# EERCASE: Uma Ferramenta Robusta para Projeto Conceitual de Banco de Dados

**Edson A. Silva, Robson N. Fidalgo**

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
CEP 50740-560 – Recife – PE – Brasil

{eas4, rdnf}@cin.ufpe.br

***Abstract.*** *Considering the plurality of Enhanced Entity–Relationship (EER) notations and the lack of robust tools, this article presents the EERCASE tool. Based on the best practices of the Model Driven Development (MDD) paradigm, EERCASE is able to prevent syntactically invalid constructions, point out and explain static semantic errors, and generate data definition code considering non-trivial constraints. EERCASE has been used to support EER modeling teaching and practice, as well as offers resources not available through other tools.*

***Resumo.*** *Considerando a pluralidade de notações Enhanced Entity–Relationship (EER) e a carência de ferramentas robustas, este artigo apresenta a ferramenta EERCASE. Construída com base nas melhores práticas do paradigma Model Driven Development (MDD), EERCASE é capaz de impedir construções sintaticamente inválidas, indicar e explicar os erros de semântica estática e gerar código de definição de dados impondo restrições não triviais. EERCASE tem sido usada para apoiar o ensino e a prática de projeto EER e oferece recursos não disponibilizados por outras ferramentas.*

## 1. Introdução

Em se tratando de Projeto Conceitual de Banco de Dados (BD), o modelo *Enhanced Entity-Relationship* (EER) é mais expressivo do que a alternativa de usar o Diagrama de Classe da UML [Bavota et al. 2011]. Contudo, a partir da sua proposta original [Chen 1976], várias reduções e extensões conceituais e gráficas surgiram [Song et al. 1995]. Dada essa pluralidade de notações, conceitos que existem em uma notação não existem em outra ou são interpretados de formas diferentes. Por exemplo: 1) a falta do conceito de atributo discriminador (i.e., chave parcial) exige que atributos identificadores sejam usados em relacionamentos ou entidades fracas, o que é uma contradição conceitual; 2) a não distinção entre links de generalização e especialização exige que as super-entidades estejam acima das sub-entidades, o que nem sempre é possível (cf. Figura 3-A); 3) notações que não permitem atributos em relacionamento ou relacionamentos n-ários dificultam a modelagem de domínios não triviais e 4) existem notações que fazem a leitura de cardinalidade e participação totalmente *look across*, totalmente *look here* ou híbrida [Song et al. 1995]. Vale destacar que notações com leitura de participação *look across* são incapazes de modelar sem ambiguidade esse conceito em relacionamentos n-ários, pois são imprecisas para representar a participação obrigatória de uma única entidade no relacionamento N-ário [Song et al. 1995], uma vez que a leitura *look across*

sempre considera N-1 entidades. Considerando estes quatro pontos e a sua boa aceitação pela comunidade de Banco de Dados (BD), este trabalho baseia-se na notação proposta por Elmasri e Navathe [Elmasri and Navathe 2016].

Proporcionalmente à pluralidade de notações EER, existem muitas ferramentas do tipo *Computer Aided Software Engineering* (CASE). Contudo, considerando a notação de Elmasri e Navathe, com exceção da EERCASE[1], se desconhece outra ferramenta CASE que ofereça suporte robusto ao projetista de BD. Neste contexto, o objetivo deste artigo é apresentar como a EERCASE oferece suporte para: 1) impedir erros sintáticos (i.e., construções gramaticalmente inválidas); 2) informar e descrever os erros de semântica estática (i.e., construções gramaticalmente válidas, mas estruturalmente contraditórias) e 3) gerar código em *Structured Query Language/Data Definition Language* (SQL/DDL) com regras para garantir restrições não triviais de participação obrigatória em relacionamentos (e.g., relacionamentos 1:N com participação obrigatória do lado 1) e totalidade e disjunção de heranças.

Sobre trabalhos relacionados, vale destacar que brModelo e TerraER são importantes ferramentas para projeto conceitual de BD, mas a notação usada na brModelo considera a abordagem *look across* para a leitura da participação de uma entidade no relacionamento e não tem o conceito de atributo discriminador e a TerraER não gera código SQL/DDL. Além disso, essas ferramentas não são 100% aderentes as regras de semântica estática propostas por Dullea, Song e Lamprou [Dullea; Song; Lamprou, 2003] e Calvanese e Lenzerini [Calvanese and Lenzerini 1994]. Por exemplo, brModelo e TerraER não são capazes de capturar os erros de semântica estática apresentados na Figura 2-B. Por fim, outras importantes ferramentas como DBDesigner, Erwin e ER/Studio não permitem modelar atributos em relacionamentos ou relacionamentos n-ários, pois não têm o construtor losango para representar um relacionamento.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta o arcabouço teórico e tecnológico e como estes foram usados para desenvolver a EERCASE. A Seção 3 descreve como usar a ferramenta e seus principais recursos. A Seção 4 versa sobre o potencial educacional da EERCASE. E, por fim, a Seção 5 apresenta as considerações finais e os trabalhos futuros.

## 2. Desenvolvimento da EERCASE

O arcabouço teórico usado para desenvolver a EERCASE é baseado no paradigma *Model-Driven Development* (MDD) [Brambilla et al. 2017], o qual, a partir de uma sintaxe concreta (i.e., notação gráfica), sintaxe abstrata (i.e., metamodelo ou gramática diagramática), semântica estática (i.e., regras de boa formação estrutural) e algoritmos para transformação de diagramas em código executável, especifica uma linguagem de modelagem e permite implementar a sua ferramenta CASE. Por sua vez, o arcabouço tecnológico usado é o seguinte: *Graphical Modeling Project* (GMP)[2] (usado para construir ferramentas CASE em Java/Eclipse) e Epsilon *Framework* (usado para

[1] https://cin.ufpe.br/~eercase (**canal no YouTube**: https://bityli.com/lLXkj)

[2] http://www.eclipse.org/modeling/gmp/.

simplificar a construção de ferramentas CASE em GMP)[3]. Além disso, a EERCASE utiliza XML *Metadata Interchange* (XMI) para armazenamento, manipulação, recuperação e intercâmbio de metadados e funciona como uma aplicação *standalone* utilizando a arquitetura *Rich Client Plataform* (RCP) da plataforma Eclipse, o que permite que esta seja independente de plataforma e de fácil distribuição.

O processo de desenvolvimento da EERCASE inicia com a codificação do metamodelo EERMM [Fidalgo et al. 2012] na linguagem Emfatic do Epsilon *Framework*. Ressalta-se que esse código, além da parte correspondente ao metamodelo EERMM (i.e., sintaxe abstrata da EERCASE), também contém as especificações gráficas da notação de Elmasri e Navathe (i.e., sintaxe concreta da EERCASE). Na sequência, a semântica estática é codificada usando a linguagem Epsilon *Validation Language* (EVL) e os algoritmos para transformação do diagrama EER e um script SQL/DDL (PostgreSQL) com gatilhos para garantir restrições não triviais de participação obrigatória mais as restrições de totalidade e disjunção de heranças são implementadas usando a linguagem Epsilon *Transformation Language* (ETL). Por fim, a ferramenta EuGENia, também do Epsilon *Framework*, é usada para abstrair os detalhes de configuração e execução do GMP.

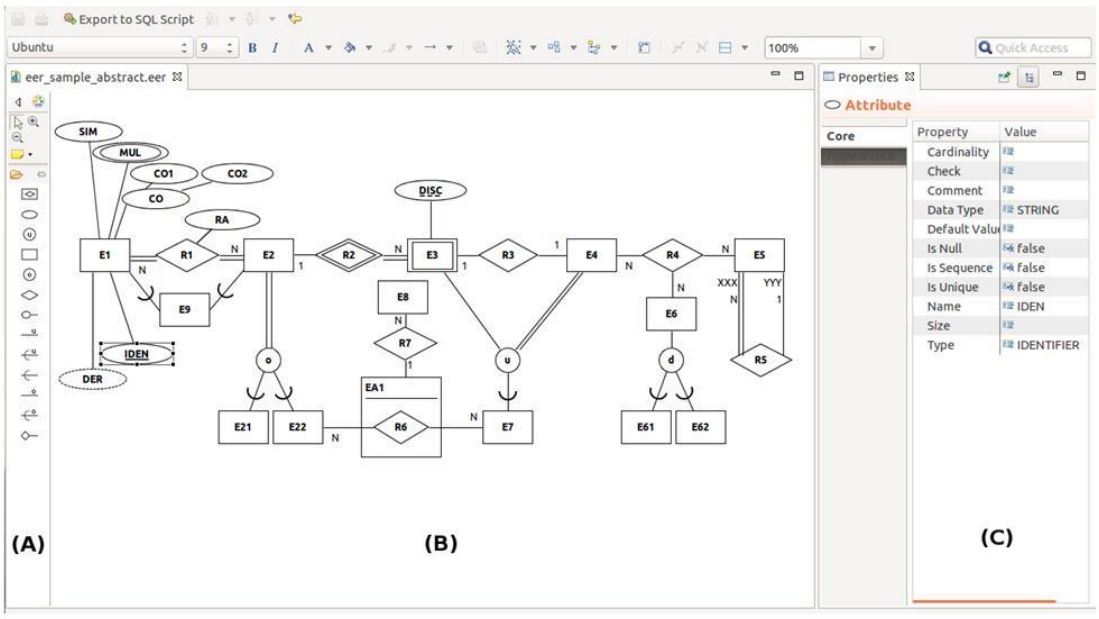
## 3. Uso da EERCASE

Na Figura 1, a interface gráfica da ferramenta EERCASE é apresentada. Nessa figura, a área “A” mostra os construtores da notação de Elmasri e Navathe, a área “B” a região de desenho do diagrama e, por fim, a área “C” as propriedades que podem ser configuradas para cada construtor EER selecionado no diagrama. A diagramação de um esquema EER começa com um clique no construtor desejado (área “A”) e depois no local que este deve aparecer na área de desenho (área "B"). Na sequência, podem-se editar as propriedades de um construtor selecionado (área "C") e adicionar novos construtores ao diagrama. De modo a mostrar a expressividade da EERCASE, na área B da Figura 1 é modelado um esquema simbólico e independente de domínio que faz uso de todos os construtores EER de Elmasri e Navathe. Além dos recursos apresentados, a EERCASE, por ser desenvolvida na plataforma Eclipse, também tem outros, por exemplo: criação de notas de texto, definição de cores para os construtores, visualização de vários esquemas em abas diferentes, controle de *zoom* e desfazer ou refazer uma ação. Vale destacar que a EERCASE é distribuída gratuitamente sob a licença Creative Commons com atribuição sem derivações (CC BY-ND 2.0 BR) para as plataformas Windows e Linux. Mais informações, incluindo link de download, vídeos demonstrativos, documentações e publicações, podem ser encontradas em http://www.cin.ufpe.br/~eercase.

Na Figura 2-A, são ilustrados exemplos de diagramas sintaticamente inválidos e impossíveis de serem desenhados com a EERCASE. Isto ocorre, pois o metamodelo EERMM funciona como uma gramática que define quais são as construções permitidas. Por sua vez, na Figura 2-B, são exemplificadas construções EER que, apesar de sintaticamente válidas, têm contradições estruturais segundo a semântica estática proposta por Dullea, Song e Lamprou, bem como por Calvanese e Lenzerini.

[3] http://www.eclipse.org/epsilon/doc/book/.

Na Figura 3-A mostra-se um esquema com um relacionamento 1:N com participação obrigatória do lado 1 e uma herança total e disjunta, enquanto que na Figura 3-B mostra-se apenas o trecho do código SQL do PostgreSQL gerado para impor a restrição de participação obrigatória do lado 1.




**(A) Construtores EER; (B) Região de desenho e (C) Propriedades de um construtor**

Figura 1: interface gráfica da ferramenta EERCASE

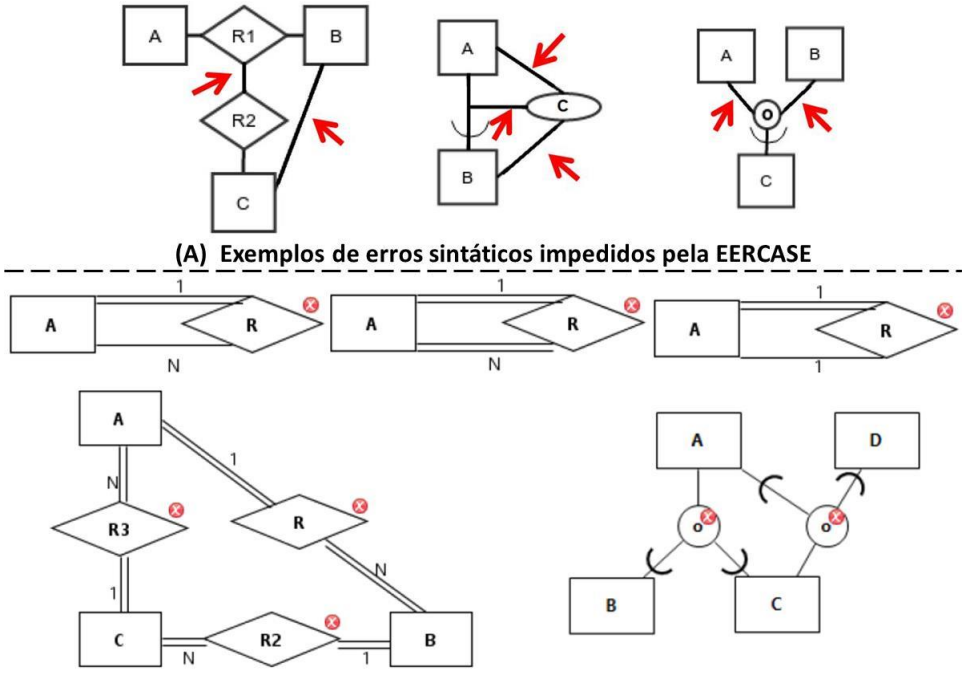
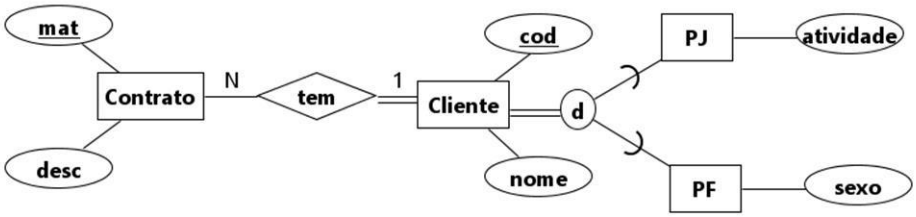



Figura 2: Exemplos de erros impedido e identificados pela EERCASE




(A) Esquema EER ilustrando restrições estruturais não triviais

```
CREATE FUNCTION FUNCTION_TRG_tem_Contrato()
RETURNS TRIGGER AS $BODY$
 DECLARE countCliente INTEGER;
 BEGIN
   SELECT COUNT(*) INTO countCliente
   FROM Contrato
   WHERE Contrato.tem_Cliente_cod = OLD.tem_Cliente_cod;
   IF countCliente = 0 THEN
      RAISE EXCEPTION 'TRG_tem_Contrato';
   END IF;
   IF (TG_OP = 'DELETE') THEN
      RETURN OLD;
   ELSIF (TG_OP = 'UPDATE') THEN
      RETURN NEW;
   END IF;
 END
$BODY$
LANGUAGE plpgsql VOLATILE;
CREATE CONSTRAINT TRIGGER TRG_tem_Contrato AFTER DELETE
OR
UPDATE OF tem_Cliente_cod ON Contrato INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE
FUNCTION_TRG_tem_Contrato();
```

```
CREATE FUNCTION FUNCTION_TRG_tem_Cliente()
RETURNS TRIGGER AS $BODY$
 DECLARE countCliente INTEGER;
 BEGIN
   SELECT COUNT(*) INTO countCliente
   FROM Contrato
   WHERE Contrato.tem_Cliente_cod = NEW.cod;
   IF countCliente = 0 THEN
       RAISE EXCEPTION 'TRG_tem_Cliente';
   END IF;
   RETURN NEW;
 END
$BODY$
LANGUAGE plpgsql VOLATILE;
CREATE CONSTRAINT TRIGGER TRG_tem_Cliente AFTER INSERT ON
Cliente INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE FUNCTION_TRG_tem_Cliente();
```

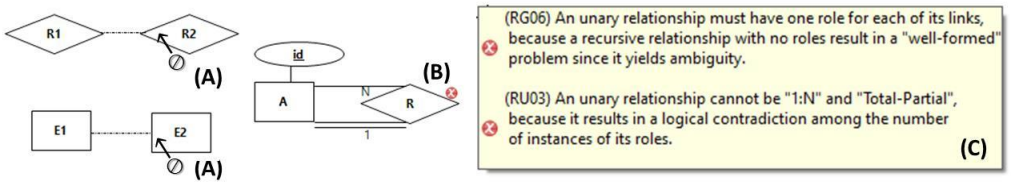
(B) Gatilho (PostgreSQL) para implementar a participação obrigatória do lado 1

Figura 3: Modelagem e geração de código de restrições não triviais em EERCASE

## 4. EERCASE como Ferramenta Educacional

Ferramentas CASE são utilizadas em várias áreas do conhecimento. Na educação, a partir dos feedbacks sobre os erros da modelagem de um dado domínio e de uma interação essencialmente prática, essas ferramentas ajudam a consolidar o aprendizado do estudante, pois permitem exercitar cenários que simulam situações reais. Isto é, ferramentas CASE fornecem um ambiente computacional que favorece o aprendizado ativo, pois têm como vantagem a capacidade de dar feedbacks individualizados e gerar código padronizado e livre de erros sintáticos e de semântica estática, recursos úteis para ajudar a reter o conteúdo aprendido por mais tempo e motivar a pesquisa de conhecimento adicional.

Desde 2012, a EERCASE tem seu uso recomendado no Centro de Informática da UFPE na disciplina básica de BD e, durante o período da pandemia da COVID-19, a EERCASE tem sido uma ferramenta útil para apoiar as atividades educacionais remotas. Isto é, a partir dos seus feedbacks gráficos que indicam se a construção é proibida (cf. Figura 4-A) e apontam onde está o erro (cf. Figura 4-B), bem como por meio das explicações dos erros (cf. Figura 4-C), a EERCASE tem apoiado os aprendizes em seus questionamentos, ajudando-os a lidar melhor com as particularidades do ensino remoto emergencial.




**(A) Feedback de erro sintático; (B) Feedback de erro de semântica estática e (C) Descrição de erro de semântica estática**

Figura 4: feedbacks gráficos exibidos pela EERCASE

## 5. Considerações Finais

É interessante que ferramentas CASE possam dar feedbacks sobre erros sintáticos e de semântica estática, além de gerar código executável considerando a implementação de restrições não triviais. Os feedbacks são recursos importantes para ensinar boas práticas de modelagem conceitual de BD, pois informam o que e onde estão ocorrendo os acertos e os erros. Por sua vez, a geração de código com regras não triviais, ajuda a aprender conceitos complexos. Portanto, ferramentas CASE com esses recursos favorecem o processo de aprendizagem, auxiliando a consolidar o conhecimento adquirido e estimulando a busca de novos conhecimentos. Neste contexto, este artigo apresenta a EERCASE como uma ferramenta robusta para projeto conceitual de BD. EERCASE é construída com base no paradigma MDD e no Epsilon *Framework*, e tem distribuição gratuitamente (CC BY-ND 2.0 BR) para Windows e Linux. Como trabalho futuro, tem-se a proposta de uma versão colaborativa na web.

## Referências

Bavota, G., Gravino, C., Oliveto, R., De Lucia, A., Tortora, G., Genero, M., and Cruz-Lemus, J. A. (2011). Identifying the weaknesses of uml class diagrams during datamodel comprehension. In International Conference on Model Driven Engineering Languages and Systems, pages 168–182. Springer.

Brambilla, M., Cabot, J., and Wimmer, M. (2017). Model-Driven Software Engineeringin Practice: Second Edition.Synthesis Lectures on Software Engineering, 3(1):1–207.

Chen, P. P.-S. (1976). The entity-relationship model—toward a unified view of data. ACM transactions on database systems (TODS), 1(1):9–36.

Calvanese, D., Lenzerini, M. On the interaction between ISA and cardinality constraints. In: Data Engineering, 1994. Proceedings.10th International Conference, v.14, no.18, p. 204-213, 1994.

Dullea, J., Song, I. Y., Lamprou, I. An analysis of structural validity in entity relationship modeling. Data and Knowledge Engineering, v. 47, n. 2, p. 167–205, 2003.

Elmasri, R. and Navathe, S. B. (2016). Fundamentals of Database Systems, Seventh Edition. Person, Boston, MA, USA.

Fidalgo, R. D. N., De Souza, E., España, S., De Castro, J., and Pastor, O. (2012). EERMM: A metamodel for the enhanced entity-relationship model. In Conceptual Modeling, volume 7532 of Lecture Notes in Computer Science, pages 515–524. Springer.

Song, Il-Yeol, Mary Evans, and Eun K. Park. A comparative analysis of entity-relationship diagrams. Journal of Computer and Software Engineering 3.4 (1995): 427-459.