- ***RQ1. A DSL precisa ser disponibilizada sob uma licença open source.*** Como o foco da proposta é no processo de ensino é fundamental que a linguagem seja de código aberto. A vantagem que este requisito proporciona é a posterior evolução e manutenção colaborativa com o envolvimento de outros desenvolvedores.

- ***RQ2. A DSL deve permitir representar textualmente modelos conceituais de BDs.*** Como é um objetivo que a solução seja outra opção em relação as abordagens gráficas, esse requisito se justifica. Isso permite o foco na compreensão do domínio e no desenvolvimento da DSL.

- ***RQ3. Os modelos conceituais devem dar suporte a definição de entidades, atributos, relações e cardinalidades.*** As ferramentas utilizadas para o desenvolvimento da linguagem precisam permitir que sejam implementados os conceitos de domínio que regem a estrutura de DER tradicional.

- ***RQ4. Os modelos conceituais devem dar suporte a definição de atributos identificadores, generalização/especialização, auto-relacionamentos e relacionamentos ternários.*** A linguagem deve permitir que conceitos mais sofisticados dos domínios sejam definidos.

- ***RQ5. A implementação da DSL deve realizar a transformação do modelo conceitual para o lógico.*** A solução deve realizar a transformação do conceitual para o lógico, exibindo o resultado gerado ao usuário.

- ***RQ6. A implementação da DSL deve gerar modelos físicos equivalentes, com base no modelo conceitual ou lógico.*** A solução precisa realizar a geração de instruções SQL para diferentes SGBDs.

5.2 Decisões de Projeto

Nesta seção são descritas as decisões de projeto para criar a DSL textual que suporte todos os requisitos discutidos na Seção 5.1. Para cada decisão de projeto são indicados os seus requisitos associados.

- ***DP1. A solução deve adotar um LW open source no auxílio da implementação da DSL textual (RQ1, RQ2).*** Mediante a investigação conduzida durante este estudo foi selecionado o LW Xtext para o desenvolvimento da proposta por ser um *framework open source* focado no desenvolvimento de DSLs textuais, fornecendo toda a infraestrutura necessária. Além disto, o Xtext é uma ferramenta com alto nível de maturidade, documentação detalhada e uma comunidade ativa.

- ***DP2. A DSL deve fornecer uma representação textual que seja equivalente ao modelo ER gráfico usualmente utilizado (RQ3, RQ4).*** Para os requisitos cobertos por esta decisão de projeto foi adotada a estratégia de se realizar uma análise nas ferramentas averiguadas no mapeamento descrito no Capítulo 4, bem como no livro referência de Heuser (2009).

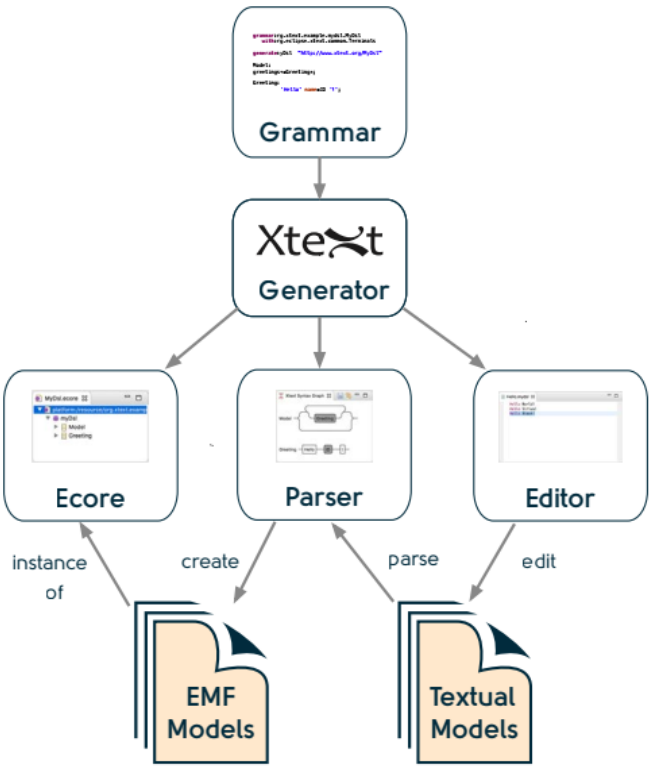
- ***DP3. A solução deve realizar a transformação entre os modelos (RQ5).*** O Xtext usa modelos do *Eclipse Modeling Framework* (EMF) como a representação na memória de qualquer arquivo de texto analisado. Esse grafo de objetos na memória é chamado de árvore sintática abstrata, do inglês *Abstract Syntax Tree* (AST). Esses conceitos também são chamados de gráficos de objeto de documento, do inglês *Document Object Graph* (DOM), modelo semântico ou simplesmente modelo. Desta forma, existe a representação do modelo da gramática na forma de um metamodelo central no núcleo do EMF, chamado de modelo *Ecore*. Tendo o *Ecore* da DSL proposta como uma representação, é possível então aplicar regras de transformação, gerando assim outros modelos.

- ***DP4. A solução deve prover a integração entre a DSL e outras tecnologias (RQ6).*** A solução deve permitir a realização da exportação dos modelos construídos para um formato de instruções SQL, representando assim o modelo físico. Inicialmente, essa integração será realizada para o SQL Server, o MySQL e o PostgreSQL. Foram estabelecidas essas tecnologias pois são alguns dos SGBDs mais utilizados no mercado, conforme foi descrito nos tópicos da subseção 3.2.1.

5.3 Arquitetura

O *framework* Xtext gera toda a infraestrutura para a criação de linguagens com base fundamentalmente nas gramáticas definidas. A Figura 17 fornece uma visão geral em um nível abstrato da arquitetura do Xtext.

Figura 17 – Arquitetura geral do Xtext.



Fonte: Obeo e TypeFox (2017).

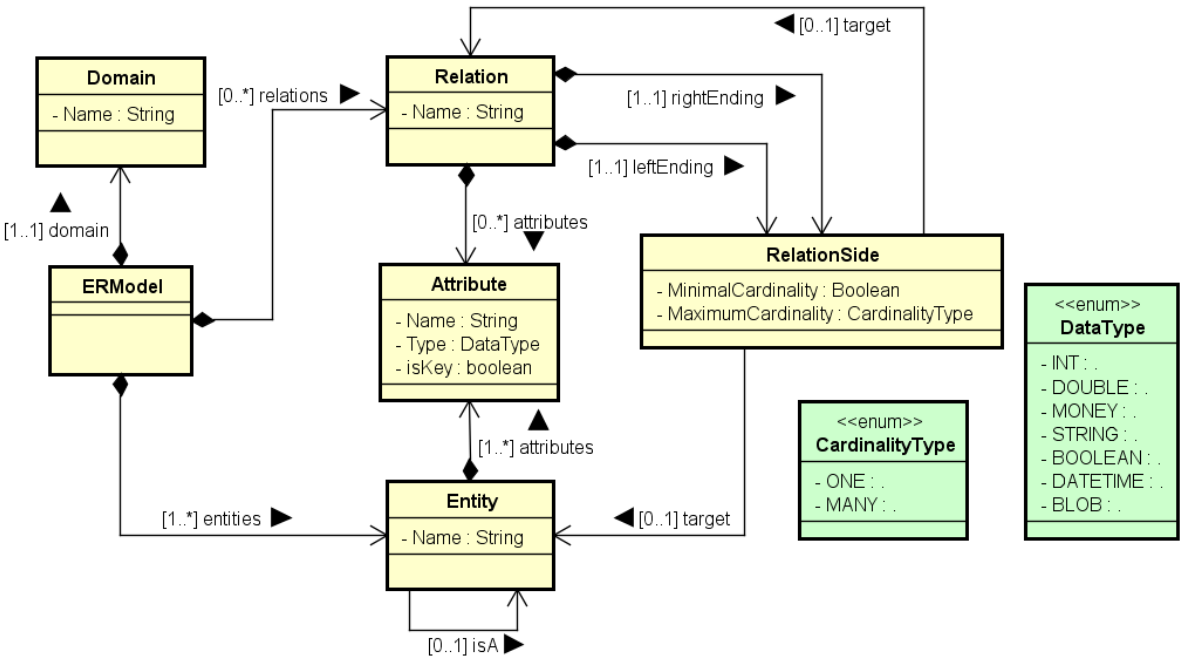
Foram implementadas duas versões da gramática no protótipo da DSL. A seguir as arquiteturas dessas implementações são descritas e pontua-se as diferenças entre elas. Essa abordagem foi definida tendo em vista que será realizada uma avaliação preliminar junto a um grupo focal. A partir dos resultados dessa avaliação, objetiva-se gerar uma versão final da gramática e, consequentemente, da arquitetura.

A Figura 18 mostra um diagrama de classes para representar o modelo *Ecore* da primeira versão criada para esta proposta. O elemento central do modelo é a classe `ERModel`, a qual corresponde por uma composição com outros elementos. O `ERModel` deve possuir um `Domain` associado, simbolizando o nome da base de dados modelada.

Um `ERModel` também deve conter um ou mais elementos `Entity`. Um elemento `Entity` pode se relacionar com outro elemento `Entity`, cobrindo assim o conceito de generalização/especialização. Um `Entity` é também uma composição de uma ou mais classes `Attribute`. Definiu-se os tipos de dados em uma lista enumerada no elemento `DataType`.

Em seguida, um `ERModel` pode ser composto de uma ou mais relações, retratado como a classe `Relation`. Uma `Relation` por sua vez é formada por duas classes `RelationSide`, os quais são as cardinalidades da esquerda e da direita em uma relação. Estas duas relações possuem uma referência chamada `Target`, que pode ser uma `Entity` ou uma `Relation`. A inclusão da possibilidade de se referenciar uma `Relation` se faz

Figura 18 – Representação do modelo *Ecore* da 1º versão da DSL.



Fonte: O autor.

necessário para cobrir a modelagem dos relacionamentos ternários.

Finalmente, tem-se o conceito das cardinalidades. Nele as cardinalidades possíveis, mantidas em atributos da `RelationSide`, são os tipos enumerados em `CardinalityType`, sendo `One` para um e `Many` para muitos. Por meio das regras da gramática garante-se que a cardinalidade mínima é implícita, tendo a palavra reservada *zero* para simbolizar quando uma cardinalidade mínima pode ser nula. Isto significa que, em uma modelagem onde uma cardinalidade mínima *zero* é omitida, assume-se que ela automaticamente é igual a um.

Estruturalmente o modelo *Ecore* da segunda versão DSL não muda significativamente.A única diferença com maior impacto é a opção pela definição da cardinalidade utilizando-se as quatro combinações possíveis como termos reservados, armazenados diretamente no atributo `Cardinality` de `RelationSide`.

5.4 Protótipo

Na fase atual do trabalho a linguagem a nível conceitual não se encontra totalmente finalizada. Existem tópicos relativos a validação de escopo, como no caso do tratamento de referências cruzadas indesejadas, e outras restrições inerentes ao modelo ER que devem ser analisadas e então implementadas. A definição da DSL criada é exibida na Figura 19.

O comando `grammar` especifica o nome da DSL, enquanto que a instrução `with` declara uma herança de outra linguagem. No caso da gramática proposta, será utilizado

Figura 19 – Implementação da 1º versão da DSL.

```
grammar org.xtext.unipampa.lesse.erdsl with org.eclipse.xtext.common.Terminals

generate erdsl "http://www.xtext.org/unipampa/lesse/erdsl/"

ERModel:
        domain=Domain ";"
        ("Entities{") entities+=Entity+ ("};")
        ("Relationships{") relations+=Relation* ("};");

Domain:
        "Domain" name=ID;

Entity:
        name=ID ("isA" isA+=[Entity])*
        ("{" attributes+=Attribute ("," attributes+=Attribute)* "}")?;

Attribute:
  name=ID ":" type=DataType (isKey?="isIdentifier")?;

Relation:
        (name=ID)?
        ("[" leftEnding=RelationSide
        "isRelatedWith"
        rightEnding=RelationSide "]")
        ("{" attributes+=Attribute
        ("," attributes+=Attribute)* "}")*;

RelationSide:
        ((minimalCardinality?="zero")?) maximumCardinality=CardinalityType
        target=[Entity] | target=[Relation];

enum DataType:
        INT="int" | DOUBLE="double" | MONEY="money" | STRING="string" |
        BOOLEAN="boolean" | DATETIME="datetime" | BLOB="file";

enum CardinalityType:
        One="one" | Many="many";
```

Fonte: O autor.

uma gramática padrão do Xtext, chamada `Terminals`, a qual fornece algumas regras predefinidas como, por exemplo, a regra `ID` para identificadores ou `INT` para inteiros. O comando `generate` é a instrução que produz a AST da linguagem.

A primeira regra, chamada de regra de entrada, define como é a estrutura geral da linguagem. Palavras e símbolos entre aspas duplas ou simples indicam as palavras reservadas. Por exemplo, o objeto `Entities` é obrigatoriamente precedido de `"Entities{"`. Este objeto representa uma espécie de *container*, sendo isto indicado por meio do operador de atribuição `+=`. Ele é um objeto que pode conter outros objetos, no caso um ou mais `Entity` (entidades). É estabelecido que cada arquivo da DSL deve também ser composto de um `Domain` (domínio) e zero ou mais `Relation` (relações).

Para melhor entendimento, deve-se deixar claro que a multiplicidade é indicada por `*` (zero ou muitos), `+` (um ou muitos) ou `?` (zero ou um). Ao não se colocar nenhum desses operadores, implicitamente se espera então apenas uma ocorrência. Em relação as atribuições, quando apenas um `=` for especificado significa que o objeto da esquerda

espera apenas um registro. Logo, para `+=` espera-se então zero, uma ou mais ocorrências.

O objeto `Domain` é precedido de uma palavra reservada com o mesmo nome, seguido de um identificador. A entidade é definida pela palavra `Entity` e um nome identificador específico para este objeto. A definição de uma herança é opcional por meio da palavra reservada `isA`. Após a definição do nome, abre-se um corpo de chaves em que são especificados os atributos da entidade. Uma entidade deve conter ao menos um atributo, mas ele não precisa ser identificador por conta da possível existência de entidades fracas.

As regras compostas só são realizadas devido a possibilidade de se agrupar expressões com o uso de parênteses, além da possibilidade de se utilizar outras regras por meio de referências cruzadas. Os colchetes entre a regra `Entity` servem para indicar que se almeja usar apenas o atributo `name` que identifica o objeto. Os atributos das entidades são definidos por um nome, herdando a regra `ID` de `Terminals`, e atributos `isIdentifier` opcionais para simbolizar chaves primárias.

A relação é definida, já dentro do corpo do bloco `Relationships`, com uma declaração opcional de sua identificação. Em seguida, são abertos colchetes e deve-se especificar dois elementos `RelationSide` como referência aos atributos `leftEnding` e `rightEnding`. Estes atributos representam os lados de uma relação. Estes objetos devem ser separados pela expressão `isRelatedWith`. Os lados da relação são definidos na regra *RelationSide*, composta de dois atributos. O atributo `minimalCardinality` é opcional, indicado pelo operador `?`, e aceita apenas a palavra reservada `zero`. O atributo `maximumCardinality` aceita um objeto `CardinalityType`.

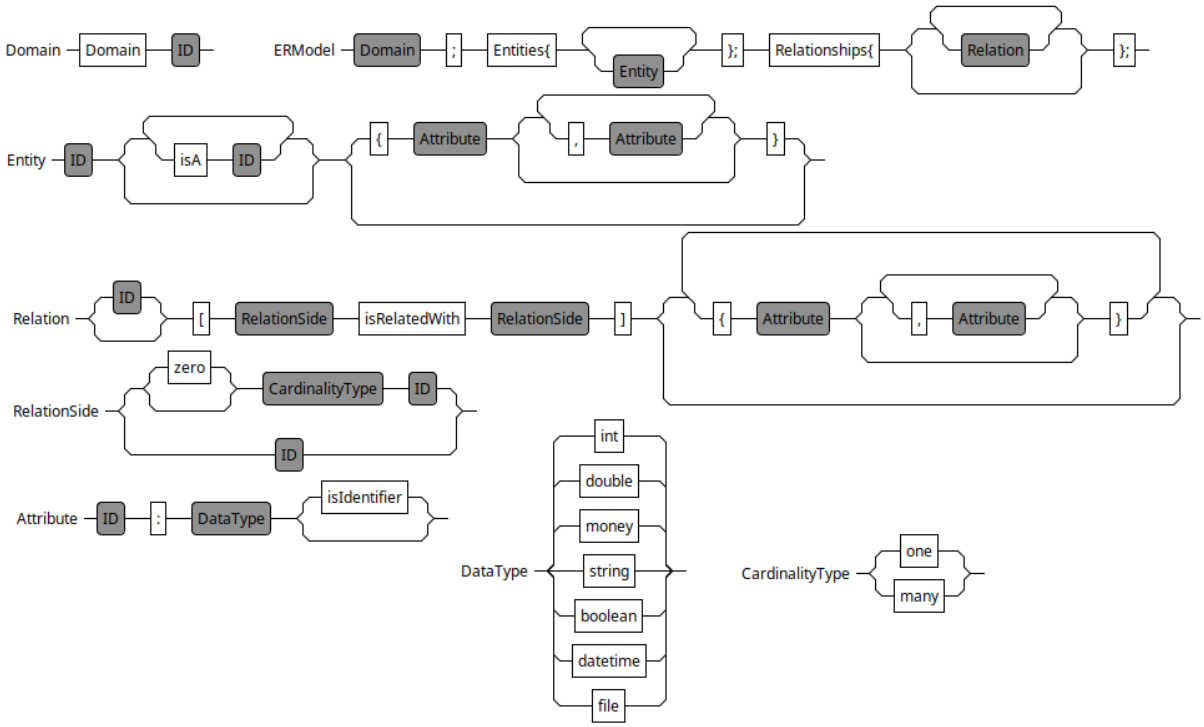
Os tipos de atributo estão contidos em uma lista enumerada chamada `DataType`, na esquerda fica a representação no modelo *Ecore*. Na direita está a palavra reservada que é usada na linguagem. O símbolo condicional `|` significa o operador lógico *OR* (ou) e serve para separar cada definição `<chave> = <valor>` como uma opção dentro da lista. Das três cardinalidades possíveis, duas estão definidas em outra lista enumerada chamada `CardinalityType`. Na Figura 20 exibe a principal mudança entre as versões da DSL proposta, a qual diz respeito a cardinalidade explícita. A Figura 21 e a Figura 22 apresentam uma representação gráfica das sintaxes de ambas as gramáticas.

Figura 20 – Fragmento de implementação da 2º versão da DSL.

```
Relation:
        (name=ID)? ("[" leftEnding=RelationSide "relates"
        rightEnding=RelationSide "]") ("{"attributes+=Attribute
        ("," attributes+=Attribute)* "}")*;

RelationSide:
        Cardinality=("(0,1)" | "(1,1)" | "(0,N)" | "(1,N)")
        target=[Entity] | target=[Relation];
```

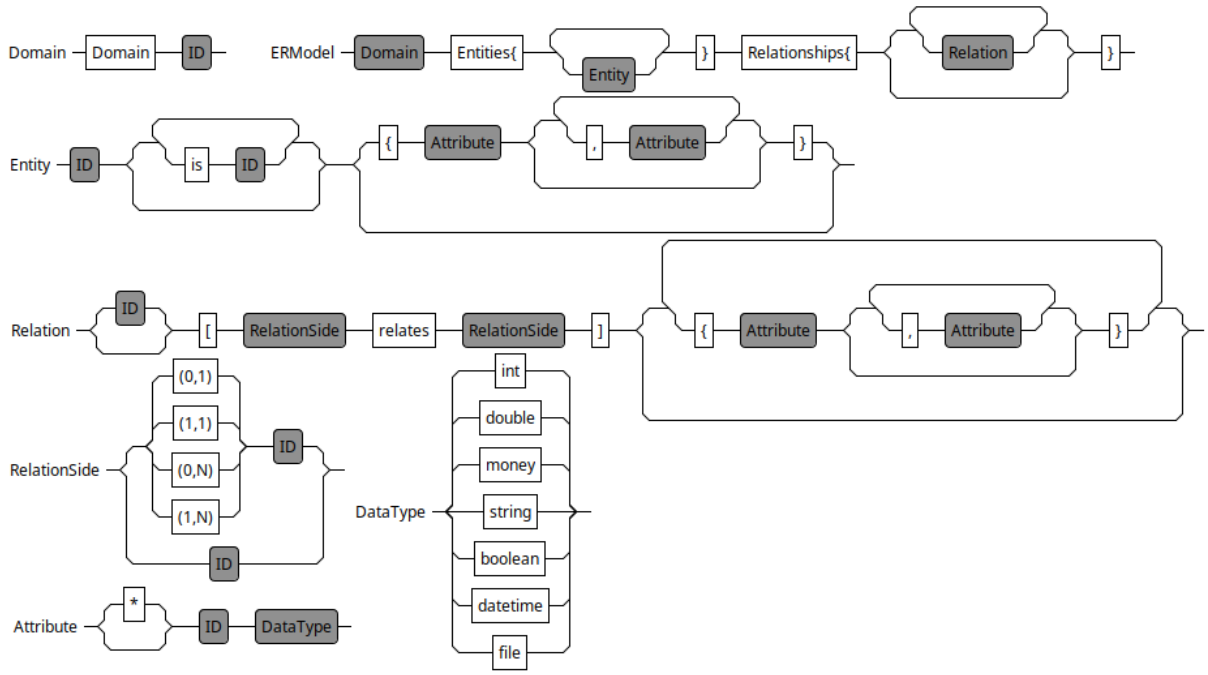
Fonte: O autor

Figura 21 – Representação da sintaxe da 1º versão da DSL.



Fonte: O autor.

Figura 22 – Representação da sintaxe da 2º versão da DSL.



Fonte: O autor.

Na Figura 23 mostra-se um exemplo de uso com um pequeno modelo que aborda uma universidade como domínio. Neste protótipo já se pode observar a modelagem de

generalização/especialização, auto-relacionamento e relacionamento ternário.

Figura 23 – Exemplo de uso da 1º versão da DSL no RCP do Eclipse.

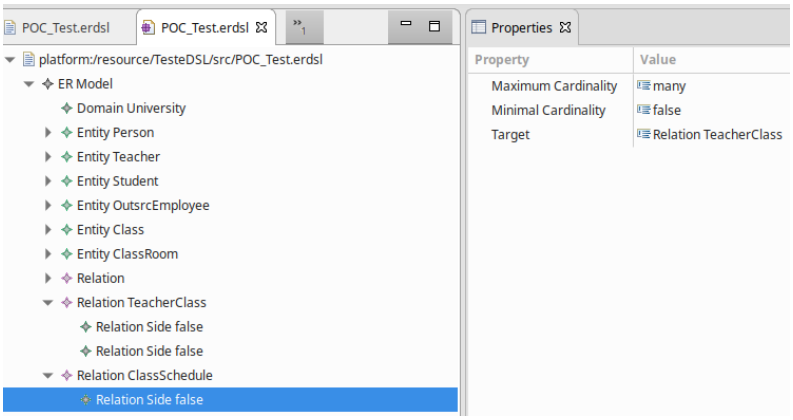
```
Domain University;

Entities{
        Person{
                PID: int isIdentifier,
                Name: string
        }
        Teacher isA Person{
                Phone: int,
                Salary:money
        }
        Student isA Person{
                Course: string
        }
        OutsrcEmployee isA Person{
            OutsourcedEID: int isIdentifier,
            Company: string
        }
        Class{
                ClassID: int isIdentifier,
                Course: string,
                Semester: string
        }
        ClassRoom {
                ClassRoomID: int isIdentifier,
                Capacity: int
        }
};

Relationships{
        [many Student isRelatedWith many Class]
        TeacherClass [many Teacher isRelatedWith many Class]
        ClassSchedule [many TeacherClass isRelatedWith many ClassRoom]
                {ClassScheduleID: int, DayOfWeek: datetime, Discipline: string}
        Supervisor [one OutsrcEmployee isRelatedWith many OutsrcEmployee]
};
```

Fonte: O autor.

É importante destacar que neste exemplo são modeladas seis entidades e quatro relacionamentos. Contudo, no processo de transformação para o modelo lógico espera-se que seja gerado um esquema textual com mais três entidades, inferindo-se isso através de, por exemplo, relacionamentos *muitos para muitos*. Isso ocorreria, nesta amostra, no relacionamento sem identificação, atribuindo automaticamente a nova entidade o nome resultante da concatenação das duas entidades que se relacionam no conceitual. Também seriam criadas novas entidades a partir da derivação dos relacionamentos nomeados `TeacherClass` e `ClassShedule`, sendo que o último carateriza um relacionamento ternário. Por fim, o auto-relacionamento `Supervisor` de `OutsrcEmployee` implicaria na adição de um novo atributo na entidade no novo modelo.

Com base nos modelos *Ecore* gerados em tempo real, como o gerado a partir do modelo descrito anteriormente e exibido na Figura 24, já está sendo implementada a transformação do modelo conceitual para o lógico. A implementação desta transformação está sendo realizada utilizando-se a Xtend, uma GPL baseada em Java.

Figura 24 – Fragmento do modelo *Ecore* gerado em tempo de execução.



Fonte: O autor.

5.5 Lições do Capítulo

Neste capítulo foram expostos os requisitos, as decisões de projeto e a arquitetura da DSL proposta. Também foi demonstrado como o protótipo da versão preliminar da linguagem foi definido, bem como um exemplo de uso.

Até o momento o Xtext mostrou-se uma LW capaz de suprir as necessidades iniciais do projeto, fornecendo suporte completo para a criação de gramáticas com notação BNF, uma meta-sintaxe amplamente usada para expressar gramáticas livres de contexto como nas estruturas de linguagens de programação no geral. Além de prover a validação da gramática criada, foi gerado um *plugin* tornando assim possível a realização do teste do protótipo em um RCP Eclipse.

6 CONSIDERAÇÕES PRELIMINARES

Este trabalho abordou o projeto de conclusão de curso expondo seu planejamento metodológico, a fundamentação teórica, um mapeamento multivocal de literatura e a proposta de uma DSL textual para modelagem conceitual de BDs.

Como principais resultados obtidos está a investigação conduzida para compreender o estado da arte e da prática no uso de DSLs para modelagem de BDs, bem como a criação de uma DSL textual utilizando o *framework* Xtext, ferramenta *open source* que permitiu a integração do protótipo a um RCP Eclipse.

Outro ponto que merece destaque foi o esforço dos pesquisadores envolvidos neste estudo para a realização de dois artigos, sendo um abordando o SLM e submetido ao *ER - International Conference on Conceptual Modeling* (ER'2019), e outro abrangendo toda o MLM, o qual foi submetido ao Simpósio Brasileiro de Bancos de Dados (SBBD'2019). Ambos os trabalhos estão em fase de avaliação.

Pretende-se que ao final do trabalho esta proposta não apenas realize modelagem, mas também a transformação dos modelos ER gerados em *scripts* SQL para diferentes tecnologias SGBDs, *e.g.* PostgreSQL, MySQL e SQL Server.

6.1 Trabalhos Futuros

Na fase atual desta pesquisa pode-se dizer que foram obtidos avanços, porém ainda é necessário realizar a finalização de diversos aspectos da proposta. Os principais pontos que devem ser investigados e desenvolvidos dizem respeito à transformação completa dos modelos conceituais para modelos lógicos, bem como a geração de *scripts* SQL que representem o modelo físico.

Pretende-se resolver estas questões com o andamento do trabalho e, com isto feito, realizar uma avaliação experimental da proposta. É importante salientar que a DSL será avaliada preliminarmente ainda no começo do TCC II, visando assim obter o refinamento da gramática da linguagem criada.

6.2 Cronograma

Para o restante da execução da pesquisa, o desenvolvimento deste trabalho se dará conforme as atividades descritas na Tabela 11.

Tabela 11 – Cronograma do Trabalho de Conclusão de Curso.

ATIVIDADE	**2019/1**					**2019/2**			
	MAR	**ABR**	**MAI**	**JUN**	**JUL**	**AGO**	**SET**	**OUT**	**NOV**
Elaboração da proposta de TCC									
Pesquisa bibliográfica da fundamentação teórica									
Planejamento e execução do MLM									
Análise de qualidade dos Resultados do MLM									
Extração e análise dos dados do MLM									
Implementação do protótipo									
Demonstração do protótipo									
Escrita do TCC I									
Aplicação das melhorias sugeridas pela banca de TCC1									
Planejamento da avaliação preliminar									
Execução da avaliação preliminar									
Análise da avaliação preliminar									
Evolução do desenvolvimento									
Planejamento da avaliação experimental									
Execução da avaliação experimental									
Análise dos resultados da avaliação experimental									
Escrita de artigos para submissão em eventos científicos									
Escrita do TCC II									

Fonte: O autor.

REFERÊNCIAS

AMELLER, D. **Considering Non-Functional Requirements in Model-Driven Engineering**. Dissertação (Mestrado) — Llenguatges i Sistemes Informàtics (LSI), June 2009. Disponível em: <http://upcommons.upc.edu/pfc/handle/2099.1/7192>. Citado na página 32.

ANSI, A. N. S. I. Interim report: ANSI/X3/SPARC study group on data base management systems 75-02-08. **FDT - Bulletin of ACM SIGMOD**, v. 7, n. 2, p. 1–140, 1975. Citado na página 25.

AYADI, M. G.; BOUSLIMI, R.; AKAICHI, J. A framework for medical and health care databases and data warehouses conceptual modeling support. **NetMAHIB**, v. 5, n. 1, p. 13, 2016. Citado 3 vezes nas páginas 46, 47 e 48.

BACKUS, J. W. The syntax and semantics of the proposed international algebraic language of the zurich acm-gamm conference. In: **IFIP Congress**. [s.n.], 1959. p. 125–131. Disponível em: <http://dblp.uni-trier.de/db/conf/ifip/ifip1959.html#Backus59>. Citado na página 34.

BAILER-JONES, D. **Scientific Models in Philosophy of Science**. University of Pittsburgh Press, 2009. Disponível em: <https://books.google.com.br/books?id=avur50J7UecC>. Citado na página 31.

Bailey, J. et al. Evidence relating to object-oriented software design: A survey. In: **First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)**. [S.l.: s.n.], 2007. p. 482–484. Citado na página 39.

BARKER, R. **CASE method - entity relationship modelling**. [S.l.]: Addison-Wesley, 1990. (Computer aided systems engineering). Citado na página 52.

BERGIN JR., T. J.; GIBSON JR., R. G. (Ed.). **History of Programming languages—II**. New York, NY, USA: ACM, 1996. Citado na página 34.

BRAMBILLA, M.; CABOT, J.; WIMMER, M. **Model-Driven Software Engineering in Practice, Second Edition**. [S.l.]: Morgan & Claypool Publishers, 2017. (Synthesis Lectures on Software Engineering). Citado 2 vezes nas páginas 31 e 32.

CELIKOVIC, M. et al. A DSL for EER data model specification. In: **Information Systems Development: Transforming Organisations and Society through Information Systems - Proceedings of the 23rd International Conference on Information Systems Development, ISD 2014, Varaždin, Croatia, September 2-4, 2014**. [S.l.: s.n.], 2014. Citado 3 vezes nas páginas 46, 47 e 48.

CHAMBERLIN, D. D.; BOYCE, R. F. Sequel: A structured english query language. In: **Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control**. New York, NY, USA: ACM, 1974. (SIGFIDET '74), p. 249–264. Citado na página 28.

CHEN, P. P.-S. The entity-relationship model—toward a unified view of data. **ACM Trans. Database Syst.**, ACM, New York, NY, USA, v. 1, n. 1, p. 9–36, mar. 1976. Citado 2 vezes nas páginas 26 e 52.

CODD, E. F. A relational model of data for large shared data banks. **Commun. ACM**, ACM, New York, NY, USA, v. 13, n. 6, p. 377–387, jun. 1970. ISSN 0001-0782. Citado na página 29.

COOK, T.; CAMPBELL, D. **Quasi-Experimentation: Design and Analysis Issues for Field Settings**. [S.l.]: Houghton Mifflin, 1979. Citado na página 51.

COUGO, P. **Modelagem conceitual e projeto de banco de dados**. [S.l.]: Elsevier Editora Ltda., 2013. Citado 3 vezes nas páginas 25, 26 e 28.

DATE, C. **Introdução a sistemas de bancos de dados**. [S.l.]: ELSEVIER EDITORA, 2004. Citado na página 25.

DATE, C.; WARDEN, A. **Relational database writings, 1985-1989**. [S.l.]: Addison-Wesley, 1990. (Relational database / C.J. Date, v. 1). ISBN 9780201508819. Citado na página 17.

DEURSEN, A. van; KLINT, P.; VISSER, J. Domain-specific languages: An annotated bibliography. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 35, n. 6, p. 26–36, jun. 2000. Disponível em: <http://doi.acm.org/10.1145/352029.352035>. Citado 2 vezes nas páginas 34 e 35.

DIMITRIESKI, V. et al. Concepts and evaluation of the extended entity-relationship approach to database design in a multi-paradigm information system modeling tool. **Comput. Lang. Syst. Struct.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, v. 44, n. PC, p. 299–318, dez. 2015. Citado 5 vezes nas páginas 37, 46, 47, 48 e 52.

ELMASRI, R.; NAVATHE, S. **Sistemas de Banco de Dados**. [S.l.]: PEARSON BRASIL, 2011. ISBN 9788579360855. Citado na página 17.

FAVERI, C. de. **Uma Linguagem Específica de Domínio para Consulta em Código Orientado a Aspectos**. Dissertação (Mestrado) — Universidade Federal de Santa Maria, aug 2013. Citado 2 vezes nas páginas 34 e 35.

FOWLER, M. Language workbenches: The killer-app for domain specific languages? 2005. Disponível em: <http://www.martinfowler.com/articles/languageWorkbench.html>. Citado na página 36.

FOWLER, M. **Domain Specific Languages**. 1st. ed. [S.l.]: Addison-Wesley Professional, 2010. Citado 2 vezes nas páginas 34 e 36.

FRANTZ, R. Z. **Enterprise Application Integration: An Easy-to-Maintain Model-Driven Engineering Approach**. Tese (Doutorado) — Universidad de Sevilla. Departamento de Lenguajes y Sistemas Informáticos, 2012. Doctoral Thesis. Citado na página 33.

Garousi, V. and Felderer, M. and Mäntylä, M. V. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. **Information & Software Technology**, v. 106, p. 101–121, 2019. Disponível em: <https://doi.org/10.1016/j.infsof.2018.09.006>. Citado na página 39.

HAMMER, M.; LEOD, D. M. Database description with sdm: A semantic database model. **ACM Trans. Database Syst.**, ACM, New York, NY, USA, v. 6, n. 3, p. 351–386, set. 1981. Citado 3 vezes nas páginas 46, 47 e 48.

HEUSER, C. **Projeto de banco de dados : Volume 4 da Série Livros didáticos informática UFRGS**. [S.l.]: Bookman, 2009. (Livros didáticos informática UFRGS). Citado 4 vezes nas páginas 26, 27, 28 e 54.

JAGANNATHAN, D. et al. Sim: A database system based on the semantic data model. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 17, n. 3, p. 46–55, jun. 1988. Citado 3 vezes nas páginas 46, 47 e 48.

KERSTEN, M. et al. Sciql, a query language for science applications. In: **Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases**. New York, NY, USA: ACM, 2011. (AD '11), p. 1–12. Citado 3 vezes nas páginas 46, 47 e 48.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [S.l.], 2007. Citado 2 vezes nas páginas 39 e 41.

KLEPPE, A. et al. **MDA Explained: The Model Driven Architecture : Practice and Promise**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. (Object technology). Citado na página 33.

KRISHNA, S. **Introduction to Database and Knowledge-base Systems**. [S.l.]: World Scientific, 1992. (Computer Science Series). ISBN 9789810206192. Citado na página 17.

LITWIN, W. et al. Msql: A multidatabase language. **Inf. Sci.**, Elsevier Science Inc., New York, NY, USA, v. 49, n. 1-3, p. 59–101, out. 1989. Citado 3 vezes nas páginas 46, 47 e 48.

MARTELLI, R.; FILHO, O.; CABRAL, A. de L. **Modelagem e banco de dados**. [S.l.]: Editora Senac São Paulo, 2018. (Informática). Citado na página 28.

MARTIN, J. **Application Development Without Programmers**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1982. Citado na página 34.

MAZAIRAC, W.; BEETZ, J. Bimql - an open query language for building information models. **Adv. Eng. Inform.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 27, n. 4, p. 444–456, out. 2013. Disponível em: <http://dx.doi.org/10.1016/j.aei.2013.06.001>. Citado 3 vezes nas páginas 46, 47 e 48.

MERNIK, M.; HEERING, J.; SLOANE, A. M. When and how to develop domain-specific languages. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 37, n. 4, p. 316–344, dez. 2005. Disponível em: <http://doi.acm.org/10.1145/1118890.1118892>. Citado na página 35.

NAKAGAWA, E. et al. **Revisão Sistemática da Literatura em Engenharia de Software: Teoria e Prática**. [S.l.]: Elsevier Editora Ltda., 2017. Citado 2 vezes nas páginas 39 e 40.

NARDI, B. A. **A Small Matter of Programming: Perspectives on End User Computing**. 1st. ed. Cambridge, MA, USA: MIT Press, 1993. Citado na página 34.

OBEO; TYPEFOX. **Xtext/Sirius - Integration The Main Use-Cases**. [S.l.], 2017. Disponível em: <https://www.obeodesigner.com/resource/white-paper/WhitePaper_XtextSirius_EN.pdf>. Citado na página 55.

PEFFERS, K. et al. A design science research methodology for information systems research. **J. Manage. Inf. Syst.**, M. E. Sharpe, Inc., Armonk, NY, USA, v. 24, n. 3, p. 45–77, dez. 2007. Citado na página 21.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: **EASE**. [S.l.: s.n.], 2008. v. 8, p. 68–77. Citado 2 vezes nas páginas 39 e 40.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico - 2ª Edição**. [S.l.]: Editora Feevale, 2013. Citado 2 vezes nas páginas 21 e 22.

RAMAKRISHNAN, R.; GEHRKE, J. **Database Management Systems**. 3. ed. New York, NY, USA: McGraw-Hill, Inc., 2003. ISBN 0072465638, 9780072465631. Citado na página 30.

ROSS, D. T. History of programming languages. In: WEXELBLAT, R. L. (Ed.). New York, NY, USA: ACM, 1981. cap. Origins of the APT Language for Automatically Programmed Tools, p. 279–338. Disponível em: <http://doi.acm.org/10.1145/800025.1198374>. Citado na página 34.

SALGADO, A. C.; MEDEIROS, C. B. Ensino de banco de dados. In: **Anais do Workshop de Informática na Escola**. [S.l.: s.n.], 1995. Citado na página 18.

SAMMET, J. E. Programming languages: History and future. **Commun. ACM**, ACM, New York, NY, USA, v. 15, n. 7, p. 601–610, jul. 1972. Disponível em: <http://doi.acm.org/10.1145/361454.361485>. Citado na página 34.

SCHMIDT, D. C. Guest editor's introduction: Model-driven engineering. **Computer**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 39, n. 2, p. 25–31, fev. 2006. Citado na página 34.

SHIPMAN, D. W. The functional data model and the data languages daplex. **ACM Trans. Database Syst.**, ACM, New York, NY, USA, v. 6, n. 1, p. 140–173, mar. 1981. Disponível em: <http://doi.acm.org/10.1145/319540.319561>. Citado 3 vezes nas páginas 46, 47 e 48.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de bancos de dados**. [S.l.]: Makron Books, 1999. ISBN 9788534610735. Citado na página 17.

SOMMERVILLE, I. **Engenharia de software**. [S.l.]: PEARSON BRASIL, 2011. Citado 2 vezes nas páginas 17 e 32.

TIAN, H. et al. Neuroql: A domain-specific query language for neuroscience data. In: **Proceedings of the 2006 International Conference on Current Trends in Database Technology**. Berlin, Heidelberg: Springer-Verlag, 2006. (EDBT'06), p. 613–624. Disponível em: <http://dx.doi.org/10.1007/11896548_46>. Citado 3 vezes nas páginas 46, 47 e 48.

Wachsmuth, G. H.; Konat, G. D. P.; Visser, E. Language design with the spoofax language workbench. **IEEE Software**, v. 31, n. 5, p. 35–43, Sep. 2014. Citado na página 36.

WAZLAWICK, R. **Metodologia de Pesquisa para Ciência da Computação**. [S.l.]: Elsevier Editora Ltda., 2017. Citado na página 21.

WEST, M. **Developing High Quality Data Models**. [S.l.]: Elsevier Science, 2011. Citado na página 28.

WEXELBLAT, R. L. (Ed.). **History of Programming Languages**. New York, NY, USA: ACM, 1981. Citado na página 34.

WOHLIN, C. et al. **Experimentation in Software Engineering**. [S.l.]: Springer Publishing Company, Incorporated, 2012. Citado na página 51.

ÍNDICE