colloquium exactarum

DESENVOLVIMENTO DE UMA FERRAMENTA PARA MAPEAMENTO DE UM MODELO CONCEITUAL DE DADOS PARA UM BANCO DE DADOS ORIENTADO A OBJETOS

DEVELOPMENT OF A TOOL FOR MAPPING A CONCEPTUAL DATA MODEL TO AN OBJECT-ORIENTED DATABASE

Janaína Iacia Silva[1], Aglaê Pereira Zaupa[2], Mário Augusto Pazoti[2]

[1]Discente da Faculdade de Informática da UNOESTE. [2]Docente da Faculdade de Informática da UNOESTE

RESUMO - Bancos de dados (BD) são agrupamentos de dados que são armazenados de forma persistente, ficando disponíveis para a realização de diversas operações sobre os dados inseridos. Para sua criação é necessário a construção de um Modelo Conceitual de Dados, que consiste em abstrair informações de determinado negócio (área de interesse), juntamente com regras para a criação das estruturas de dados. O presente trabalho propõe o desenvolvimento de uma ferramenta que permite a criação de uma modelagem conceitual de dados, mapeamento para um diagrama de classes e a criação do Script de um Banco de Dados Orientado a Objetos (BDOO), juntamente com suas respectivas classes em C#. A maioria dos elementos necessários para a modelagem conceitual e os elementos correspondentes a UML para o mapeamento foram implementados, bem como a utilização do BDOO Caché para demonstração.
Palavras-chave: modelagem de dados; banco de dados; banco de dados orientado a objetos.

ABSTRACT - Databases (DB) are groupings of data that are stored persistently, staying available to perform various operations on the data entered. For its creation it's necessary to build a conceptual data model, which consists in abstracting certain business information (interest area), along with rules for creation of the structures data. This paper proposes the development of a tool that enables the creation of a conceptual data modeling, mapping to a class diagram and the creation of an Object-Oriented Database (OODB) Script, along with their respective classes in C#. Most of the necessary elements for conceptual modeling and the corresponding elements of the UML for mapping were implemented, and the use of Caché OODB for demonstration.
Keywords: data modeling; database; object - oriented database.

Recebido em: 10/10/2013
Revisado em: 15/11/2013
Aprovado em: 19/12/2013

1 INTRODUÇÃO

Os Bancos de dados (BD) surgiram da necessidade de armazenar as informações de maneira persistente, além de padronizá-las (formato predeterminado). Proporcionam maior facilidade de manutenção, segurança e integridade dos dados, além de rapidez em consultas aos usuários no âmbito de manipulação dos dados. Considera-se que com o crescente avanço tecnológico, essa forma de armazenamento tornou-se uma das mais utilizadas, se comparado a outras, como por exemplo o sistema de arquivos. Date (2003, p.10) afirma que "Um banco de dados é uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa".

A criação de um banco de dados ocorre por meio de levantamentos e análises dos requisitos para a implementação de uma aplicação, e então a partir dos aludidos levantamentos faz-se a modelagem conceitual dos dados, que corresponde à abstração das informações relevantes para que o sistema seja construído.

O modelo entidade-relacionamento, considerado um padrão para a modelagem conceitual de dados, é a técnica mais utilizada para a modelagem de dados. Criado em 1976 por Peter Chen, sua representação é feita de forma gráfica chamada de D.E.R (Diagrama de Entidade-Relacionamento), onde Peter Chen possibilitou a representação de forma gráfica, simples e correta, conceitos que todos conhecem, porém não conseguiam expressá-los. Segundo Cougo (1997, p.34),

> Chen teve a chance de observar e enunciar aquilo que todos já sabiam mas estavam incapacitados de aplicar de modo concreto ao processo de modelagem: "Observamos, manipulamos, reproduzimos ou simulamos coisas, seus relacionamentos e características e, portanto, assim devemos representa-las!.

Com o avanço da tecnologia e dos dados a serem manipulados, surgiram os Bancos de Dados Orientado a Objetos (BDOO), que fornecem funcionalidades anteriormente inexistentes, possibilitando a manipulação de dados mais complexos. Em criação, objetivou-se trabalhar com o paradigma de programação orientada a objetos, encontradas nas linguagens como Java, C++, C#, entre outros. Assim como os bancos de dados relacionais possuem formas gráficas de representação, os bancos orientados a objetos são representados por meio de um diagrama de classes, sendo este definido por Booch, Rumbaugh e Jacobson (2005, p.107) como, "[...] um conjunto de classes, interfaces e colaborações e seus relacionamentos".

Existem vários software que fornecem recursos tanto para a criação de modelos conceituais de dados quanto para a

criação de diagramas de classes. Dentre estes foram estudados o brModelo (CÂNDIDO, 2011), Astah Community e Astah Professional (SILVA et al., 2010), onde o brModelo refere-se a criação de modelos conceituais de dados enquanto os Astah's referem-se à criação de diagrama de classes. Destaca-se que o Astah Professional contém todas as funções liberadas, sendo que este possui licença paga, comparadas as do Astah Community (antigo Jude Community), que este possui licença free.

Este artigo está disposto em cinco seções, a saber: a segunda apresenta conceitos sobre Modelagem de Dados; a terceira discorre sobre os Sistemas Gerenciadores de Bancos de Dados Orientado a Objetos (SGBDOO); a quarta engloba a Ferramenta desenvolvida; e por fim, a última seção refere-se às conclusões e sugestões para trabalhos futuros.

2 MODELAGEM DE DADOS

A modelagem de um banco de dados é uma descrição em uma linguagem específica dos objetos do mundo real, relações e características, linguagem esta que pode ser representada de maneira textual ou de maneira gráfica. Silberschatz, Korth e Sudarshan (2006, p.05) definem modelagem de dados (ou modelo de dados) como:

> [...] uma coleção de ferramentas conceituais para descrever dados, relações de dados, semântica de dados e restrições de consistência. Um modelo de dados oferece uma maneira de descrever o projeto de banco de dados no nível físico, lógico e de view.

Quando Silberschatz, Korth e Sudarshan (2006) falam sobre o nível físico, lógico e view (também conhecido como conceitual), referem-se às formas como os dados podem ser representados, visto que possuem diferentes regras para sua obtenção.

2.1 Modelagem Conceitual

Consiste na demonstração de um banco de dados. Segundo Cougo (1997, p.28):

> Define-se como modelo conceitual aquele em que os objetos, suas características e relacionamentos têm a representação fiel ao ambiente observado, independente de limitações quaisquer impostas por tecnologias, técnicas de implementação ou dispositivo físico.

O modelo conceitual refere-se a uma das etapas iniciais para a criação de um banco de dados, onde Machado e Abreu (2004, p.23) afirma que o modelo conceitual refere-se "[...] a primeira etapa do projeto de um sistema de aplicação em banco de dados." Nesta ocorre o levantamento de requisitos e a análise destes, realizando-se a abstração do maior número de informações

possíveis, para que os objetos do mundo real que espera-se armazenar e manipular sejam identificados.

Diz-se que o modelo conceitual deve ser utilizado no nível de conversação, onde o usuário transmite informações ao desenvolvedor sendo a recíproca também verdadeira. Durante esse nível, não se deve levar em conta a forma de implementação e nem qual a ferramenta a ser utilizada para a criação do banco (SGBD – Sistema Gerenciador de Banco de Dados), mas que o modelo conceitual gerado deverá ser imutável.

2.1.1 Modelo Entidade-Relacionamento (MER)

O MER é composto por três elementos básicos: Entidade, Atributos e Relacionamentos. Dentre as várias definições existentes, o MER consiste em mostrar ao usuário como os dados serão armazenados, demonstrando as entidades do banco de dados, os itens que elas possuem (atributos) e como essas entidades estão interligadas (relacionamentos). É trazer algo que não é concreto na visão dos clientes para o mundo real, facilitando assim o entendimento. Heuser (2009, p.72) afirma, “Um modelo ER é um modelo formal, preciso, não ambíguo. Isto significa que diferentes leitores do mesmo modelo ER devem sempre entender exatamente o mesmo”. Logo, mesmo que desenvolvedores projetem de diferentes formas um modelo de banco de dados, o entendimento entre os envolvidos devem ser de comum acordo.

As entidades são representações abstratas de objetos do mundo real, que deseja-se monitorar o comportamento, armazenando e recuperando suas informações. De acordo com a notação criada por Chen, um conjunto de entidades é representado por um retângulo e dentro deste o nome da entidade, como mostra a Figura 1. Cada elemento de um conjunto de entidades ocorre apenas uma vez.

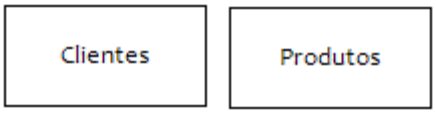


Figura 1. Representação de Entidades

Os atributos são informações, características ou propriedades relevantes de um conjunto de entidades. Dentre as várias classificações existentes para os atributos, serão abordados os tipos de atributos simples e identificadores, visto que estes foram implementados na ferramenta desenvolvida. Os atributos simples ou atômicos são indivisíveis e levam consigo uma única informação, já os atributos identificadores possuem valores únicos no conjunto de entidades.

Atributos são representados conforme a Figura 2, onde o círculo totalmente preenchido refere-se a um atributo identificador, e os demais círculos são atributos simples.

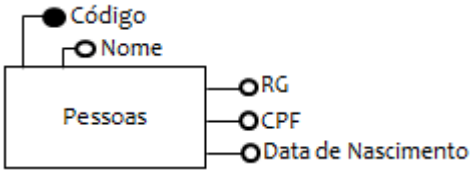


Figura 2. Representação de Atributos

Os relacionamentos descrevem a forma como as entidades estão interligadas. Heuser (2009, p. 36) define relacionamento como: "conjunto de associações entre ocorrências de entidades.". Relacionamentos podem ocorrer entre duas ou mais entidades, tendo sua representação conforme a Figura 3, com um losango possuindo o nome do relacionamento e as linhas que o ligam às entidades.

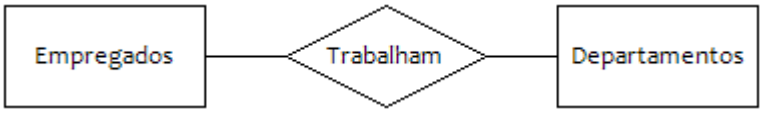


Figura 3. Representação de Relacionamento

Em conjunto a este conceito, existe o relacionamento de dependência, onde surge a entidade fraca, que por definição só existe a partir da existência de outra entidade forte. Assim, quando há a exclusão da entidade forte do relacionamento, a entidade fraca também deve ser excluída. A entidade fraca é representada com dois retângulos como segue a Figura 4.

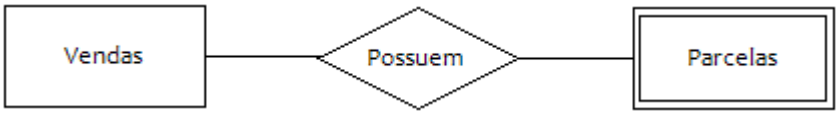


Figura 4. Representação de Entidade Fraca

A entidade associativa é a redefinição de um relacionamento, que passa a ser tratado como se fosse também uma entidade" (HEUSER, 2009, p.60). Esse conceito existe, pois o modelo entidade-relacionamento não suportam relacionamento entre relacionamentos. Sua representação está ilustrada na Figura 5.

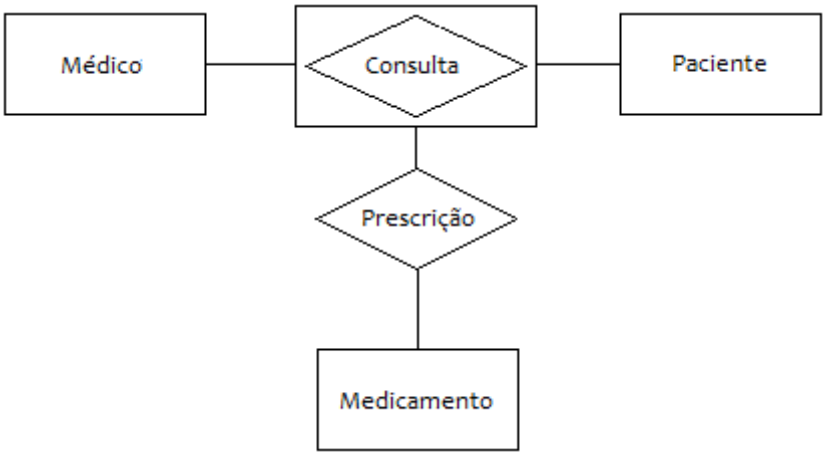


Figura 5. Representação de Entidade Associativa

Fonte: Heuser, 2009

A cardinalidade do relacionamento define a quantidade máxima e mínima de vezes que uma entidade pode ocorrer no relacionamento. Conforme afirmação de Silberschatz, Korth e Sudarshan (2006, p.139):

> [...] fatores de cardinalidade, expressão o número de entidades ao qual outra entidade pode ser associada por um conjunto de relacionamento.[...] A cardinalidade de mapeamento apropriada para determinado conjunto de relacionamento, obviamente, depende da situação real que o conjunto de relacionamento está modelando.

Existem dois tipos de cardinalidade: a máxima que expressa a quantidade máxima de ocorrências de uma entidade no relacionamento, podendo ser 1 (um) ou N (muitos); e a mínima, a qual corresponde a quantidade mínima de ocorrência de uma entidade no relacionamento. Segundo Heuser (2009, p.45), "Para fins de projeto de BD, consideram-se apenas duas cardinalidades mínimas: a cardinalidade mínima 0 e a cardinalidade mínima 1."

As cardinalidades são representadas juntamente com as linhas que interligam as entidades no relacionamento, da seguinte forma: cardinalidade "(mínima, máxima)" como segue na Figura 6, onde tem-se como leitura que um empregado trabalha em nenhum ou no máximo um departamento, e um departamento possui pelo menos um ou muitos (vários) empregados.

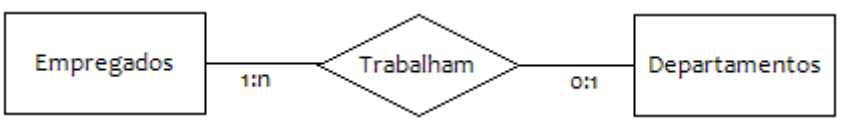


Figura 6. Representação de Cardinalidade

2.1.2 Modelo Entidade-Relacionamento Estendido (EER)

Como o próprio nome sugere, é uma expansão do modelo Entidade – Relacionamento (ER), onde novos conceitos foram inseridos. Criado com o intuito de permitir que os bancos de dados sejam mais precisos, para assim modelar sistemas com requisitos mais complexos, diferenciados das aplicações tradicionais (ELMASRI; NAVATHE, 2011, p.161).

No modelo estendido alguns conceitos foram acrescentados: como a herança e a agregação.

A herança compreende os seguintes conceitos: Superclasses, Subclasses, Generalização/Especialização.

As superclasses são entidades em um nível superior, que a partir de subdivisões (derivações), geram as subclasses, entidades de nível inferior, surgindo assim o conceito de especialização, onde as superclasses possuem os atributos comuns entre as subclasses, e as subclasses possuem atributos parcialmente distintos. Segundo Elmasri e Navathe (2011, p.163) "O conjunto de subclasses que forma uma especialização é definido com base em algumas características distintas das entidades na superclasse".

Já a generalização consiste no processo inverso ao da especialização. Silberschatz, Korth e Sudarshan (2006, p.154) afirmam que:

> A generalização ocorre a partir do reconhecimento de que diversos conjuntos de entidades compartilham os mesmos recursos (ou seja, eles são descritos pelos mesmos atributos e participam nos mesmos conjuntos de relacionamentos). Na base de suas semelhanças, a generalização sintetiza esses conjuntos de entidades em um único conjunto de entidades de nível superior. [...] ela também permite a economia de representação em que atributos compartilhados não são repetidos.

Tais conceitos aludidos são representados conforme a Figura 7.

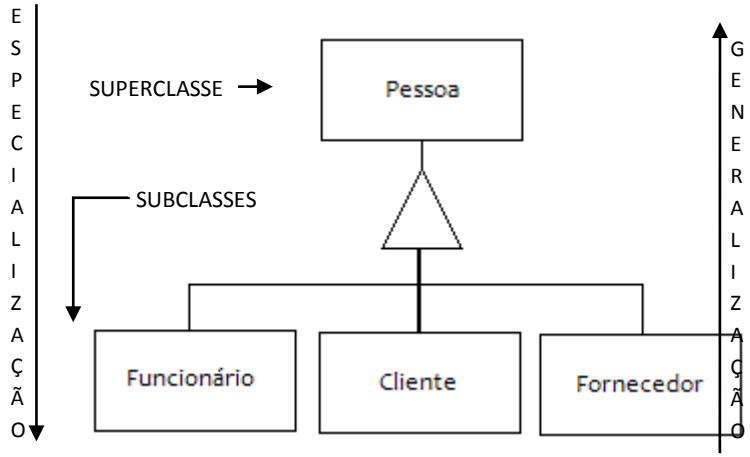


Figura 7. Representação de Herança

A agregação é um caso particular de associação, um relacionamento estrutural possuindo regras para sua determinação. Existem três casos relacionados ao seu uso no modelo EER. O primeiro diz respeito à agregação de valores a atributos para a criação de um objeto (entidade). O segundo trata de uma representação comum de um relacionamento, sendo este um relacionamento de agregação. Já o terceiro implica na possibilidade de combinar objetos relacionados por meio de uma instância de um relacionamento em particular em um objeto agregado de alto nível (ELMASRI; NAVATHE, 2011).

Cougo (1997, p.131) define agregação como "[...] outro recurso que poderá ser aplicado aos modelos para facilitar o entendimento semântico e tornar mais claros os graus de relacionamentos ternários, ou de maior número".

Para representar uma agregação toma-se como exemplo a Figura 8.

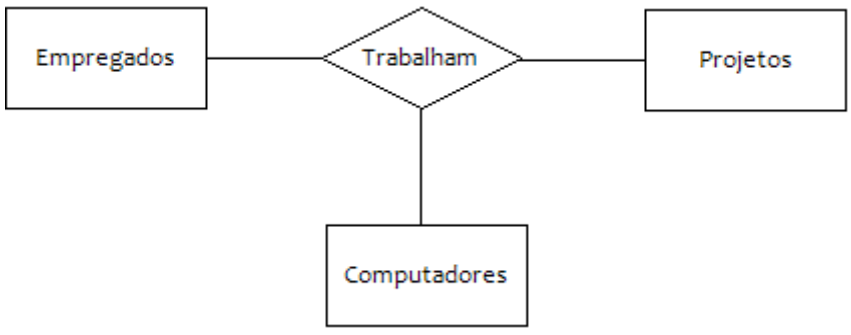


Figura 8. Exemplo Relacionamento Ternário

Com a existência de um relacionamento ternário, é obrigatória a ocorrência das três entidades em questão, porém em uma situação real nem sempre há a necessidade de um computador estar

ligado ao relacionamento Empregados-Projetos, surgindo assim a agregação, como mostra a Figura 9.

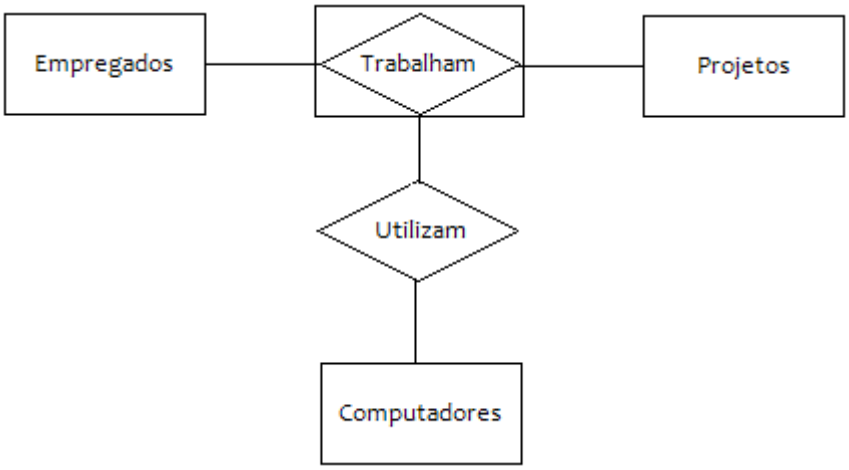


Figura 9. Diagrama ER com Agregação

Este diagrama pode ser lido da seguinte maneira: Empregados trabalham em projetos e juntamente a este relacionamento pode-se ter ou não a utilização de computadores. A figura representa um relacionamento binário, onde uma entidade chama-se computadores e a outra entidade engloba o relacionamento Empregados–Projetos.

2.2 Modelagem de Dados Orientado a Objetos

A modelagem de dados orientada a objetos, atualmente tem como padrão a UML (Unified Modeling Language), criada com base nos conceitos da abordagem E.R (entidade-relacionamento) (HEUSER, 2009).

Passou a existir a partir do momento que surgiram as linguagens orientadas a objetos e pessoas começaram a pensar em maneiras gráficas para sua representação. Referida representação foi criada de diversas formas e por diferentes autores, porém com a mesma essência.

Segundo Fowler (2005, p.25) “A UML nasceu da unificação das muitas linguagens gráficas de modelagem orientadas a objetos que floresceram no final dos anos oitenta, início dos noventa”.

2.2.1 Diagrama de Classes

O diagrama de classes é utilizado em modelagens de sistemas OO, no entanto também em modelagens de banco de dados, visto que alguns objetos (classes) devem ser persistentes, armazenados assim em um banco de dados (relacionais, orientados a objetos ou objetos relacionais).

São compostos por classes, interfaces, colaborações e seus relacionamentos, sendo base para outros diversos diagramas (BOOCH; RUMBAUGH; JACOBSON, 2005).

Booch, Rumbaugh e Jacobson (2005, p.112) afirmam que:

> A UML é adequada para a modelagem de esquemas lógicos dos bancos de dados, além dos próprios bancos de dados físicos. Os diagramas de classes da UML são um superconjunto dos diagramas entidade-relacionamento (E.R), uma ferramenta básica de modelagem para o projeto lógico de banco de dados. Enquanto os diagramas E.R clássicos tem seu foco apenas nos dados, os diagramas de classes vão um pouco além, permitindo ainda a modelagem de comportamentos.

O diagrama de classes é composto por vários elementos. No entanto, nesse trabalho serão utilizados as classes, atributos, operações, classes associativas, relacionamentos, multiplicidade e herança.

As **classes** segundo Booch, Rumbaugh e Jacobson (2005, p.49) são definidas como "[...] uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica". Os **atributos** são características relevantes de uma classe e as **operações** referem-se aos métodos, serviços por ela oferecidos. As **classes associativas** são classes que estão ligadas a um relacionamento de associação, sendo utilizadas para armazenar as informações das classes referentes ao relacionamento. Os **relacionamentos** ou **associações** indicam como as classes são interligadas. A **agregação** é uma associação, porém possui o conceito de "todo – parte", onde a parte existe independente do todo. Já a **composição** consiste em uma variação da agregação, pois é uma associação em que a parte não existe sem o todo.

Todo tipo de associação possui a **multiplicidade** que indica a quantidade de instâncias de outra classe que uma classe pode possuir. Segundo Booch, Rumbaugh e Jacobson (2005, p.129), "A multiplicidade é a especificação do intervalo permitido de cardinalidade que uma entidade poderá assumir". Finalizando a **herança** (generalização) que indica a capacidade de um objeto (classe) herdar os atributos e comportamentos referentes a outro objeto do mesmo tipo. Podendo ser nomeadas como classes genéricas e classes específicas. A Figura 10 representa os conceitos citados acima.

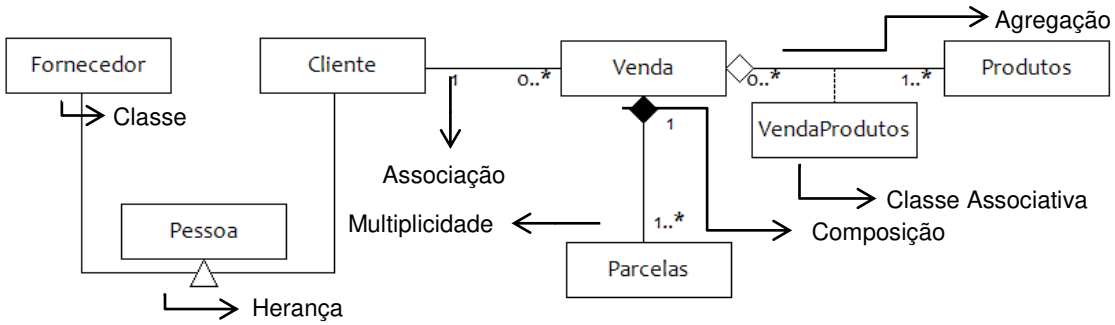


Figura 10. Exemplo de Diagrama de Classes

3 SISTEMAS GERENCIADORES DE BANCO DE DADOS ORIENTADO A OBJETOS

Os Sistemas Gerenciadores de Bancos de Dados Orientados a Objetos (SGBDOO) tem como uma de suas origens as linguagens de programação orientadas a objetos, as quais possibilitam a manipulação dos dados como objetos de maneira transparente. Frente a isto, pesquisadores investigaram maneiras de incorporar os recursos referentes a estas linguagens, entre outros, nos banco de dados, surgindo assim os bancos de dados orientados a objetos (DATE, 2003).

Date (2003) diz que alguns dos termos referentes à abordagem de objetos mais importantes são os objetos, as classes, métodos e mensagens.

Os objetos dizem respeito a abstrações referentes a elementos do mundo real, os quais possuem atributos e comportamentos, que possibilitam o armazenamento de suas informações. Segundo Elmasri e Navathe (2011, p.237): "Um objeto normalmente possui dois componentes: estado (valor) e comportamento (operações)". Os objetos possuem um conceito de identidade do objeto (OID) onde, cada objeto possui um identificador único, gerado pelo SGBOO. A identidade única é gerada com o intuito de os objetos não perderem sua integridade, além de facilitar sua identificação (ELMASRI; NAVATHE, 2011). Como em um banco de dados existem vários objetos similares, com os mesmos nomes, tipos e métodos, torna-se inviável a criação de uma classe para cada um, logo a classe de objetos ocorre pela junção de objetos que possuem atributos e comportamentos em comum.

Neste tipo de abordagem os objetos são encapsulados, ou seja, não é possível ter o acesso à estrutura física dos objetos, somente das operações (métodos) que estes fornecem, além das mensagens que Date (2003) define como a chamada dos métodos.

A persistência consiste em uma das características principais de bancos orientados a objetos, visto que a função do banco de dados é armazenar informações

para posterior recuperação. Esta característica fornece a diferenciação entre banco de dados orientados a objetos e as linguagens de programação orientadas a objetos, já que nas linguagens ao finalizar a execução de um programa, os objetos anteriormente instanciados deixam de existir. A herança indica que um objeto é capaz de herdar os atributos e comportamentos referentes a outro objeto, podendo esta ser única, onde um objeto herda as características de outro objeto, ou herança múltipla, em que um objeto pode herdar características de um ou mais objetos. Por fim o polimorfismo, que refere-se a métodos criados com o mesmo nome, porém com diferentes comportamentos, para que sejam aplicados a diferentes tipos de objetos (ELMASRI; NAVATHE, 2011).

4 FERRAMENTA

O mapeamento conceitual de dados para um banco de dados orientado a objetos é obtido a partir de uma modelagem conceitual de dados, a qual deve ser realizada utilizando a ferramenta implementada.

A ferramenta proposta neste trabalho foi desenvolvida na linguagem C#, utilizando como ambiente de desenvolvimento o Microsoft Visual Studio Ultimate 2012. Possibilita a criação de uma modelagem conceitual de dados, seu mapeamento para a modelagem orientada a objetos (diagrama de classes), e a partir desta a criação do script do banco de dados orientado a objetos Caché, e a criação das classes em linguagem C#. Adicionalmente, há a possibilidade das modelagens serem salvas e recuperadas, utilizando a serialização/ desserialização em binário.

4.1 Estudo de Caso

Para facilitar a apresentação da ferramenta e dos conceitos envolvidos, foi definido um estudo de caso que será utilizado como exemplo nas seções subsequentes. A Figura 11 apresenta a interface inicial da ferramenta desenvolvida e ilustra este estudo de caso que corresponde a um sistema acadêmico.

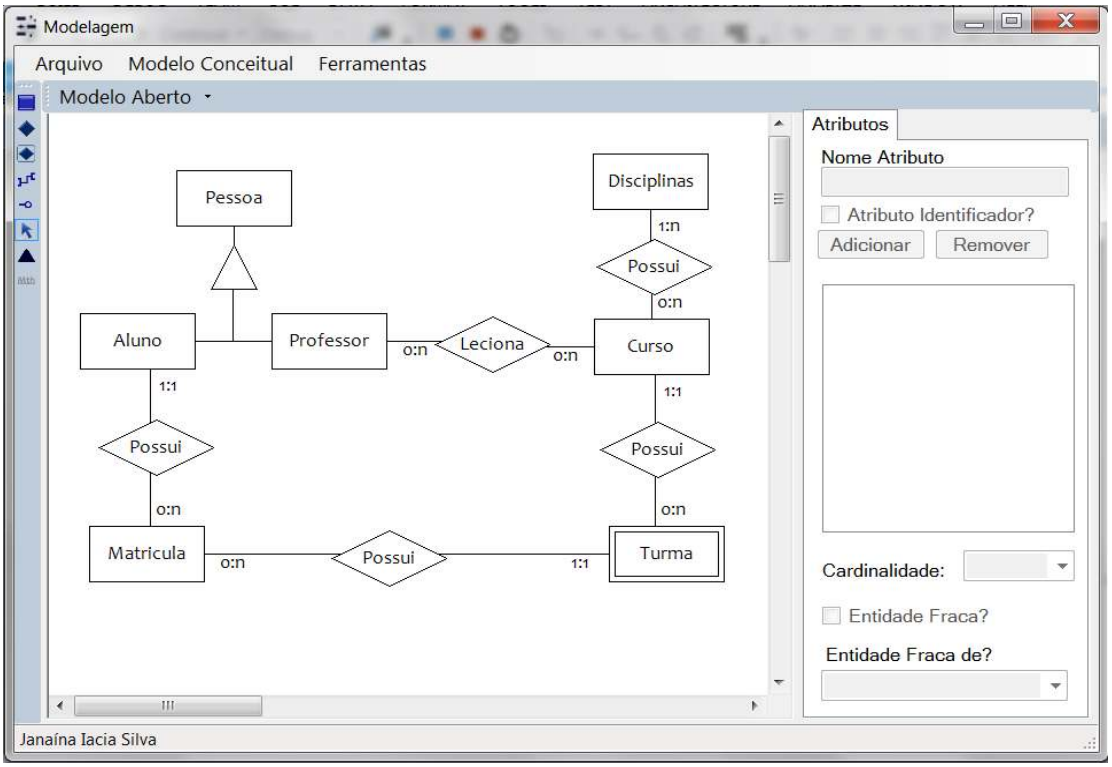


Figura 11. Estudo de Caso utilizando a Ferramenta Desenvolvida

4.2 Modelagem Conceitual

Referente à modelagem conceitual foram implementados os conceitos de entidades (fracas e associativas), atributos (identificadores e simples ou atômicos), relacionamentos, cardinalidades e herança (generalização/ especialização). Para gerenciar o desenvolvimento foi definido um diagrama de classes contendo a classe abstrata IComponente com os atributos e métodos, bem como as classes especializadas (Entidade, EntidadeAssociativa, Relacionamento e GeneralizacaoEspecializacao) que herdam seus atributos e sobrescrevem os métodos, pois cada classe possui diferentes implementações, além de possuírem atributos e métodos próprios, como ilustra a Figura 12.

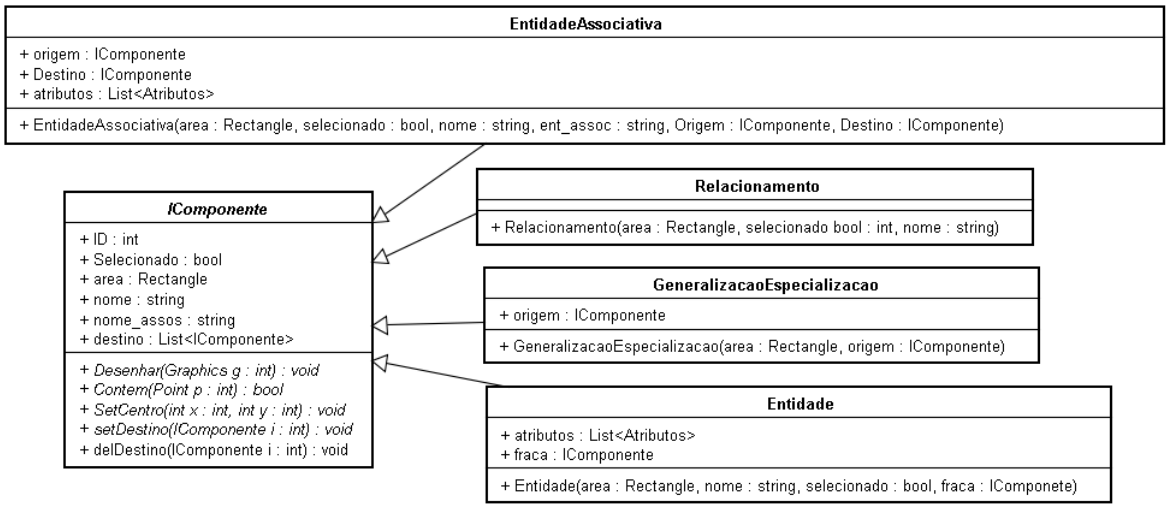


Figura 12. Diagrama de Classes Implementação Modelo Conceitual

A ferramenta permite a inserção dos elementos referentes à modelagem conceitual citados por meio da seleção do botão com a opção desejada, encontrados do lado esquerdo da ferramenta ou no menu Modelo Conceitual como apresenta a Figura 11. Tais elementos podem ser renomeados, caso possuam nome (entidades/entidades associativas/ relacionamentos), adicionados atributos (entidades/ entidades associativas), reposicionados conforme o local desejado em tela, e excluídos se necessário.

Como não é sempre conveniente a representação dos atributos juntamente com o D.E.R, sobrecarregando visualmente o modelo (HEUSER, 2009), os atributos na ferramenta ficam ocultos, sendo exibidos apenas no momento em que a entidade ou classe está selecionada.

4.3 Mapeamento para Diagrama de Classes

O mapeamaneto para a modelagem orientada a objetos gera o diagrama de classes, onde sua implementação engloba os elementos da UML correspondentes ao modelo conceitual. Para o mapeamento foi definido um diagrama de classes com a classe abstrata ComponentesOO e as classes especializadas (Classe, ClasseAssociativa, Herança), que herdam seus atributos e métodos, conforme a Figura 13.

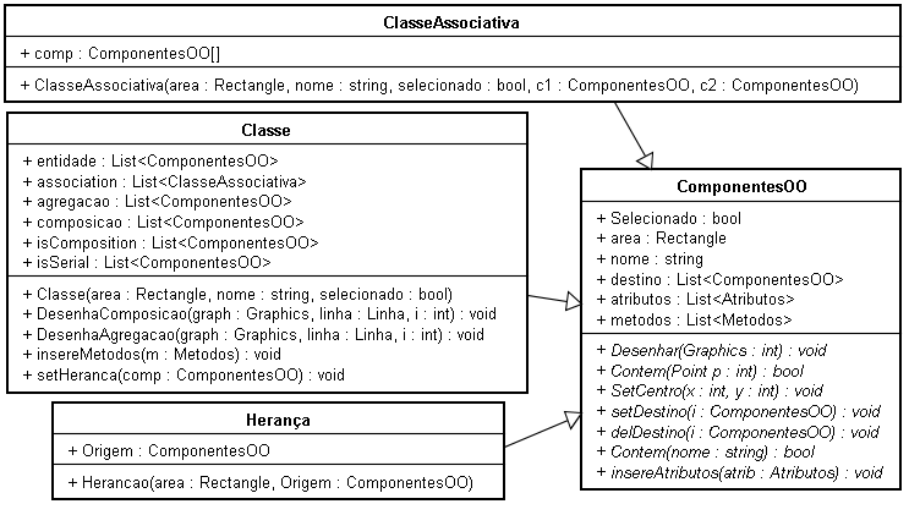


Figura 13. Diagrama de Classes Implementação Modelo OO

Na modelagem do diagrama de classes, os recursos disponíveis são a inserção de atributos e métodos nas classes e classes associativas e a definição dos tipos de atributos para a criação do script do banco OO e das classes em C#.

O mapeamento dos elementos do modelo conceitual para o diagrama de classes ocorre da seguinte maneira:

a) Entidades e seus atributos tornam-se classes e atributos da classe;

b) Relacionamento que envolve uma entidade fraca considera-se como composição;

c) Herança nas duas modelagens possui a mesma representação, no caso o triângulo, porém no diagrama de classes, este sempre estará conectado diretamente a classe “genérica”;

d) A ocorrência da cardinalidade (1,N) no relacionamento possui duas formas de interpretação no diagrama de classes, podendo este relacionamento ser dos tipos associação ou agregação, onde a ferramenta permite que o usuário escolha o tipo de desejado, visto que Booch, Rumbaugh e Jacobson (2005, p.69) conceituam a agregação como um relacionamento do tipo “tem-um”, indicando a ocorrência mínima, porém não definindo a máxima. Caso esse tipo de cardinalidade não ocorra e o relacionamento não for uma composição, ele é considerado como associação;

e) Na ocorrência de um relacionamento muitos-para-muitos a classe associativa é criada, ligada diretamente ao relacionamento.

Como resultado tem-se a Figura 14, que mostra o diagrama de classes gerado a partir do mapeamento, considerando a modelagem realizada na Figura 11.

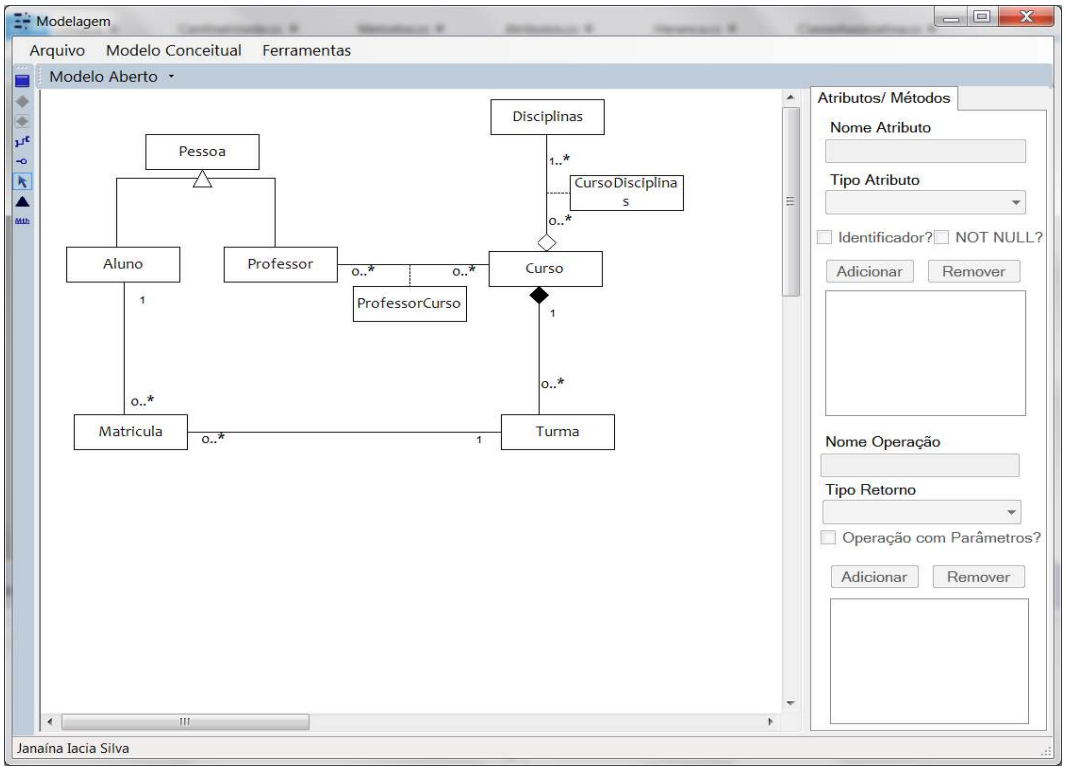


Figura 14. Diagrama de Classes Resultante apresentado pela Ferramenta Desenvolvida

Tanto a modelagem conceitual como a orientada a objetos, utilizam as classes Linha, Atributos e Cardinalidade (utilizada pela classe Linha), já a classe Metodo é utilizada apenas pelo modelo orientado a objetos, as quais são definidas conforme a Figura 15.

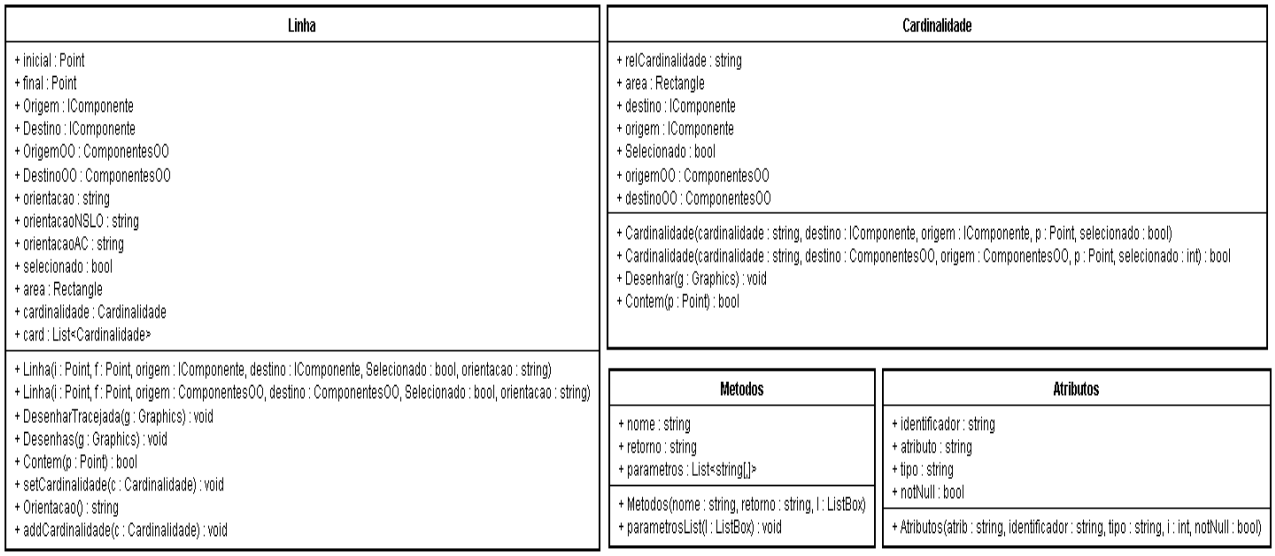


Figura 15. Classes Herdadas pelos Modelos na Ferramenta

4.4 Script Orientado a Objetos

A partir do diagrama de classes há a possibilidade de gerar um script para o banco

de dados orientado a objetos Caché, que possui funcionalidades como a criação de bases de dados orientadas a objetos, seu próprio ambiente de programação para criação de aplicações web, acesso aos dados tanto em forma de objetos como em SQL, sendo acessados de forma direta ou com a utilização de drivers ODBC, JDBC, das linguagens Java, C# e C++.

A documentação do Caché indica algumas vantagens de utilização, dentre elas o desempenho, escalabilidade, desenvolvimento rápido, flexibilidade, entre outras (INTERSYSTEMS..., 2008).

Os scripts podem ser gerados de duas maneiras: por meio de um documento XML, o qual é possível sua importação no SGBDOO, onde as classes serão reconhecidas e criadas separadamente; e por meio das classes .cls (extensão das classes criadas no Caché), porém neste formato, o usuário necessita criar cada classe manualmente no Caché e alimentá-las com os códigos gerados pela ferramenta. Estes são ilustrados a seguir de acordo com a Figura 16 e Figura 17, que referem-se à representação de uma das classes do diagrama de classes resultante do estudo de caso.

```
<?xml version="1.0"?>
- <Export zv="Cache for Windows (x86-64) 2012.1.2 (Build 702)" version="25" generator="Cache">
  - <Project name="ScriptArtigo">
    - <Items>
        <ProjectItem name="User.Pessoa" type="CLS"/>
        <ProjectItem name="User.Aluno" type="CLS"/>
        <ProjectItem name="User.Professor" type="CLS"/>
        <ProjectItem name="User.Matricula" type="CLS"/>
        <ProjectItem name="User.Turma" type="CLS"/>
        <ProjectItem name="User.Curso" type="CLS"/>
        <ProjectItem name="User.Disciplinas" type="CLS"/>
        <ProjectItem name="User.ProfessorCurso" type="CLS"/>
        <ProjectItem name="User.CursoDisciplinas" type="CLS"/>
      </Items>
    </Project>
  + <Class name="User.Pessoa">
  + <Class name="User.Aluno">
  - <Class name="User.Professor">
      <Super>Pessoa</Super>
    - <Property name="dataAdmissao">
        <Type>%Date</Type>
      </Property>
    - <Property name="salarioProf">
        <Type>%Double</Type>
      </Property>
    - <Property name="matriculaProf">
        <Type>%Integer</Type>
      </Property>
    - <Property name="ProfessorCurso">
        <Type>User.ProfessorCurso</Type>
        <Cardinality>many</Cardinality>
        <Inverse>Professor</Inverse>
        <Relationship>1</Relationship>
      </Property>
    </Class>
  + <Class name="User.Matricula">
  + <Class name="User.Turma">
  + <Class name="User.Curso">
  + <Class name="User.Disciplinas">
  + <Class name="User.ProfessorCurso">
  + <Class name="User.CursoDisciplinas">
</Export>
```

Figura 16. Script XML Gerado para Banco de Dados Caché

```
Class User.Professor Extends Pessoa
{

Property dataAdmissao As %Date;

Property salarioProf As %Double;

Property matriculaProf As %Integer;

Relationship ProfessorCurso As User.ProfessorCurso [ Cardinality = many, Inverse = Professor ];

}
```

Figura 17. Script Classe Gerado para o BD Caché Professor.cls

4.5 Classes C#

Geradas a partir do diagrama de classes, são embasadas no conceito de programação orientada a objetos. As classes possuem os atributos e métodos atribuídos a cada uma pelo usuário no momento da modelagem, e os objetos das classes, referente as classes que esta possui relacionamento, definidos através da multiplicidade pré-determinada. Para multiplicidades 1 e 0..1 apenas um objeto é referenciado na classe, já para multiplicidades 0..* e 1..* cria-se um List<> (vetor dinâmico existente na linguagem C#), onde indica que a classe pode possuir mais de um objeto referente a outra classe. A Figura 18 apresenta a estrutura da classe Professor gerada pela ferramenta.

```
public class Professor : Pessoa
{
    private int ID;
    private DateTime dataAdmissao;
    private double salarioProf;
    private int matriculaProf;
    private List<ProfessorCurso> professorcurso;

    public bool inserir (DateTime dataAdmissao ,double salarioProf ,int matriculaProf)
    {
    }
    public bool alterar (int ID ,DateTime dataAdmissao ,double salarioProf ,int matriculaProf)
    {
    }
    public bool excluir (int ID)
    {
    }
}
```

Figura 18. Classe em C# Professor

5 CONCLUSÕES

O projeto proporciona um aprofundamento em estudos sobre o paradigma orientado a objetos e as técnicas de modelagem relacionadas, possibilitando a

criação de artefatos para a realização da análise orientada a objetos.

Apresenta uma alternativa para a modelagem conceitual de dados e seu mapeamento para um diagrama de classes, auxiliando os usuários de banco de dados orientados a objetos no momento da modelagem, além de facilitar a programação orientada a objetos a partir das classes geradas em C#.

Como trabalhos futuros, alguns conceitos referentes à modelagem conceitual podem ser implementados, como: atributos compostos, multivalorados e derivados, relacionamentos ternários e autorrelacionamento. Já na modelagem orientada a objetos, a implementação dos elementos correspondentes, entre outros existentes na UML, além da criação e manipulação partindo da modelagem orientada a objetos, não necessitando a criação de um modelo conceitual. Acrescentar a criação de classes referentes a outras linguagens como Java e C++ e a criação de scripts para outros bancos orientados a objetos como Órion, db4O (db4 Object), entre outros.

REFERÊNCIAS

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML** – Guia do usuário. 2. ed. Rio de Janeiro: Elsevier, 2005.

CÂNDIDO, C.H. **brModelo:** ferramenta de modelagem conceitual de banco de dados. 2011. 35f. Monografia (Pós-Graduação em Banco de Dados) - UNIVAG - MT e Universidade Federal de Santa Catarina.

COUGO, P. **Modelagem conceitual e projeto de banco de dados**. Rio de Janeiro: Campus, 1997.

DATE, C.J. **Introdução a sistemas de banco de dados**. 8. ed. Rio de Janeiro: Elsevier, 2003.

ELMASRI, R.; NAVATHE, S.B. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson, Addison Wesley, 2011.

FOWLER, M. **UML essencial:** um breve guia para a linguagem-padrão de modelagem de objetos. 3. ed. Porto Alegre: Bookman, 2005.

HEUSER, C.A. **Projeto de banco de dados**. 6. ed. Porto Alegre: Bookman, 2009.

INTERSYSTEMS CORPORATION. **Manual de tecnologia caché**. 2008. Disponível em: <http://www.intersystems.com.br/static/pdf/manual-cache.pdf>. Acesso em: 26 out. 2012.

MACHADO, F.; ABREU, M. **Projeto de banco de dados** – uma visão prática. 11. ed. São Paulo: Érica, 2004.

SILBERSCHATZ, A.; KORTH, H.F.; SUDARSHAN, S. **Sistema de banco de dados**. 5. ed. Rio de Janeiro: Campus, 2006.

SILVA, D.M. et al. Projeto de Software com Astah*. **Engenharia de Software Magazine**, v.30, n.3, p.25-31. 2010. Disponível em: <http://www.devmedia.com.br/revista-engenharia-de-software-30/18356>. Acesso em: 01 abr. 2012.