

FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
MANTENEDORA DO CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA–UNIVEM
BACHARELADO DE CIÊNCIA DA COMPUTAÇÃO

ROBERTO YOSHIKAZU SHIOTUKI

GERAÇÃO DE DOMÍNIO DE NEGÓCIO PARA ESTRUTURAS DE BANCO DE DADOS
BASEADOS NOS DIAGRAMAS DA UML

MARÍLIA

2007

ROBERTO YOSHIKAZU SHIOTUKI

GERAÇÃO DE DOMÍNIO DE NEGÓCIO PARA ESTRUTURAS DE BANCO DE DADOS
BASEADOS NOS DIAGRAMAS DA UML

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação, da Fundação de Ensino “Eurípides Soares da Rocha”, mantenedora do Centro Universitário Eurípides de Marília – Univem, como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador:

Prof. Dr. Edmundo Sérgio Spoto

MARÍLIA

2007

SHIOTUKI, Roberto Yoshikazu

Geração de Domínio de Negócio para estruturas de Banco de Dados Baseado nos Diagramas da UML / Roberto Yoshikazu Shiotuki;

Orientador: Prof. Edmundo Sérgio Spoto, Marília, SP: [s.n], 2007.

66 f.

Monografia (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília – Fundação de Ensino Eurípides Soares da Rocha.

1.UML 2.Banco de Dados 3. Regras de Negócios

CDD: 005.74



CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Roberto Yoshikazu Shiotuki

**GERAÇÃO DE DOMÍNIO DE NEGÓCIO PARA ESTRUTURAS DE BANCO DE DADOS
BASEADO NOS DIAGRAMAS DA UML**

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 6.0 (Seis) _____)

Orientador: Edmundo Sérgio Spoto

1º. Examinador: Ildeberto Aparecido Rodello

2º. Examinador: Maria Istela Cagnin Machado

Marília, 20 de novembro de 2007.

*Dedico este trabalho ao meu pai
Tadashi Shiotuki e a minha Mãe
Machiko Shiotuki pelo estímulo,
Paciência, amor, sacrifício e compreensão.
A todos professores e amigos que acreditaram
no meu trabalho.*

*Enfim, a todos que, direta ou indiretamente, contribuíram para conclusão de mais
esta etapa.*

AGRADECIMENTOS

Agradeço, primeiramente, a DEUS, que me possibilitou realizar este trabalho.

Em especial ao meu orientador Prof^a Edmundo Sérgio Spoto, pelo apoio, conversas e discussões no processo de elaboração desta monografia, que compartilhou parte da sua sabedoria, conduzindo o trabalho de maneira firme, porém amigo, deixando uma contribuição extremamente importante e positiva nesta fase da minha vida acadêmica.

A minha namorada Aline Gabriela Simoni que na jornada do meu trabalho sempre dando apoio moral.

Aos meus amigos e a todos que, direta ou indiretamente, colaboraram para a conclusão deste trabalho.

Roberto

Shiotuki, Roberto Yoshikazu. **Geração de Domínio de Negócio para Estruturas de Banco de Dados Baseado Nos Diagramas Da UML. 2007. 66 f.** Monografia (Graduação em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007

RESUMO

Este trabalho propõe o estudo dos conceitos relacionados a diagramas da UML e Banco de Dados para desenvolvimento de regras de negócios e Dependência funcional para a geração das estruturas de Banco de Dados Relacional, bem como um estudo de caso que utiliza a ferramenta *Enterprise Architect* (EA). O estudo de caso desenvolvido é um sistema para consultório dentário e consiste basicamente no cadastro de clientes, agenda, controle de consultas e de pagamentos.

Palavras-Chaves: UML, Banco de Dados OO, Regras de Negócios.

Shiotuki, Roberto Yoshikazu. **Geração de Domínio de Negócio para Estruturas de Banco de Dados Baseado Nos Diagramas Da UML. 2007. 66 f.** Monografia (Graduação em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007

ABSTRACT

In this work is presented the study of the concepts related diagrams from the UML and Database for development of Business Rules deal and Functional Dependence for the generation of the structures of Database Relational, as well as a case study that uses the tool *Enterprise Architect* (EA). The study of developed case it is a system for dental doctor's office and consists basically of a register cadastre it of customers, agenda, consultations control and payments.

Keywords: UML, Database OO, Business Rules.

LISTA DE FIGURAS

Figura 1 - Exemplo de Casos de Uso	20
Figura 2 - Exemplo de Diagrama de Classes.....	22
Figura 3 – Estrutura básica do Bloco PL/SQL	27
Figura 4 – Bloco PL/SQL.....	28
Figura 5 – estrutura básica de <i>procedure</i>	29
Figura 6 - Entidade e atributos	30
Figura 7 - Classificação das Assertivas de Ação.....	37
Figura 8 - Diagrama de Caso de uso.....	42
Figura 9 - Diagrama de Classe	44
Figura 10- Modelo Entidade Relacionamento.....	46
Figura 11 - Diagrama de caso de uso Odontológico	49
Figura 12 – Inserção de dados na Tabela Paciente.....	50
Figura 13 – Inserção na de dados Tabela Agenda.....	51
Figura 14 – <i>Procedure</i> MULTA.....	52
Figura 15 – seleção da Tabela Pagamento.....	52
Figura 16 – <i>Procedure</i> Execute MULTA.....	53
Figura 17 – Alterações das Tabelas PACIENTE, TRATAMENTO E PRONTUÁRIO.....	54
Figura 18 – <i>Procedure</i> NOVO	54

LISTA DE TABELAS

Tabela 1 - Elementos do diagrama de caso de uso	21
Tabela 2 - Lista de Atores do Diagrama de Caso de Uso.....	42
Tabela 3 - Regra de Negócio e classificação da tabela PACIENTE	50
Tabela 4 - Regra de Negócio e classificação da tabela AGENDA.....	51
Tabela 5 - Regra de Negócio e classificação da tabela PAGAMENTO	53
Tabela 6 - Regra de Negócio e classificação da Tabela Dentista.....	55

LISTA DE ABREVIATURAS

BDR	Banco de Dados Relacional
DDL	Linguagem de Definição de Dados - <i>Data Defenition Language</i>
DML	Linguagem de Manipulação de Dados - <i>Data Manipulation Language</i>
EA	<i>Enterprise Architect</i>
ER	Entidade Relacionamento
MER	Modelo entidade relacionamento
OMT	Técnica de Modelagem de Objetos
OO	Orientação a Objetos
OOD	Projeto Orientado a Objeto
OOSE	Engenharia de <i>Software</i> Orientada Objetos
PL/SQL	<i>Procedural Language extensions to SQL</i>
RN	Regras de Negócios
SGBD	Sistema Gerenciador de Banco de Dados
SI	Sistema de Informação
UML	Linguagem de Modelagem Unificada - <i>Unified Modeling Language</i>

SUMÁRIO

INTRODUÇÃO.....	14
Objetivos.....	14
CAPÍTULO 1 - UML (UNIFIED MODELING LANGUAGE).....	16
1.1 História da UML.....	16
1.2 Diagramas da UML	17
1.3 Diagrama de Caso de Uso	20
1.4 Diagrama de Classe	21
1.5 Considerações Finais	23
CAPÍTULO 2 – CONCEITOS E DEFINIÇÕES DE BANCO DE DADOS.....	24
2.1 Nível de Abstração	24
2.2 Linguagem de Definição de Dados (DDL).....	25
2.3 Linguagem de Manipulação de Dados (DML).....	26
2.4 PL/SQL.....	26
2.5 Stored Procedures	28
2.6 Modelo Entidade Relacionamento (MER)	29
2.7 Chaves	30
2.8 Modelo Relacional.....	31
2.9 Considerações Finais	32
CAPÍTULO 3 – REGRAS DE NEGÓCIOS	33
3.1 Classificações de Regras de Negócios	34
3.2 Categorias de Regras de Negócios	36
3.3 Considerações Finais	38
CAPÍTULO 4 – ESTUDO DE CASO.....	39
4.1 Descrição do Estudo de Caso	39
4.2 Diagrama de Casos de Uso.....	40

4.3 Diagrama de Classes.....	43
4.4 Modelo do Banco de Dados	45
4.5 - Regras de Negócios para Estruturas de Banco de Dados.....	48
4.6 Considerações Finais	55
CONCLUSÃO.....	56
TRABALHOS FUTUROS	57
REFERÊNCIAS	58
Apêndice A – Scripts das tabelas	60
Apêndice B – Inserções dos dados das tabelas.....	64
Apêndice C – Procedure para calcular multa	66

INTRODUÇÃO

Para o desenvolvimento do trabalho, em particular, serão abordados apenas os principais diagramas de modelagem da UML (Casos de Usos e Classes), o estudo enfocou a UML como sendo uma parte da metodologia responsável pela modelagem das estruturas de Banco de Dados baseados nos diagramas da UML. Ao decorrer do trabalho será utilizada a ferramenta *Enterprise Architect* para Geração de Domínio de Negócio a partir dos Diagramas da UML (Casos de Usos e Classes) e o Banco de Dados *Oracle* para criação das estruturas de Banco de Dados Relacional.

Foi feito um estudo das regras de negócios e suas classificações para aplicar regras de negócios para estruturas de Banco de Dados, para tal aplicação de regras de negócios para estruturas de Banco de Dados foram extraídas a partir dos Diagramas da UML (Casos de Usos e Classes).

Objetivos

Este trabalho tem por objetivo fazer um estudo da UML para extrair as regras de negócios e Dependência Funcional para a geração das estruturas de Banco de Dados Relacional. Será feito um estudo de caso de um sistema Odontológico, um sistema de pequeno porte voltado ao cadastro e manutenção de clientes para consultório dentário utilizando a ferramenta *Enterprise Architect* (EA), pois é uma ferramenta prática e fácil de ser manipulada. É uma ferramenta versátil, de fácil instalação, e o programa *Oracle 9i* para criação de tabelas relacionadas com o sistema odontológico.

Organização da Monografia

No Capítulo 1 é fornecida uma compreensão da UML como, História da UML, Diagramas (Use Case, Classe), são descritas as definições de cada um deles, neste trabalho, em particular, serão abordados apenas os principais diagramas de modelagem da UML (Casos de Usos e Classes) dos quais serão detalhados para o entendimento deste trabalho.

No Capítulo 2 são apresentados os conceitos de Banco de Dados como Banco de Dados Relacional, estudo dos comandos DML, DDL, PL/SQL, *STORED PROCEDURES* com finalidade de gerar estruturas de Banco de Dados Relacional OO baseados nos Diagramas da UML.

No Capítulo 3 são descritas as definições e classificações de regras de negócios.

No Capítulo 4 é abordado o estudo de caso de um sistema odontológico e suas funcionalidades como, por exemplo, cadastro de paciente, consulta paciente, agendamento. E conseqüentemente gerando os diagramas de casos de uso, diagrama de classe. São extraídas as regras de negócios para estrutura de Banco de Dados a partir dos diagramas de caso de uso e de classes.

CAPÍTULO 1 - UML (*UNIFIED MODELING LANGUAGE*)

A UML (*Unified Modeling Language*) é uma linguagem de modelagem que permite descrever um projeto de *software*. O entendimento de UML é dado pela sua concepção de métodos, processo de análise, projeto e construção de *software*, criando visões do sistema que está em construção.

É uma linguagem apoiada por um metamodelo único, que ajuda na descrição e no projeto de sistemas de *software*, particularmente daqueles construídos utilizando o estilo orientado a objetos (FOWLER *et al.*, 2005).

1.1 História da UML

De acordo com Nogueira (2007), na metade da década de 1990, Grady Booch (*Rational Software Corporation*), Ivar Jacobson (*Objectory*) e James Rumbaugh (*General Electric*) criadores de métodos orientados a objetos, começaram a desenvolver uma linguagem orientada a objeto e partiram para a criação de uma Linguagem de Modelagem Unificada (UML). Com isso esperavam fornecer ao mercado uma linguagem mais concreta e madura.

Usando técnicas orientadas a objeto criaram uma linguagem que iria desde o conceito até o sistema executável, não somente a sistemas complexos, mas também a sistemas menores e também a outros problemas que não fossem sistemas de informação, podendo ser utilizado por seres humanos e máquinas.

Segundo Larman (2000), a UML começou como um esforço conjunto de Grady Booch e Jim Rumbaugh em 1994, para combinar seus dois métodos populares, os métodos Booch e OMT (*Object Modeling Technique*). Mais tarde, Ivar Jacobson se juntou a eles (o

criador do método *OOSE*, *Object Oriented Software Engeneering*). Em resposta a uma solicitação do *OMG* (*Object Management Group*, uma entidade de padronização estabelecida pela indústria) para definir uma linguagem e notação de modelagem padronizada, UML foi submetida como candidata em 1997.

Muitas empresas ficaram interessadas, foi então criado um consórcio com várias empresas parceiras em dedicar recursos com o propósito de trabalhar com uma definição mais forte e completa da UML.

1.2 Diagramas da UML

A Linguagem UML inclui dez diagramas diferentes para representar os aspectos estruturais, comportamentais e físicos de um software (BOOCH *et al.*, 2000).

De acordo com Fowler *et al.* (2005), são descritas a seguir as definições de cada um deles como, diagrama de classes, diagrama de objetos, diagrama de casos de uso, diagrama de seqüência, diagrama de interação, diagrama de colaboração, diagrama de estados, diagrama de atividades, diagrama de componentes e diagrama de implantação, neste trabalho, em particular serão abordados apenas os principais diagramas de modelagem da UML (Casos de Usos e Classes) dos quais serão detalhados para o entendimento deste trabalho.

- **Diagrama de Classes:** Um diagrama de classes descreve os tipos de objetos presentes no sistema e os vários tipos de relacionamentos estáticos existentes entre eles. Os diagramas de classes também mostram as propriedades e as operações de uma classe e as restrições que se aplicam à maneira como os objetos estão conectados (FOWLER *et al.*, 2005).

- **Diagrama de Objetos:** Um diagrama de objetos é um instantâneo dos objetos em um sistema em um determinado ponto no tempo. Como ele mostra instâncias, em vez e

classes, um diagrama de objetos é freqüentemente chamado de diagrama de instâncias. A diferença é que o diagrama de objetos também é usado como parte dos diagramas de colaboração. Exibe um conjunto de objetos e seus relacionamentos. São diagramas que abrangem uma visão estática da estrutura ou do processo de um sistema, como ocorre nos diagramas de classes, mas sob perspectiva de casos reais ou de protótipos (FOWLER *et al.*, 2005).

- **Diagrama de Casos de Uso:** Os diagramas de caso de uso especificam o comportamento de um sistema ou de parte de um sistema e é uma descrição de um conjunto de seqüências de ações e seus relacionamentos. Abrangem a visão estática de casos de uso do sistema.

- **Diagrama de Seqüência:** Um diagrama de seqüência mostra a colaboração dinâmica entre os vários objetos de um sistema. Em um diagrama de seqüência um ator ou um objeto é mostrado como uma caixa no topo de uma linha vertical, chamada “linha da vida do objeto”. A linha de vida representa a existência do objeto ao longo do tempo. A “ativação” indica quando um objeto está executando uma ação e é representada por uma caixa retangular na linha da vida. Cada mensagem é representada por uma seta entre as linhas da vida de dois objetos. A ordem de leitura das mensagens é de cima para baixo (ELMASRI, 2005).

- **Diagrama de Interação:** Um diagrama de Interação exibe uma interação, consistindo de um conjunto de objetos e seus relacionamentos, incluindo as mensagens que podem ser trocadas entre eles. São utilizados para fazer a modelagem dos aspectos dinâmicos do sistema, envolvendo a modelagem de instâncias concretas ou protótipos de classes, interfaces e componentes.

- **Diagrama de Colaboração:** Um diagrama de colaboração mostra de maneira semelhante ao diagrama de seqüência, a colaboração dinâmica entre os objetos. Normalmente

pode-se escolher entre utilizar o diagrama de colaboração ou o diagrama de seqüência (LARMAN, 2000).

- **Diagrama de Estados:** É uma técnica conhecida para descrever o comportamento de um sistema. Existem várias formas de diagramas de estados desde os anos 60 e as mais antigas técnicas orientadas a objetos os adotaram para mostrar comportamento (FOWLER *et al.*, 2005).

Os diagramas de estados abrangem a visão dinâmica de um sistema. São importantes principalmente para a modelagem de comportamento de uma interface, classe ou colaboração.

- **Diagrama de Atividades:** Os diagramas de atividade apresentam uma visão dinâmica do sistema, modelando o fluxo de controle de uma atividade para outra. Podem ser considerados fluxogramas com estados. Uma atividade é um estado de “fazer alguma coisa”, que podem ser um processo do mundo real ou uma operação em alguma classe do banco de dados. Normalmente, os diagramas de atividades são usados para modelar o fluxo de trabalho e as operações empresariais internas para uma aplicação (ELMASRI, 2005).

- **Diagrama de Componentes:** exibe as organizações e as dependências existentes em um conjunto de componentes. Abrange a visão estática de implementação de um sistema. Está relacionado aos diagramas de classes, pois tipicamente os componentes são mapeados para uma ou mais classes, interfaces ou colaborações.

- **Diagrama de Implantação:** Representam a distribuição de componentes executáveis, bibliotecas, tabelas e arquivos pela topologia de *hardware*. Eles descrevem os recursos físicos de um sistema, inclusive nodos, componentes e conexões, e são usados para mostrar a configuração dos elementos em tempo de execução e os processos de *software* que neles residem (ELMASRI, 2005).

1.3 Diagrama de Caso de Uso

O diagrama de caso de uso é uma técnica para captar os requisitos funcionais de um sistema. Eles servem para descrever as interações típicas entre os usuários de um sistema e o próprio sistema, fornecendo uma narrativa sobre como o sistema é utilizado.

No caso de uso, os usuários são referidos como atores. Um ator é um papel que um usuário desempenha com relação ao sistema. Os atores podem ser o cliente, representante de serviço ao cliente, gerente de vendas. Os atores realizam os casos de uso e um único ator pode realizar muitos casos de uso; inversamente, um caso de uso pode ter vários atores executando-o (FOWLER *et al.*, 2005).

Portanto, um diagrama completo com todos os casos de uso e atores representa funcionalidades do sistema. Seus elementos são os próprios casos de uso, atores, sistema e interações existentes entre os dois primeiros elementos. Na Figura 1 é ilustrado um exemplo de Casos de Uso.

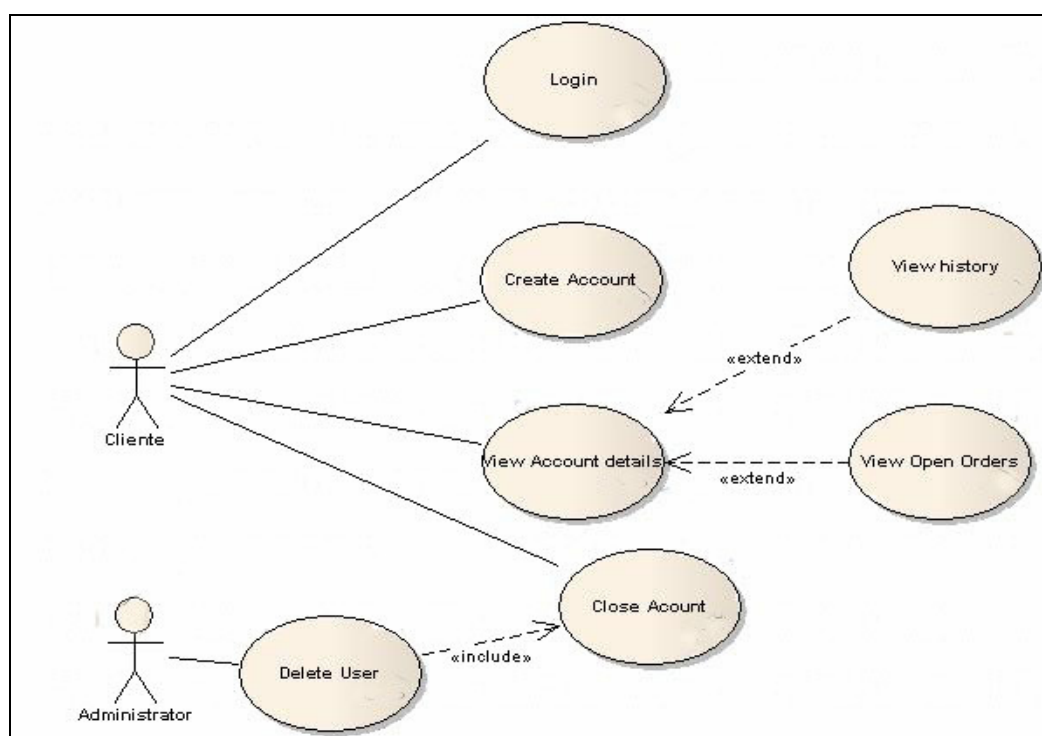


Figura 1 - Exemplo de Casos de Uso

Na Tabela 1 é descritos os conteúdos (caso de uso, ator, sistema, relacionamento) de um caso de uso como visto na Figura 1:

Tabela 1 - Elementos do diagrama de caso de uso

ELEMENTOS DO DIAGRAMA	DESCRIÇÃO
Caso de Uso	Descreve a seqüência de eventos que um ator faz no uso do sistema. São funcionalidades do sistema, ou seja, tem como objetivo decidir e fornecer uma descrição clara e consistente do que o sistema pode fazer. É iniciado por um ator e pode modelar diálogos entre atores, ou mesmo entre casos de uso.
Ator	Ator não é parte do sistema, tipicamente solicita ações do sistema e recebe reações. Cada ator pode participar de vários casos de uso, representando ações que os usuários do sistema podem desempenhar, podendo receber mensagens e interagir ativamente como sistema. Porém, pode ser um humano, máquina ou sistema.
Sistema	É o sistema a ser modelado.
Relacionamentos	Podem ser do tipo “comunicação”, “extensão” (<i>extends</i>) e “usa” (<i>uses</i>).

1.4 Diagrama de Classe

De acordo com Booch *et al.*(2000), o Diagrama de Classes é o mais encontrado em sistemas de modelagem orientados a objetos. Ele é composto por classes, interfaces e colaborações e seus relacionamentos. As classes contêm métodos e atributos e abrangem uma visão estática da estrutura do sistema. Graficamente, um diagrama de classes é uma coleção de vértices e arcos.

Com a UML, utilizam-se os diagramas de classes para visualizar os aspectos estáticos, seus relacionamentos e para especificar detalhes da construção, conforme ilustra Figura 2.

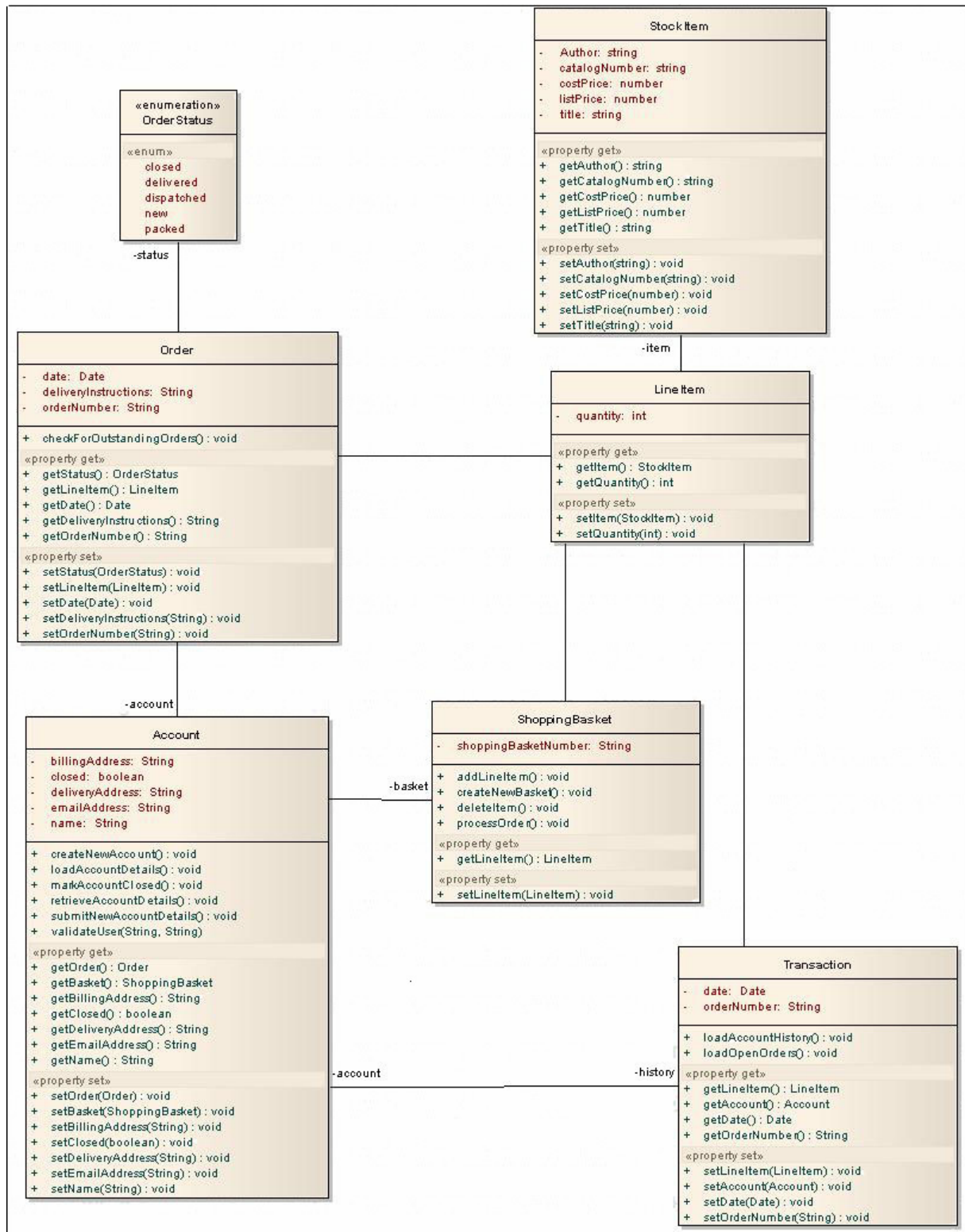


Figura 2 - Exemplo de Diagrama de Classes

1.5 Considerações Finais

Neste capítulo foram descritos todos os conceitos dos diagramas da UML em particular os diagramas de Casos de Uso e diagrama de Classes, sendo que esses foram detalhados para elaborar um sistema odontológico.

CAPÍTULO 2 – CONCEITOS E DEFINIÇÕES DE BANCO DE DADOS

Segundo Elmasri (1999), um Sistema Gerenciador de Banco de Dados (SGBD) é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados. O conjunto de dados, comumente chamado Banco de Dados, contém informações sobre uma empresa em particular. O principal objetivo de um SGBD é proporcionar um ambiente tanto conveniente quanto eficiente para a recuperação e armazenamento das informações do Banco de Dados. Sistemas de Banco de Dados são projetados para gerir grandes volumes de informações.

Um sistema de Banco de Dados proporciona dois tipos de linguagens: uma específica para os esquemas do Banco de Dados e outra para expressar consultas e atualizações.

O objetivo principal de um sistema de Banco de Dados é proporcionar aos usuários uma visão abstrata dos dados. Isto é, o sistema esconde determinados detalhes de como os dados são mantidos e como estão armazenados. Isto é por meio da definição de três níveis de abstração de dados, nível físico, nível lógico e nível de visão, como apresentado a seguir.

2.1 Nível de Abstração

Abstração de dados é a definição de um tipo de dado por seu comportamento e estado, utilizando-se métodos. A manipulação desse dado é realizada apenas através de seus métodos.

Segundo Silberschats *et al.* (1999), para que se possa usar um sistema, ele precisa ser eficiente na recuperação das informações. Esta eficiência está relacionada à forma pela qual foram projetadas as complexas estruturas de representação desses dados no Banco de Dados. Já que muitos dos usuários dos sistemas de Banco de Dados não possuem formação em

computação, foram desenvolvidos sistemas que omitem essa complexidade, por meio dos diversos níveis de abstração, de modo a facilitar a interação dos usuários com o sistema. São definidos três níveis de abstração.

- Nível físico: é o nível mais baixo de abstração e descreve como os dados estão realmente armazenados;

- Nível lógico: este nível descreve quais dados estão armazenados no Banco de Dados e quais os relacionamentos entre eles. Assim, o Banco de Dados como um todo é descrito em termos de um número relativamente pequeno de estrutura simples.

- Nível de visão: é o mais alto nível de abstração descreve apenas parte do Banco de Dados. O nível de abstração das visões de dados é definido para simplificar esta interação com o sistema, que pode fornecer muitas visões para o mesmo Banco de Dados.

2.2 Linguagem de Definição de Dados (DDL)

De acordo com Silberschats *et al.* (1999), um esquema de dados é especificado por um conjunto de definições expressas por uma linguagem especial chamada linguagem de definição de dados (*Data-Defenition Language – DDL*). O resultado da compilação dos parâmetros DDLs é armazenado em um conjunto de tabelas que constituem um arquivo especial chamado dicionário de dados ou diretório de dados.

Um dicionário de dados é um arquivo de metadados, isto é, dados a respeito de dados. Em um sistema de Banco de Dados, esse arquivo ou diretório é consultado antes que o dado real seja modificado.

Exemplos de comandos DDLs:

- *CREATE TABLE* – criação de tabela;
- *DROP TABLE* – exclusão de tabela ;

- *ALTER TABLE* – alteração de tabela;
- *CREATE INDEX* – criação de índice .

2.3 Linguagem de Manipulação de Dados (DML)

As instruções admissíveis de DML (*data manipulation language*) são *SELECT*, *INSERT*, *UPDATE*, *DELETE*. Cada um desses comandos opera uma função: *SELECT* retorna linhas de uma tabela de Banco de Dados que correspondem ao critério dados pela cláusula *WHERE*; *INSERT* adiciona linhas a uma tabela de Banco de Dados; *UPDATE* modifica as linhas em uma tabela de Banco de Dados que correspondem à cláusula *WHERE*; e *DELETE* remove as linhas identificadas pela cláusula *WHERE* (URMAN, 2002).

Exemplos de comandos DMLs:

- *INSERT* – inserção de registros;
- *UPDATE* – atualização de registros;
- *DELETE* – exclusão de registros;
- *SELECT* – consulta tabela;
- *TRANSACTION* - transações.

2.4 PL/SQL

PL/SQL é muito similar à linguagem SQL, mas acrescenta construções de programação similares a outras linguagens que permite a utilização integrada no código dos comandos da DML.

De acordo com Ramalho (1997), a linguagem PL/SQL trabalha com conceito de bloco estruturado. Além de criações de *procedures* e funções, pode-se criar um bloco de comandos sem nome específico e executá-lo. As *procedures* e funções permitem que parâmetros sejam passados para ela, enquanto um bloco não tem essa capacidade nem tão pouco retorna qualquer valor como uma função.

As vantagens de se usar PL/SQL são vários, como redução do número de chamadas de uma aplicação para o servidor, portabilidade entre as plataformas em que o Oracle é executado e detectar gerenciamento de erros.

Um bloco PL/SQL em uma estrutura básica, como seção de Declaração, em que todos os objetos são declarados, seção de Execução, em que os comandos PL/SQL são colocados e a seção de Exceção, em que os erros são tratados. Na Figura 3 é ilustrada uma estrutura básica do bloco PL/SQL:

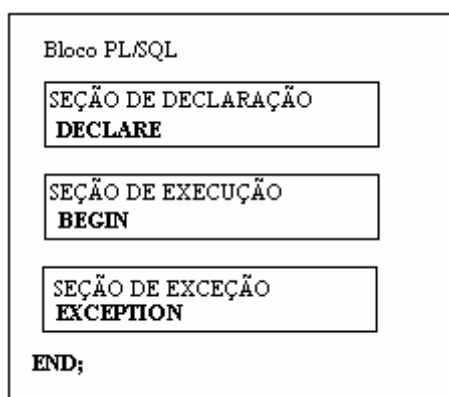


Figura 3 – Estrutura básica do Bloco PL/SQL

A seção Declaração é opcional, entretanto se o bloco usar variáveis ou constantes, todas deve ser previamente declarado antes de serem referenciadas em um comando para iniciar essa seção, deve ser usada a palavra chave declare. Na Figura 4 é ilustrado um exemplo básico de PL/SQL (RAMALHO,1997).

```
DECLARE i INT := 1;
BEGIN
  FOR i IN 1..10 LOOP
    INSERT INTO TABLE(field1,field2)
    VALUES(I,SYSDATE)
  END LOOP;
END;
```

Figura 4 – Bloco PL/SQL

A seção de Execução é iniciada com a declaração *BEGIN*. Os comandos especificados nessa seção devem seguir as mesmas regras usadas para a execução de um comando no *SQL*PLUS*. Devem obrigatoriamente terminar com um “;”. Essa seção pode conter comandos SQL, comandos de atribuição, comandos de controle lógico como “*if...then*”, “*loop*”, “*for..loop*”, “*while*”, “*while...loop*” e, entre outros.

Seção de Exceção serve para tratar um erro que eventualmente ocorra durante a execução de um programa PL/SQL. Quando um erro ocorre dentro de um programa PL/SQL, é apresentada uma mensagem de erro.

2.5 *Stored Procedures*

De acordo com Urman (2002), uma *stored procedure* é uma instrução PL/SQL por si própria. Ela não é chamada como parte de uma expressão. Quando uma *stored procedure* é chamada, o controle passa para a primeira instrução executável dentro da *stored procedure*. Quando a *stored procedure* termina, o controle é reassumido na instrução seguinte a chamada de *procedure*. A esse respeito, *stored procedures* PL/SQL comportam-se da mesma maneira que as *procedures* em outras linguagens de programação. Diferentemente de um *trigger*, que é executado automaticamente, uma *procedure* precisa ser chamada a partir de um programa ou manualmente pelo usuário. O uso de *stored procedure* possui grandes vantagens como

reusabilidade de código que pode ser usada por diversos usuários em diversas ocasiões, como um script *SQL*Plus*, dentro de um *trigger* ou aplicação.

Uma *procedure* possui duas partes. Uma seção de especificação e o corpo da *procedure*. Na Figura 5 é ilustrado um exemplo de uma *procedure* (RAMALHO, 1997).

```
CREATE [OR REPLACE] PROCEDURE nome
(lista de parâmetros) IS
BEGIN
    Comandos
END [nome]
```

Figura 5 – estrutura básica de *procedure*

2.6 Modelo Entidade Relacionamento (MER)

Segundo Silberschats *et al.* (1999) o modelo entidade relacionamento (E-R) tem por base a percepção de que o mundo real é formado por um conjunto de objetos chamados entidades e pelo conjunto dos relacionamentos entre esses objetos. Ela pode ser um objeto de existência física, uma pessoa, carro, ou pode ser um objeto de existência conceitual, uma companhia, um trabalho. Desta forma os conceitos de MER foram criados para serem mais compreensíveis pelos usuários. É usado para facilitar principalmente no desenvolvimento de projeto de Banco de Dados. Uma entidade é representada por um conjunto de atributos que são propriedades descritivas de cada membro de um conjunto de entidades. Por exemplo, a entidade cliente pode ter atributos definidos como código, nome, endereço, cidade, estado, bairro, etc. O valor dos atributos que descrevem cada entidade é quem ocupa a maior parte dos dados armazenados na base de Banco de Dados. Na Figura 6 é ilustrado um exemplo de entidade e seus atributos.

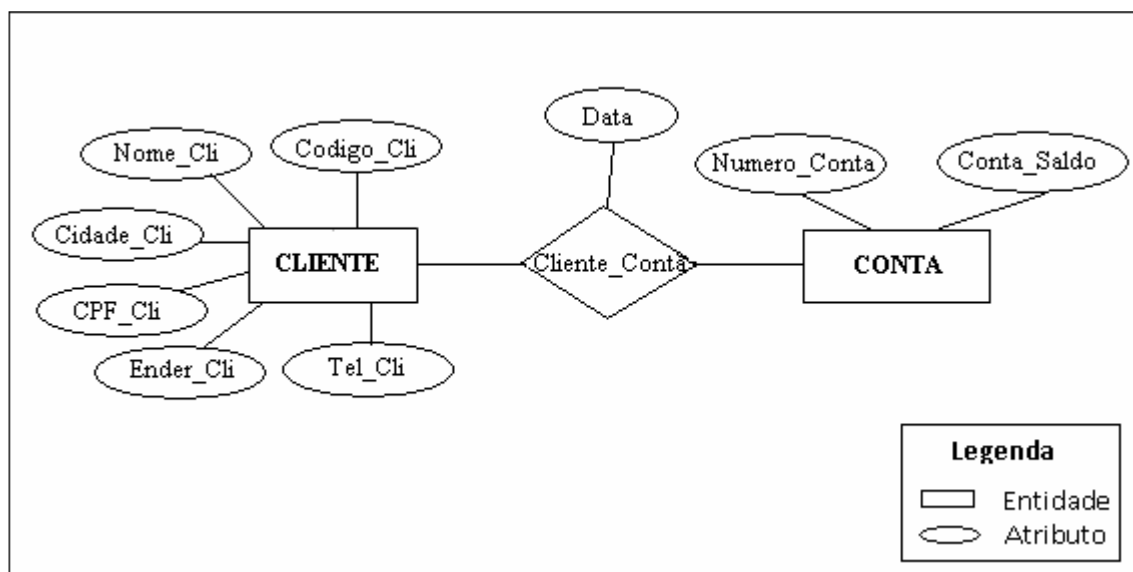


Figura 6 - Entidade e atributos

Um relacionamento é uma associação entre uma ou várias entidades. Por exemplo, Maria possui uma conta, constitui um relacionamento entre entidade Cliente e a Entidade Conta. Entre o relacionamento das entidades há restrições importantes, denominadas cardinalidades como, “uma para um”, “uma para muitos”, “muitos para um”, “muitos para muitos” que são formas de representar a frequência com que existe o relacionamento.

2.7 Chaves

É importante especificar como as entidades dentro de um dado conjunto de entidades e os relacionamentos dentro de um conjunto de relacionamentos podem ser identificados. Conceitualmente, entidades e relacionamentos individuais são distintos, entretanto, na perspectiva do banco de dados, a diferença entre ambos deve ser estabelecida em termos de seus atributos, o conceito de *chave* permite-nos fazer tais distinções. São descritas a seguir as chaves primária, secundária e estrangeira.

Segundo Date (2000), chave primária é o identificador único de um registro em um arquivo que pode ser constituído de um campo ou a combinação de campos, de tal maneira

que não existam dois registros no arquivo com o mesmo valor da chave primária. A chave primária fornece ao arquivo: consistência aos processos de inclusão, alteração e exclusão de dados e preenchimento que deve ser obrigatória, ou seja, seu valor deve ser atômico.

Chave secundária é formada por um ou mais campos e é utilizada como parâmetro de filtro para a seleção de registros no arquivo de consulta. Uma característica é que a chave secundária não precisa ser única.

Chave estrangeira é uma coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de outra tabela em relacionamento, ou na chave primária da própria tabela quando for um auto-relacionamento. A chave estrangeira é o mecanismo que permite a implementação de relacionamento em um banco de dados relacional.

Existem algumas ocasiões em que mais de uma coluna ou combinações de colunas podem servir para distinguir uma linha das demais. Uma das colunas (ou combinações de colunas) é escolhida como chave primária. As demais colunas ou combinações são chamadas chaves alternativas.

2.8 Modelo Relacional

O modelo relacional usa um conjunto de tabelas para representar tanto os dados quanto a relação entre eles. Cada tabela possui múltiplas colunas e cada uma possui um nome único.

De acordo com Date (2000), em uma tabela identificam-se cada uma de suas linhas de uma chave única, ou chave primária. Essa chave pode ser simples, formada por apenas uma coluna, ou composta, formada por duas ou mais colunas. Os Bancos de Dados possuem mecanismos para garantir que a unicidade da chave primária seja preservada.

2.9 Considerações Finais

Finalizando este capítulo, definem-se os conceitos de Banco de Dados como uma coleção de dados. Um típico banco de dados representa alguns aspectos da vida do mundo real. Foram abordados os principais tipos de comandos para manipular e criar tabelas como as DDL e DML. Em seguida apresentou-se a estrutura de um bloco PL/SQL e que as instruções de DML dentro da PL/SQL em particular evitam inconsistências de dados. A *stored procedures* que se comportam de maneira parecida com as *procedures* de outras linguagens de programação. Definiu os conceitos básicos de entidades e atributos do modelo entidade relacionamento.

CAPÍTULO 3 – REGRAS DE NEGÓCIOS

Na seção 3.1 deste capítulo serão abordadas as regras de negócios com a finalidade de modelá-las as regras de negócios nos Diagramas da UML (Caso de Uso e Classes).

Segundo Bajec e Crisper (2001), Regras de negócios são compostos por termos e fatos e regras (propriamente ditas). Há vários tipos de regras de negócios e várias classificações voltadas para diversos propósitos e audiências, alguns dos quais se voltam mais para os negócios, outros dos quais se voltam mais para os Sistemas de Informação.

A primeira aparência da frase “regras de negócios” foi no ano de 1984, quando Daniel Appelton escreveu um artigo chamado *Bussines Rule: The Missing Link*. Appelton discutiu problemas que eram causados por uma carência na padronização dos termos de negócios. Ele aponta que os analistas da área não utilizam termos de negócios padronizados para as soluções de aplicações comuns a todas as empresas, existindo, então, vários termos para uma mesma aplicação que varia de empresa para empresa (BAJEC e KRISPER, 2001).

Logo a publicação desse artigo, vários profissionais da área de Sistema de Informação notaram que as regras de negócios poderiam ser captadas em processos de máquina, assim, podem cumprir as regras e assegurar que os processos dos negócios sejam conduzidos e controlados de acordo com os padrões, procedimentos e regras das empresas. Posteriormente, os pesquisadores de Sistema de Informação e os profissionais da área começaram a ter novas visões para trabalhar com tais regras.

De acordo com Bajec e Krisper (2001), várias pesquisas foram realizadas com as ações que podem ser realizadas nos Banco de Dados usando componentes ativos, como *stored procedures*. O desempenho do Banco de Dados e as funções são utilizados como instrumentos para reforçar as regras de negócios.

Atualmente, a área promissora para a aplicação das regras de negócios é a pesquisa de integração entre as definições de negócios da empresas com o suporte dos Sistemas de Informação. Está cada vez mais óbvio que um Sistema de Informação pode atuar de maneira a gerar atributos competitivos para as organizações se este estiver de acordo com as necessidades dos negócios, tornando os trabalhos mais fáceis. Geralmente, as políticas, os procedimentos das empresas e as regras estão em constante mudança, os quais ocorrem espontaneamente devido às novas reações de mercado e dos resultados das estratégias, além das decisões tomadas pela empresa. As mudanças nem sempre estão adaptadas ao Sistema de Informação (BAJEC e KRISPER, 2001).

Os softwares devem suportar as mudanças que ocorrem nas empresas, pois requerem novas implementações, de novas regras, gerando uma necessidade de reprogramação para atingir as novas metas, objetivos e políticas da empresas.

No processo de desenvolvimento de regra de negócio, os procedimentos e funcionamentos do sistema deverão ser organizadas para que seja executada. Para que uma regra de negócio seja executada, como por exemplo, calcular na folha de pagamento de funcionário o salário total sob horas extras mensais, é necessário primeiramente que se execute a regra “Calcular o valor total de horas”, e para que seja executada, é necessária a quantidade de horas e o valor por hora.

3.1 Classificações de Regras de Negócios

Abaixo são descritas as classificações de regras de negócios, mas para finalidade deste trabalho serão usados como exemplo “Regras de Rejeição” e “Projeção”, pois essas classificações afetam diretamente com os eventos do Banco de Dados.

De acordo com Ross (ROSS, 1998, apud ALENQUER, 2002) as regras de negócios são classificadas em:

- **Regras de Rejeição** (*Rejector*): são regras que rejeitam um evento porque ele conduz o sistema a um estado que não é permitido. As restrições de integridade em bancos de dados são exemplos desse tipo de regra.

Exemplos:

“Uma conta não pode ter saldo negativo”;

“Um novato não pode participar do clube honorário”;

“O número de assentos de um curso não pode exceder 30”;

“Um funcionário aposentado não pode participar do conselho de funcionários”.

“Um time não pode ter mais que 11 jogadores em campo”.

- **Regras de Projeção** (*Projector*): são regras que projetam o evento em outros eventos, ou seja, geram novas alterações no estado do sistema ou executam ações. Assim sendo, em sistemas de informação, essas regras alteram os fluxos de execução.

Exemplos:

“Ao criar um cliente, deve ser criada uma nova conta para ele”.

“Se uma conta tiver saldo maior que 5.000, enviar folheto sobre investimentos”.

- **Regras de Produção** (*Productor*): são regras que não respondem a eventos, elas apenas definem informações do negócio.

Exemplos:

“Saldo é igual a crédito menos débito”;

“Uma conta é “Ouro Especial” se seu saldo atual é superior a 30.000 e seu médio saldo médio é superior a 20.000.”;

“O custo de um produto é igual à soma dos custos de seus componentes”;

“O salário de um vendedor é igual ao salário fixo mais o que ele conseguiu vender multiplicado pelo percentual de comissão”.

3.2 Categorias de Regras de Negócios

Outras regras de negócios foram definidas por HAY e HEALY (2000) que são divididas em três grandes categorias: Assertivas de Estrutura, Assertivas de Ação e Derivações.

- **Assertivas de Estrutura** (*Structural Assertion*): formam o vocabulário e a estrutura básica a partir das quais as regras de negócios são expressas. As Assertivas de Estrutura formam a ontologia do negócio.

Assertivas de Estrutura são classificadas em Termos (*Term*) e Fatos (*Fact*). Termos são os elementos que compõem o vocabulário do negócio. Os Termos podem ser Comuns (*Common Term*) ou de Negócio (*BusinessTerm*). Um Termo Comum diz respeito àqueles cujo significado é de conhecimento geral, independente do domínio em questão. Por exemplo, "supermercado", "preço" e "embalagem" são exemplos de Termos Comuns.

O Termo de Negócio tem que ser definido a partir de um ou mais Fatos. Por exemplo, o Termo de Negócio “venda” é definido a partir do Fato “um cliente pode levar um artigo de uma loja ao pagar o preço desse artigo”.

Fatos são Assertivas de Estrutura que relacionam dois ou mais Termos. É importante notar que um Fato pode relacionar não apenas Termos particulares, mas também Termos genéricos. O exemplo anterior (“um cliente pode levar um artigo de uma loja ao pagar o preço deste artigo”) diz respeito a qualquer cliente, qualquer artigo e qualquer loja; e não uma instância em particular, embora um Fato também possa descrever situações particulares.

- **Assertivas de Ação** (*Action Assertion*): são regras que restringem o alcance das estruturas gerais definidas pelas Assertivas de Estrutura. Tratam do estabelecimento de limites, exceções e regulamentos para as estruturas.

Uma Assertiva de Ação é composta por um objeto âncora e um ou mais objetos correspondentes. Em uma regra da forma “Se ... Então ...”, o objeto âncora seria a parte logo após o “Se”, e os objetos correspondentes ficariam logo após o “Então”.

Assertivas de Ação podem utilizar modificadores para obterem o efeito restritivo da maneira adequada. Modificadores são expressões que alteram o significado de frases; por exemplo, "tem que ser", "pode ser", "pode ser apenas", entre outras.

O objeto âncora pode ser de qualquer tipo de regra de negócio (Assertiva de Estrutura, Assertiva de Ação ou Derivação). Em contrapartida, toda Assertiva de Ação tem que ser uma propriedade da regra de negócio associada.

Segundo Hay e Healy (2000), assertivas de Ação são classificadas em Condição (*Condition*), Restrição de Integridade (*Integrity Constraint*) e Autorização (*Authorization*), sendo estas mais relacionadas com ações do Banco de Dados, pois as tabelas possuem Integridades e Condição, pois para certos eventos no banco há uma condição. Na figura 7 é ilustrado as subcategorias da Assertiva de Ação, Restrição de Integridade, Condição e Autorização.

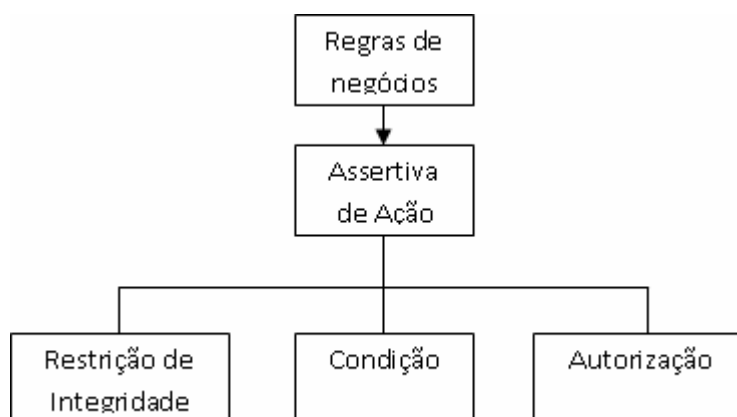


Figura 7 - Classificação das Assertivas de Ação

Uma Condição é uma assertiva que deve ser testada para saber se alguma outra regra deve ser aplicada. Por exemplo: “o preço caiu mais de 20%”, “acabou o artigo no estoque”, “houve aumento de imposto”.

Uma restrição de integridade é uma assertiva que sempre representa uma verdade. Enquanto uma Condição pode ou não ser verdadeira, uma Restrição de Integridade sempre o é, não importa a situação. Por exemplo, “todo artigo tem que ter um preço”, “o volume de vendas é sempre um número positivo ou nulo”.

Uma Autorização define uma prerrogativa ou privilégio a respeito de um ou mais Construtos. As Autorizações seguem a forma: “Apenas x pode fazer y”, onde geralmente x é um usuário e y é uma ação a ser executada. Por exemplo: “apenas o gerente do supermercado pode aceitar compras a prazo”.

Além destas classificações, as assertivas de ação possuem uma tipificação, de modo que este conjunto possa crescer conforme a necessidade de quem modela as regras. O modelo define alguns tipos úteis: *Enabler*, *Timer* e *Executive*.

Para entender melhor no próximo capítulo será apresentado diagrama de caso de uso que abordará alguns tipos de regras de negócios diretamente relacionadas com as ações que envolvem os dados armazenados (BD).

3.3 Considerações Finais

Concluindo este capítulo, foram abordados os conceitos de regras de negócios, estas regras podem ser aplicadas em quaisquer tipos de empresas de grande e pequeno porte, assim como também em sistemas de pequeno e grande porte. Foram estudadas suas classificações e categorias que serão usadas no Banco de Dados da Seção 4.5.

CAPÍTULO 4 – ESTUDO DE CASO

O estudo de caso apresentado neste capítulo exercitou os conceitos de OO, UML, Banco de Dados Relacional e análise de documentos de requisitos funcionais para apoiar o desenvolvimento de diagramas de Casos de Uso, Classe. Existem várias ferramentas para a modelagem dos diagramas da UML, como a *Rational Rose*, *SmartDraw*, *Enterprise Architect* e entre outras. A ferramenta de apoio para o desenvolvimento do estudo de caso escolhido foi o *software Enterprise Architect 7.0* para a modelagem dos diagramas UML (Use case, Classes), pois é uma ferramenta prática e fácil de ser manipulada.

O *software Enterprise Architect* é uma ferramenta versátil, de fácil instalação, é uma ferramenta comercial, que possui as seguintes edições, *Corporate Edition*, *Professional Edition*, *Desktop Edition* e TRIAL. Para a realização deste trabalho foi utilizada a versão TRIAL, disponível por trinta dias.

Nesta Seção serão abordados os estudos de casos e diagramas de classes do consultório odontológico.

4.1 Descrição do Estudo de Caso

Apresenta-se um estudo de caso de um consultório odontológico. No consultório “Odontologia Nikkey” trabalha um apenas Dentista e que funciona nos seguintes dias e horários: de segunda à sexta das 08:30h às 18:00h. No caso desse consultório odontológico os horários de atendimento ao paciente funcionam de meia em meia hora, portanto o primeiro horário disponível é 08:30h e o próximo é às 09:00h e assim sucessivamente até as 18:00h.

Quando um paciente deseja marcar uma consulta pela primeira vez, primeiramente é verificada a agenda do dentista e oferecido o primeiro horário disponível (data e hora), a hora

para ser marcado de acordo com o que o paciente deseja. Se o paciente concordar com o horário, a secretária registra na agenda o nome do paciente e horário combinado.

Na próxima consulta, antes de iniciar o tratamento, o profissional preenche a ficha clínica do prontuário e o tratamento é iniciado. O prontuário possui uma folha em que são anotados os procedimentos realizados. Após cada consulta, o procedimento realizado é anotado juntamente com a data e o dente em que o trabalho foi realizado. Quando um tratamento é terminado em um dente, o mesmo é anotado no odontograma final do prontuário odontológico. O pagamento é recebido dentro de períodos pré-estabelecidos pelo dentista.

4.2 Diagrama de Casos de Uso

Seguindo a visão geral do estudo de casos do sistema odontológico, foi elaborado um diagrama de casos de uso utilizando a Ferramenta *Enterprise Architect* para estruturar algumas tabelas (Agenda, Paciente, Dentista, Pagamento) do Banco de Dados, na Seção 4.5 são ilustradas em detalhe as estruturas de Banco de Dados.

A partir dos diagramas de casos de uso foi elaborada uma estrutura de Banco de Dados. Para tal extração de estrutura de Banco de Dados foi elaborado da seguinte forma:

No Caso de uso “Cadastrar Paciente”, o paciente fornece os dados para secretária realizar o cadastramento como nome, endereço, sexo, RG, CPF, etc. Ao realizar o cadastro do paciente é necessário que campos da tabela Paciente (CodPaciente, NomePaciente, DataNascPaciente e SexoPaciente) sejam todos preenchidos, caso contrário não é permitido a inserção dos dados na tabela Paciente, então criou a necessidade de criar uma estrutura de Banco de Dados para tal situação. Na Seção 4.5 será ilustrado em detalhe.

No Caso de Uso “Cadastrar” e “Controlar Agenda”, a secretária consulta na agenda se há horários (data e hora) disponíveis para próximas consultas, e se há vagas os dados são cadastrados e armazenados na tabela Agenda, caso contrário procura outro horário disponível.

Para essa situação descrita, foi criada uma estrutura para a tabela Agenda usando o comando “*Select*” que procura se na Tabela pelo campo DataConsulta e HoraConsulta se há alguma hora e data marcado.

No Caso de Uso “Controlar Finanças”, sua funcionalidade é controlar o valor do tratamento de cada paciente, caso a data de vencimento pré-determinado vencer, é cobrado uma multa, a partir dessa situação foi criado uma estrutura de Banco de Dados na tabela Pagamento usando *stored procedures* para realizar o cálculo de multa.

No Caso de Uso “Cadastrar Dentista”, é disparado pelo ator “Secretária” que realiza o cadastro do dentista. Caso algum dia se o dentista precisar sair da clínica ele é substituído por outro, mas o histórico dos pacientes que fazem tratamentos com esse dentista permanecerá relacionado, sendo assim, se algum dia alguém precisar da relação dos históricos entre o dentista e o paciente esses dados estarão armazenados no Banco de Dados.

As situações descritas no Caso de Uso acima na Seção 4.5 serão mostradas em detalhes nas estruturas de Banco de Dados a partir desses diagramas de Casos de Uso.

O modelo de Caso de uso descrito neste trabalho é ilustrado na Figura 8. Na Tabela 2 são descritos os atores “Secretária”, “Dentista” e “Financeiro” dos Casos de Uso.

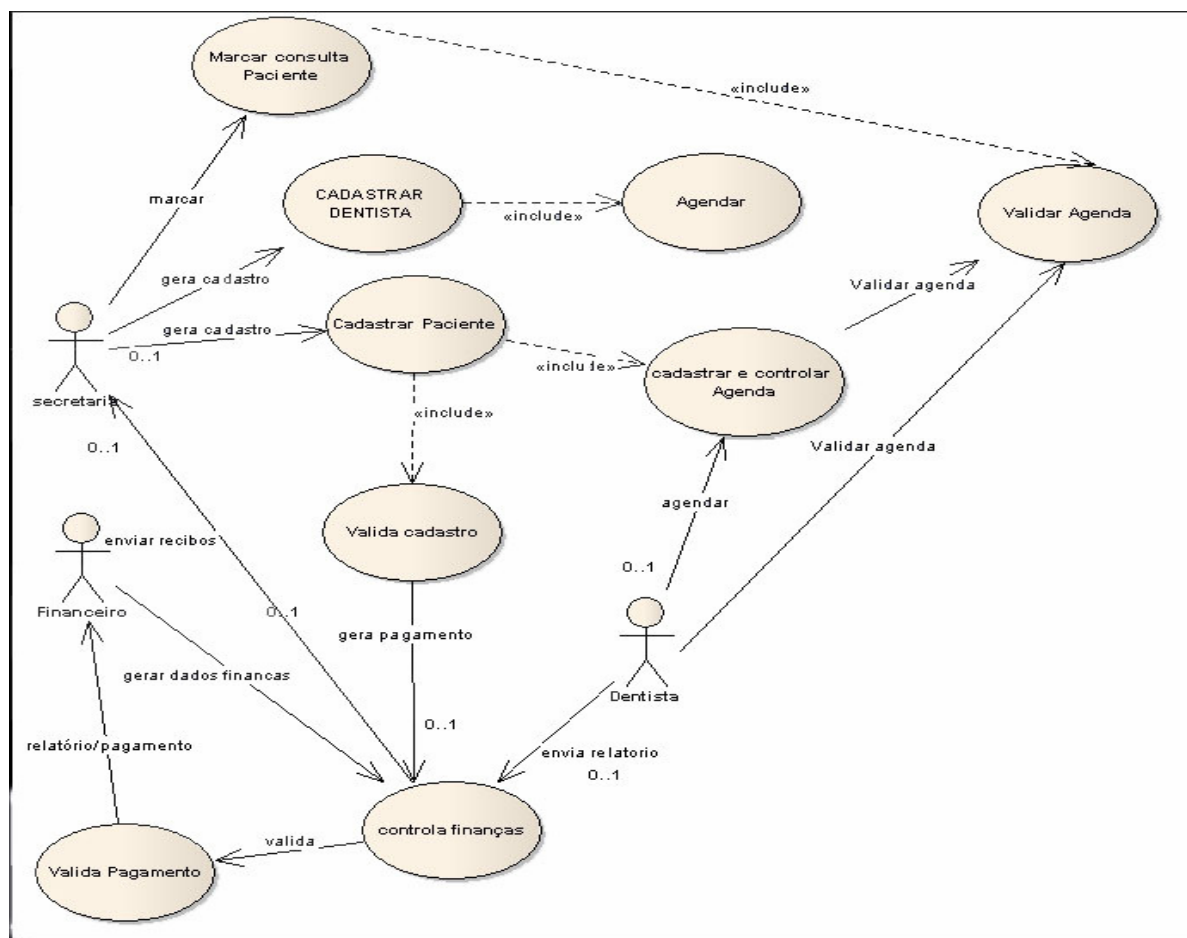





Figura 8 - Diagrama de Caso de uso

Tabela 2 - Lista de Atores do Diagrama de Caso de Uso

Atores	Descrições
 Secretária	Ao chegar ao consultório, o paciente fornece dados para a secretária para a realização do cadastro, agendamento, próximas consultas.
 Dentista	Após registrar os dados do paciente, inicia-se uma consulta ou um tratamento.
 Financeiro	Responsável pelo controle de pagamentos, o pagamento pode ser a prazo ou a vista, o mesmo pode ser após o final do tratamento ou no início de acordo com o entendimento entre o paciente e o dentista.

4.3 Diagrama de Classes

Na Figura 9 possuem nove diagramas de classes do cenário sistema odontológico:

Paciente: Na classe Paciente serão controladas as informações do Paciente como (nome, RG, Endereço, CPF e outras informações descritas no diagrama).

Secretária: A classe Secretária é responsável em representar as outras classes da clínica. Ela será formada pelos atributos nome, código e endereço e seu comportamento será representado pelos métodos cadastrar secretária, Marcarconsulta e outras que serão acionadas através de mensagens para outras classes do sistema.

Agenda: Responsável pelo agendamento de horários dos pacientes;

Prontuário: Tem como função de controlar as informações referentes ao prontuário de cada paciente;

Dentista: É responsável pelos dados do dentista, bem como dos tratamentos de cada paciente;

Tratamento: O tratamento é iniciado pela classe Agenda e Dentista;

Procedimento: São descritos os procedimentos e valor para cada tipo de tratamento;

Tratamento_Procedimento: A classe Tratamento se relaciona com a classe Procedimento, e a classe intermediária Tratamento_Procedimento possui os atributos Numero_Dente e Numero_Face que durante o tratamento é selecionado qual dente esta sendo tratado;

Pagamento: A classe pagamento é acionada quando o tratamento é terminado contendo os atributos, a data de pagamento e data de vencimento.

Na Figura 9 é apresentado o diagrama de classe que compõe o cenário do sistema odontológico:

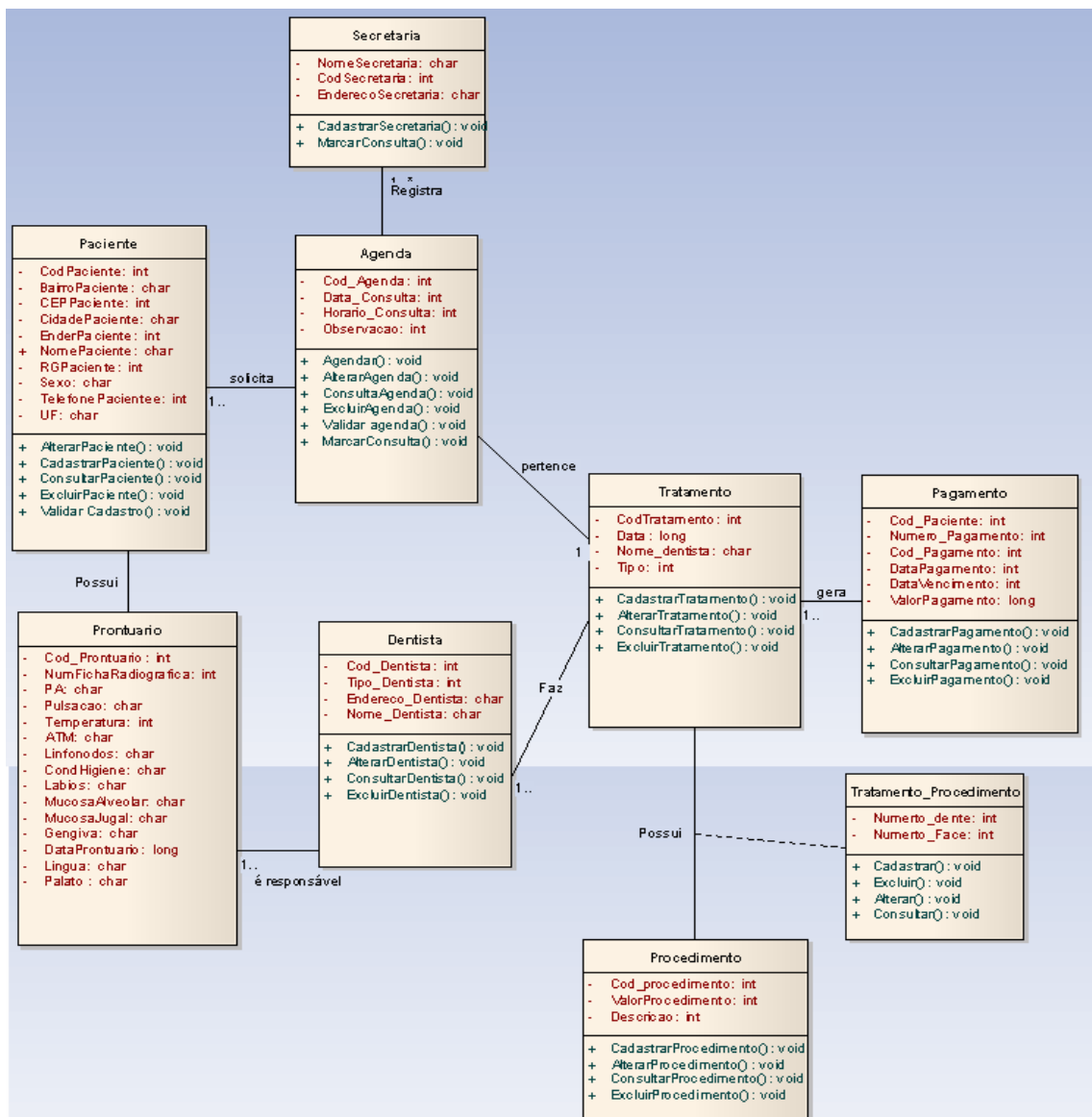


Figura 9 - Diagrama de Classe

As classes apresentadas na Figura 9 apresentam as associações existentes entre elas e seus respectivos atributos e métodos. A partir deste exemplo serão apresentadas a seguir as definições de escolha das regras de negócios relacionadas com ações que envolvem os dados persistentes do sistema (armazenados no BD).

4.4 Modelo do Banco de Dados

A partir dos diagramas da UML (Casos de Uso e Classes), embora seus conceitos estejam baseados em técnicas orientados a objetos, os modelos resultantes de estrutura e comportamentos podem ser usados para projetar Banco de Dados Relacionais, já que o diagrama de classe é semelhante ao modelo de entidade relacionamento. A estrutura dos diagramas de classe fornece uma especificação estrutural do esquema de Banco de Dados, mostrando nome, os atributos e as operações de cada classe. Sua função geral é descrever a coleção de objetos de dados e seus relacionamentos, o que é compatível com a meta do projeto conceitual de Banco de Dados.

Na Figura 10 são mostradas as entidades e seus relacionamentos, uma tabela de modelo entidade relacionamento que foi modelado usando a ferramenta *COMPUTER ASSOCIATES ERWIN* 4.0. Apresenta a mesma estrutura de tabela e dados do diagrama de classes apresentadas na Figura 9.

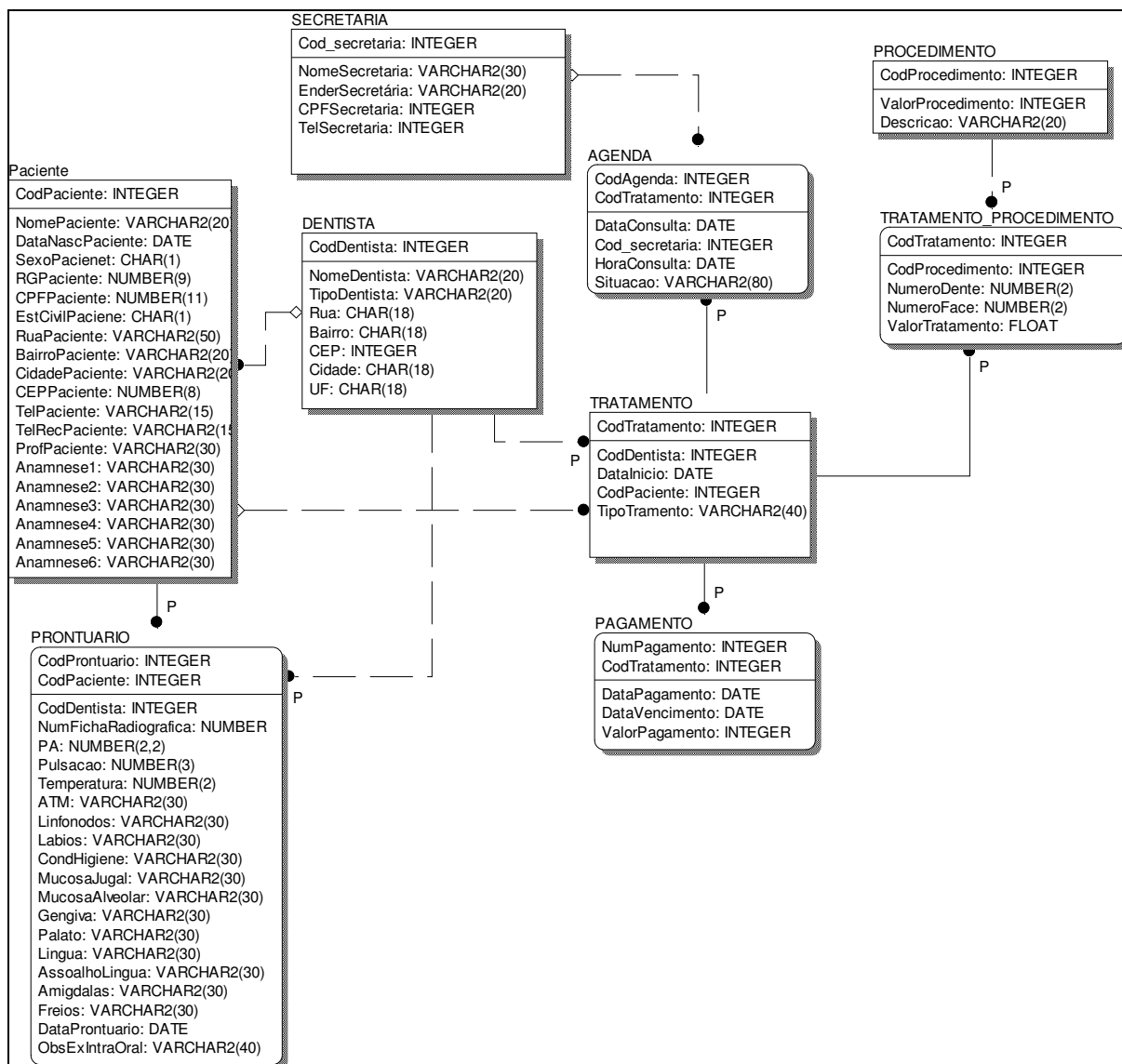


Figura 10- Modelo Entidade Relacionamento

Na Figura 10 tem-se nove tabelas, nas quais serão aplicadas algumas regras de negócios no banco de dados de algumas tabelas como PACIENTE, PAGAMENTO, AGENDA e DENTISTA.

A tabela DENTISTA armazena os dados do dentista, como nome, tipo de dentista, endereço.

Na Tabela SECRETÁRIA são armazenados os dados da secretária, como nome, endereço, cpf, telefone.

Na Tabela PACIENTE são armazenados os dados do paciente, que é identificado por um CodPaciente que é um número único. Além dos dados pessoais, armazena também os dados da Anamnese, assim garantindo assim uma possível consulta posteriormente.

Na Tabela PRONTUÁRIO estão armazenados os dados do prontuário, tais como, exame geral, extra-oral e intra-oral obtidos durante a consulta, é identificado por um conjunto de dois atributos, CodPaciente e CodProntuario formando uma chave primária única.

Na Tabela TRATAMENTO tem como função de armazenar os dados da primeira consulta do cliente e do início do tratamento, é identificado pelos atributos CodTratamento e CodDentista.

Na Tabela PAGAMENTO são armazenados todos os dados do pagamento, tais como, data do pagamento, data do vencimento e valor do pagamento, é identificado por um conjunto de dois atributos CodTratamento e NumPagamento formando uma chave primária única.

Na Tabela TRATAMENTO_PROCEDIMENTO estão armazenados os dados do tipos de procedimentos que o tratamento recebe, tais como, número do dente, número da face e o valor do tratamento, é identificado por uma chave CodTratamento.

Na Tabela PROCEDIMENTO são armazenados os dados dos procedimentos, tais como, o tipo do procedimento e o valor, é identificado por um atributo único denominado, CodProcedimento, que é chave primária.

Na Tabela AGENDA, tem como função de armazenar todos os possíveis dados do tratamento, como datas, tipo e evolução do tratamento, é identificado por um conjunto de dois atributos, CodTratamento e CodAgenda formando uma chave primária única.

4.5 - Regras de Negócios para Estruturas de Banco de Dados

Nesta Seção aplicam-se algumas regras de negócios no Banco de Dados, visando mostrar que existem regras que são específicas das Ações de Negócio destinadas ao Banco de Dados do Sistema. Para tanto, os conceitos dos diagramas da UML (Caso de Uso e Classes), Banco de Dados e regras de negócios vistos anteriormente foram importantes para o desenvolvimento do trabalho.

As possíveis regras de negócios e sua classificação sobre o Banco de Dados foram extraídas dos diagramas de casos de usos vistos na Seção 4.2. Na Figura 11 são ilustrados os casos de usos dos atores “Secretária”, “Dentista” e “Financeiro”.

A seguir são descritos os casos de usos envolvidos no sistema Odontológico os quais foram ilustrados na Figura 10:

- **Caso de Uso: Cadastrar Paciente** – O paciente ao chegar ao consultório, fornece os dados (nome, RG, endereço, telefone, etc.) para a secretária realizar o cadastro. Os dados do paciente serão armazenados na tabela PACIENTE, caso o paciente já seja cadastrado não é necessário realizar o cadastro do mesmo.

Após realizar o cadastro, é obrigatório realizar um agendamento de retorno do paciente ao consultório.

Para que sejam cadastrados com sucesso os registros do paciente na tabela PACIENTE, os seguintes campos CODPACIENTE, NOMEPACIENTE, DATANASCPACIENTE, SEXOPACIENTE, são campos obrigatórios inserí-los para realizar os cadastros dos mesmos sendo que os demais campos não são obrigatórios inserí-los. Caso na inserção se uns dos campos obrigatórios não forem inseridos no banco é retornada uma mensagem de erro.

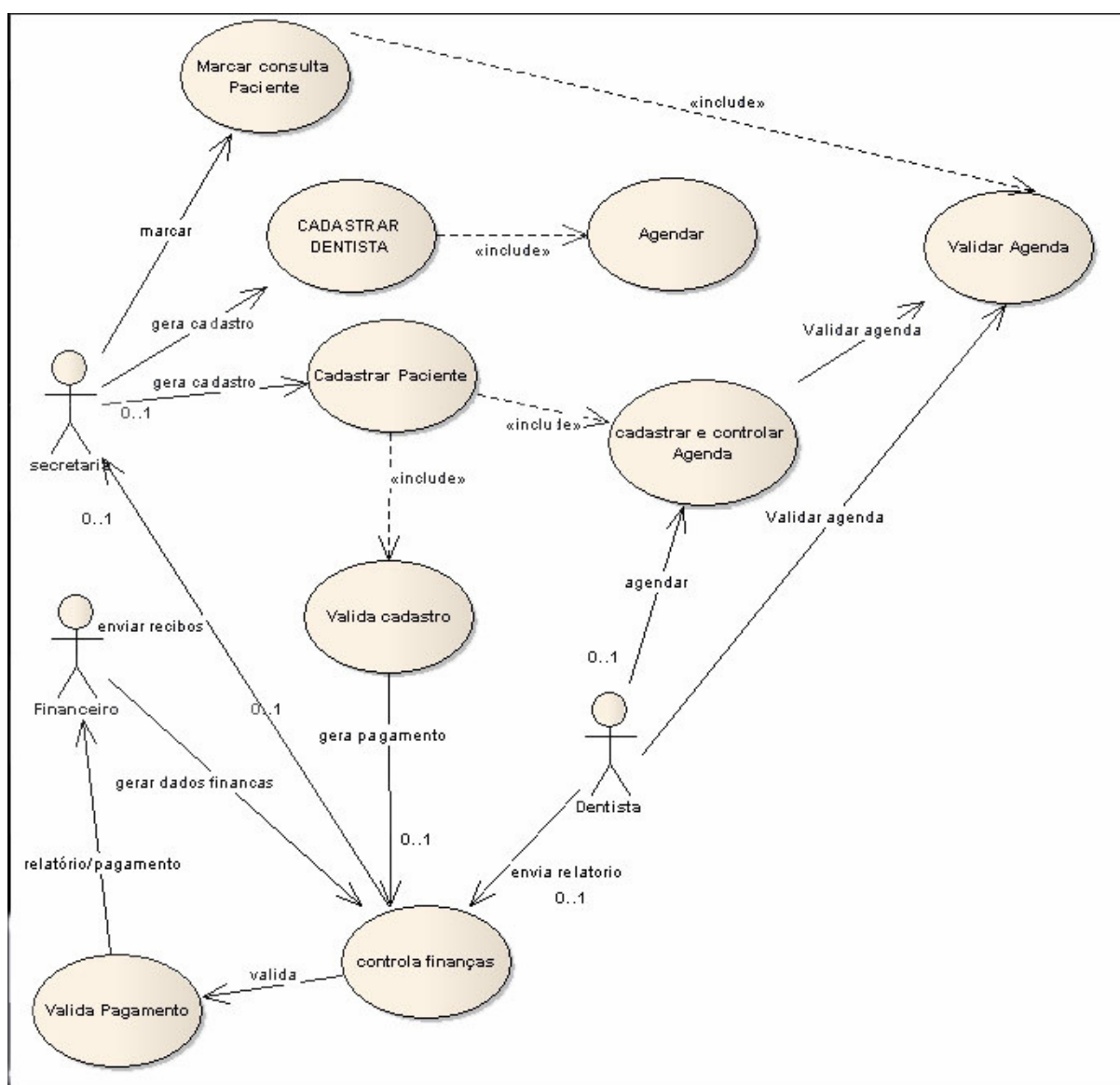


Figura 11 - Diagrama de caso de uso Odontológico

A seguir são descritos os procedimentos de como realizar um cadastro do paciente.

Para que o cadastro seja realizado com sucesso, antes foram aplicadas regras para que possa ser executado o cadastro do mesmo. Os valores da tabela Paciente como SEXOPACIENTE, DATANASCPACIENTE e SEXOPACIENTE são obrigatórios preenche-los, caso contrário não é possível realizar o cadastro.

Na situação descrita se classifica como “Rejeição”, pois não é possível realizar um cadastro dos dados do paciente se os valores obrigatórios não forem de preenchimento. Na Figura 12 é apresentada a inserção de dados na tabela PACIENTE.

```
SQL> INSERT INTO PACIENTE VALUES(2,'ALINE','26-11-1981',NULL,350997822,22438819855,'S','RUA TAL','BAIRRO TAL','MARILIA',17516290,34141959,34141957,'ESTUDANTE','NAO','NAO','NAO','NAO','SIM','SIM','SIM','NAO','SIM','NAO','NAO','NAO','SIM','NAO','SIM','NAO','NAO',1);
INSERT INTO PACIENTE VALUES(2,'ALINE','26-11-1981',NULL,350997822,22438819855,'S','RUA TAL','BAIRRO
*
ERRO na linha 1:
ORA-01400: não é possível inserir NULL em ("SCOTT"."PACIENTE"."SEXOPACIENTE")

SQL>
```

Figura 12 – Inserção de dados na Tabela Paciente

Após a execução do comando “insert”, observa-se que foi retornada uma mensagem de erro: “não é possível inserir NULL em (“SCOTT” . “PACIENTE”.”SEXOPACIENTE”)”. Isso ocorre devido ao campo SEXOPACIENTE ser definido como “NOT NULL”. Na Tabela 3 são descritas as Regra de Negócio e sua Classificação no momento da inserção da tabela PACIENTE.

Tabela 3 - Regra de Negócio e classificação da tabela PACIENTE

Regras de negócios	Classificação	Categoria da Assertiva
Para realizar um cadastro é necessário que os campos obrigatórios sejam inseridos, caso contrário é impossível continuar armazenar dados na tabela	Rejeição: Na inserção dos registros não é permitido campos NULLOS: no caso o campo SEXOPACIENTE.	Condição: se campo algum campo estiver vazio no caso SEXOPACIENTE, retorna uma mensagem de Erro.

Os registros dos pacientes podem ficar armazenados temporariamente ou permanentemente, caso o mesmo paciente por algum tempo volte ao consultório novamente.

- **Caso de Uso: Cadastrar e Controlar Agenda** – Na Seção 4.2 foi descritos tal situação para esse Caso de Uso. No Caso de Uso será descrito um exemplo de um paciente que deseja marcar um agendamento no dia 09/11/07 e hora 15:30h, primeiramente a secretária verificará se há horário disponível, se há horário disponível é realizado o agendamento, caso contrário não é agendado.

Para a situação descritas na página anterior será ilustrada de como a estrutura de Banco de Dados na Tabela Agenda tratará tal situação. A seguir é executado o comando “select” na tabela AGENDA para verificar se naquele momento existia data e hora que coincidem com os horários em que o paciente deseja marcar. Na Figura 13 é ilustrado um exemplo do comando “select”.

```

1 SELECT DATACONSULTA, HORACONSULTA FROM AGENDA
2* WHERE DATACONSULTA='09/11/07' AND HORACONSULTA='15:30'
SQL> /

DATACONS HORACONSUL
-----
09/11/07 15:30

```

Figura 13 – Inserção na de dados Tabela Agenda

Observa-se na Figura 13 que a data e hora em que o paciente desejou marcar não é possível de ser agendado, pois já o horário (data e hora) está alocado para um outro paciente, nesse caso é necessário procurar um outro horário disponível para realizar o agendamento.

Nessa situação aplicam regras de negócios “Condição” da categoria Assertiva de Ação que são relacionadas diretamente com Banco de Dados.

Na Tabela 4 são descritas as Regra de Negócio e sua Classificação no momento do agendamento da tabela AGENDA.

Tabela 4 - Regra de Negócio e classificação da tabela AGENDA

Regras de negócios	Classificação	Categoria da Assertiva
Para realizar o agendamento, primeiramente deve verificar se na tabela agenda há horários livres, se houver, é marcado agendamento, caso contrário procura outro horário disponível na tabela	Rejeição: Não é possível realizar um agendamento, pois há data e hora marcada para pelo outro paciente.	Condição: Se houver data e hora disponível é marcada a consulta, senão procura outro horário disponível

- **Caso de Uso: Controlar Finanças** – Tem a função de controlar o recebimento do pagamento do pacientes. O pagamento é realizado dentro da data de vencimento previsto.

Após o paciente realizar o pagamento para a secretária é emitido um comprovante de pagamento em duas vias, o boleto possui a data do pagamento, data do vencimento, valor e multa, caso se a data de vencimento estiver vencida.

A tabela PAGAMENTO armazena data de pagamento, data de vencimento, valor do pagamento e multa.

Analisam-se as situações descritas acima que interferem no banco e aplicam-se as regras de negócios referentes à tabela PAGAMENTO. Se a data de vencimento estiver vencido é cobrado uma multa.

A seguir foram criados *procedures* para que seja calculado o valor da multa caso a data de vencimento estiver vencido. Na Figura 14 é apresentada a *procedure* que calcula a multa.

```
SQL> CREATE OR REPLACE PROCEDURE MULTA(UDATAVENC IN DATE)
 2 IS BEGIN
 3 DECLARE UPORCENTAGEM NUMBER(05,2);
 4 BEGIN
 5 UPORCENTAGEM:=2/100;
 6 IF (SYSDATE>UDATAVENC) THEN
 7 UPDATE PAGAMENTO SET VALORPAGAMENTO = VALORPAGAMENTO + (VALORPAGAMENTO * UPORCENTAGEM)
 8 WHERE (DATAVENCIMENTO = UDATAVENC);
 9 END IF;
10 END;
11 END;
12 /
Procedimento criado.
```

Figura 14 – Procedure MULTA

Observa-se na Figura 15 é ilustrada uma execução do comando de seleção para visualizar os dados da tabela PAGAMENTO, na coluna VALORPAGAMENTO possui os valores 100 e 150 reais que é o valor do tratamento.

```
SQL> select * from pagamento;
NUMPAGAMENTO CODTRATAMENTO DATAPAGA DATAVENC VALORPAGAMENTO
-----
          1             1          29/11/07          100
          2             1          09/11/07          150
```

Figura 15 – seleção da Tabela Pagamento

Observa-se na Figura 16 é ilustrada uma multa referente à data vencida, no caso da coluna DATAVENCIMENTO que contém os valores 09/11/07 passou a ter uma multa de 150

a 153 reais, na Figura 16 é ilustrada a execução da *procedure* MULTA para o calculo da multa.

```

SQL> execute multa('09/11/07');
Procedimento PL/SQL concluído com sucesso.
SQL> select * from pagamento;
NUMPAGAMENTO CODTRATAMENTO DATAVENC VALORPAGAMENTO
-----
1 1 29/11/07 100
2 1 09/11/07 153

```

Figura 16 – Procedure Execute MULTA

No momento da execução da *procedure* MULTA a categoria de Assertiva de Condição está ligada diretamente ao banco, pois pode se interpretar da seguinte maneira Assertiva de Condição: se a data atual for maior que a data de vencimento, então é cobrado uma multa de 2%. Na Tabela 5 são ilustradas as regras de negócios e suas classificações:

Tabela 5 - Regra de Negócio e classificação da tabela PAGAMENTO

Regras de negócios	Classificação	Categoria da Assertiva
Após o cliente realizar o pagamento é emitido um comprovante de pagamento, caso a data de vencimento seja igual ou maior a data atual, é aplicado uma multa ao paciente	Projeção: se a data de vencimento for maior ou igual à data atual, aplicar uma multa de 5%	Condição: se a data de vencimento for maior ou igual a data atual é aplicado uma multa

- **Caso de Uso: Cadastrar Dentista** – Tem a função de cadastrar, alterar, excluir os dados do dentista. Toda a transação realizada é diretamente ligada ao Banco de Dados, nesse caso A tabela Dentista.

A tabela DENTISTA está relacionada com outras tabelas como PACIENTE, TRATAMENTO e PRONTUARIO. Para realizar uma transação nas tabelas PACIENTE, TRATAMENTO e PRONTUARIO, primeiramente é necessário que o dentista esteja cadastrados na tabela DENTISTA.

Se por algum motivo o dentista precisar sair do consultório, então é substituído o antigo dentista pelo novo dentista, mas os dados dos paciente continuam relacionadas com o antigo dentista, pois se algum dia precisar o histórico entre o dentista e o paciente os dados entre paciente e o antigo dentista estão armazenados no Banco. Então todos os pacientes que faziam tratamento com o antigo dentista neste momento passam a ser tratados pelo novo dentista.

Para essa situação, foi criada uma *procedure* para que os dados dos pacientes, tratamento e prontuário sejam atualizados de acordo com o novo dentista, o valor na inserção de um novo dentista é por passagem de parâmetros. A figura 17 ilustra a *procedure* criada.

```

SQL> CREATE OR REPLACE PROCEDURE novo(CODNOVO IN INTEGER)
 2  IS BEGIN
 3  BEGIN
 4  INSERT INTO DENTISTA VALUES
 5  (CODNOVO,'ROBERTO','PROTESE','SAO LUIZ','CENTRO',1751030,'MARILIA','SP');
 6
 7  INSERT INTO PACIENTE VALUES
 8  (1,'ALINE','26-11-1981','F',350997822,22438819855,'S','RUA TAL',
 9  'BAIRRO TAL','MARILIA',17516290,34141959,34141957,'ESTUDANTE','NAO','NAO','NAO','NAO','SIM',
10  'SIM','SIM','NAO','SIM','NAO','NAO','NAO','SIM','NAO','SIM','NAO','NAO',CODNOVO);
11
12  INSERT INTO TRATAMENTO VALUES
13  (1,CODNOVO,'05-08-07',1,'CANAL');
14
15  INSERT INTO PRONTUARIO
16  VALUES(1,1,CODNOVO,1,12,34,1,'NORMAL','NORMAL','NORMAL','NORMAL','NORMAL','NORMAL','NORMAL',
17  'NORMAL','NORMAL','NORMAL','NORMAL','NORMAL','15-11-07','NORMAL');
18
19  END;
20  END;
21  /
Procedimento criado.

```

Figura 17 – Alterações das Tabelas PACIENTE, TRATAMENTO E PRONTUÁRIO.

A seguir é executado a *procedure* “*procedure novo*” para o cadastramento de um novo dentista no caso com o “CodDentista = 2”. Conforme ilustrado na Figura 18.

```

SQL> EXECUTE NOVO(2);

Procedimento PL/SQL concluído com sucesso.

SQL> |

```

Figura 18 – *Procedure* NOVO

Com a execução desse *procedimento* todos os pacientes, prontuário e tratamento foram atualizado de acordo com o novo dentista. Na Tabela 6 é ilustrada as regras de negócio e suas classificações:

Tabela 6 - Regra de Negócio e classificação da Tabela Dentista.

Regras de negócios	Classificação	Categoria da Assertiva
O antigo dentista foi substituído pelo novo dentista, mas deve-se manter os dados do antigo dentista relacionadas com as demais tabelas (Paciente, Tratamento e Prontuário)	Rejeição: Não é permitido excluir o dentista, pois há integridade referencial entre as tabelas Paciente, Prontuário e Tratamento.	Restrição de Integridade: Há chaves CodDentista nas tabelas Paciente, Prontuário e Tratamento

4.6 Considerações Finais

Para elaboração do trabalho foram estudados os diagramas da UML, os conceitos de Banco de Dados e regras de negócios com a finalidade de extrair regras de negócios baseados nos diagramas da UML (Casos de Uso e Classes).

As regras de negócios obtidos a partir dos diagramas da UML para estrutura de Banco de Dados podem ser aplicadas várias regras de negócios de acordo com a necessidade da empresa, cada empresa adota suas regras para determinado sistema ou projeto. Podem-se aplicar regras tanto nos aplicativos quanto nos Banco de Dados.

Pelos diagramas de Use Case facilitou a elaboração de regras de negócios para estrutura de Banco de Dados analisando alguns diagramas de use case (Cadastrar Paciente, Cadastrar e Controlar Agenda, Controlar Finanças e Cadastrar Dentista).

CONCLUSÃO

O desenvolvimento “Geração de Domínio de Negócio para Estruturas de Banco de Dados Baseado nos diagramas da UML”, utilizou-se a Ferramenta Enterprise Architect, estudo da UML e Banco de Dados *Oracle 9i*, assim como na maioria das pequenas e grandes empresas utilizam ferramentas semelhantes como a *Rational Rose*, *Case Studio* para a implementação de um sistema.

Os procedimentos para o desenvolvimento do trabalho foram realizados da seguinte forma: estudo do comportamento da clínica odontológica como seu funcionamento interno (horário de serviço, os procedimentos de atendimento ao cliente, data de pagamento estipulado pelo dentista e outras fatores de funcionamento que foram importantes para o desenvolvimento do trabalho); estudos e Conceitos de Banco de Dados; estudos das regras de negócios, classificações e categorias; estudo dos diagramas da UML (Casos de Uso e Classes) que foram elaborados e analisados os comportamentos dos Casos de Uso (Cadastrar Paciente, Cadastrar e Controlar Agenda, Controlar Finanças e Cadastrar Dentista). De acordo com o comportamento e métodos dos Casos de Uso foi elaborada a estrutura de Banco de Dados de para tratar os eventos que se relacionam diretamente ao Banco de Dados. Todos estes fatores foram importantes para a realização do trabalho.

O uso dos diagramas da UML (Caso de Uso e Classe) facilitou o trabalho em extrair regras de negócios para estruturação de Banco de Dados aplicando-se as regras e suas classificações.

Para qualquer desenvolvimento na aplicação de um sistema, modelagem de banco de dados, diagramas da UML e entre outras, há uma infinidade de regras de negócios e classificações que podem ser aplicadas sobre os mesmos.

TRABALHOS FUTUROS

Para trabalhos futuros pode-se explorar o máximo das regras de negócios de todos os diagramas de Casos de Uso. Pelos comportamentos de Casos de Uso elaborar um Banco de Dados consistentes e atômicos usando regras de negócios.

Pode-se implementar uma interface gráfica da clínica odontológica relacionando com o Banco de Dados deixando que as estruturas de Banco de Dados tratem tais eventos ocorridos pelo usuário do sistema.

REFERÊNCIAS

BAJEC, M.; KRISPER, M. **Managing business rules in enterprises**. *University of Ljubljana, Faculty of Computer and Information Science*, 2001.

BOOCH, G.; JACOBSON, I.; RUMBAUGH, J. **UML Guia do Usuário**. Rio de Janeiro: Editora Campus, 2000. 473 p.

Date, C.J. **Introdução a sistema de Banco de Dados**. Rio de Janeiro: Editora Campus, 2000.

ELMASRI, R.; NAVATHE, S. B. **Sistema de Banco de Dados**. São Paulo: Editora Person Addison Wesley, 2005.

FOWLER, M.; KOBRYN, C.; BOOCK, G.; JACOBSON, I.; RUMBARUGH, J. **UML ESSENCIAL “Um breve guia para a linguagem de modelagem de objetos” (3ª EDIÇÃO)**. Porto Alegre: Editora BOOKMAN, 2005.

HAY, David; HEALY, Kery Anderson. **Defining Business Rules - What Are They Really?** [S.l.: S.n.], 2000.

LARMAN, C. **Utilizando UML e Padrões “Uma introdução à Análise e ao Projeto Orientados a Objetos”**. Porto Alegre: Editora Bookman, 2000.

LOPES ALENQUER, Pablo. **Regras de negócios para Análise em Ambientes OLAP**. Dissertação submetida ao corpo docente do Instituto de Matemática e Núcleo de Computação Eletrônica (IM/NCE) – Universidade Federal do Rio de Janeiro – UFRJ, 2002.

NOGUEIRA, A. **Histórico da UML**. Disponível em: http://www.imasters.com.br/artigo/2994/uml/historico_da_uml. Acesso em 01 março, 2007.

RAMALHO, A. **Oracle Personal Oracle 7.3 & Power Objects 2.0**. São Paulo: Editora Makron Books, 1997.

SILBERSCHATZ, A.; KORTH, H., F.; SURDARSHAN, S. **Sistema de Banco de Dados**. São Paulo: Editora Makron Books LTDA, 1999.

THE BUSSINESS RULE GROUP. **Guide Business Rules Project: Final Report**, revision 1.3, july 2000. Disponível em: <http://www.businessrulesgroup.org>. Acesso em 09 agosto 2007.

UML. **Unified Modeling Language**. Disponível em: http://www.apostilando.com/download_final.php?cod=2038&autenticado=nao. Acesso em 05 março, 2007.

URMAN,S. **Oracle 9i Programação PL/SQL**. Rio de Janeiro: Editora Campus, 2002.

Apêndice A – Scripts das tabelas

```

CREATE TABLE AGENDA (
  CodAgenda      INTEGER NOT NULL,
  CodTratamento INTEGER NOT NULL,
  DataConsulta  DATE NOT NULL,
  Cod_secretaria INTEGER NULL,
  HoraConsulta  DATE NOT NULL,
  Situacao      VARCHAR2(80) NULL
);

ALTER TABLE AGENDA
  ADD ( PRIMARY KEY (CodAgenda, CodTratamento) );

```

```

CREATE TABLE DENTISTA (
  CodDentista      INTEGER NOT NULL,
  NomeDentista     VARCHAR2(20) NULL,
  TipoDentista     VARCHAR2(20) NULL,
  Rua              CHAR(18) NULL,
  Bairro           CHAR(18) NULL,
  CEP              INTEGER NULL,
  Cidade           CHAR(18) NULL,
  UF               CHAR(18) NULL
);

```

```

ALTER TABLE DENTISTA
  ADD ( PRIMARY KEY (CodDentista) );

```

```

CREATE TABLE Paciente (
  CodPaciente      INTEGER NOT NULL,
  NomePaciente     VARCHAR2(20) NOT NULL,
  DataNascPaciente DATE NOT NULL,
  SexoPacient      CHAR(1) NOT NULL,
  RGPaciente       NUMBER(9) NULL,
  CPFPaciente      NUMBER(11) NULL,
  EstCivilPaciene  CHAR(1) NULL,
  RuaPaciente      VARCHAR2(50) NULL,
  BairroPaciente   VARCHAR2(20) NULL,
  CidadePaciente   VARCHAR2(20) NULL,
  CEPPaciente      NUMBER(8) NULL,
  TelPaciente      VARCHAR2(15) NULL,
  TelRecPaciente   VARCHAR2(15) NULL,
  ProfPaciente     VARCHAR2(30) NULL,
  Anamnese1        VARCHAR2(30) NULL,
  Anamnese2        VARCHAR2(30) NULL,
  Anamnese3        VARCHAR2(30) NULL,
  Anamnese4        VARCHAR2(30) NULL,
  Anamnese5        VARCHAR2(30) NULL,
  Anamnese6        VARCHAR2(30) NULL,
  Anamnese7        VARCHAR2(30) NULL,
  Anamnese8        VARCHAR2(30) NULL,
  Anamnese9        VARCHAR2(30) NULL,
  Anamnese10       VARCHAR2(30) NULL,
  Anamnese11       VARCHAR2(30) NULL,

```

```

Anamnese12    VARCHAR2(30) NULL,
Anamnese13    VARCHAR2(30) NULL,
Anamnese14    VARCHAR2(30) NULL,
Anamnese15    VARCHAR2(30) NULL,
Anamnese16    VARCHAR2(30) NULL,
Anamnese17    VARCHAR2(30) NULL,
CodDentista   INTEGER NULL
);

```

```

ALTER TABLE Paciente
ADD ( PRIMARY KEY (CodPaciente) );

```

```

CREATE TABLE PAGAMENTO (
NumPagamento    INTEGER NOT NULL,
CodTratamento   INTEGER NOT NULL,
DataPagamento   DATE NULL,
DataVencimento   DATE NULL,
ValorPagamento  INTEGER NULL
);

```

```

ALTER TABLE PAGAMENTO
ADD ( PRIMARY KEY (NumPagamento, CodTratamento) );

```

```

CREATE TABLE PROCEDIMENTO (
CodProcedimento  INTEGER NOT NULL,
ValorProcedimento INTEGER NULL,
Descricao        VARCHAR2(20) NULL
);

```

```

ALTER TABLE PROCEDIMENTO
ADD ( PRIMARY KEY (CodProcedimento) );

```

```

CREATE TABLE PRONTUARIO (
CodProntuario    INTEGER NOT NULL,
CodPaciente      INTEGER NOT NULL,
CodDentista      INTEGER NOT NULL,
NumFichaRadiografica NUMBER NULL,
PA               NUMBER(5) NULL,
Pulsacao         NUMBER(3) NULL,
Temperatura      NUMBER(2) NULL,
ATM              VARCHAR2(30) NULL,
Linfonodos       VARCHAR2(30) NULL,
Labios           VARCHAR2(30) NULL,
CondHigiene      VARCHAR2(30) NULL,
MucosaJugal      VARCHAR2(30) NULL,
MucosaAlveolar   VARCHAR2(30) NULL,
Gengiva          VARCHAR2(30) NULL,
Palato           VARCHAR2(30) NULL,
Lingua           VARCHAR2(30) NULL,
AssoalhoLingua   VARCHAR2(30) NULL,
Amigdalas        VARCHAR2(30) NULL,
Freios           VARCHAR2(30) NULL,
DataProntuario   DATE NULL,
ObsExIntraOral   VARCHAR2(40) NULL
);

```

);

ALTER TABLE PRONTUARIO
ADD (PRIMARY KEY (CodProntuario, CodPaciente));

CREATE TABLE SECRETARIA (
Cod_secretaria INTEGER NOT NULL,
NomeSecretaria VARCHAR2(30) NULL,
EnderSecretaria VARCHAR2(20) NULL,
CPFSecretaria INTEGER NULL,
TelSecretaria INTEGER NULL
);

ALTER TABLE SECRETARIA
ADD (PRIMARY KEY (Cod_secretaria));

CREATE TABLE TRATAMENTO (
CodTratamento INTEGER NOT NULL,
CodDentista INTEGER NOT NULL,
DataInicio DATE NULL,
CodPaciente INTEGER NULL,
TipoTratamento VARCHAR2(40) NULL
);

ALTER TABLE TRATAMENTO
ADD (PRIMARY KEY (CodTratamento));

CREATE TABLE TRATAMENTO_PROCEDIMENTO (
CodTratamento INTEGER NOT NULL,
CodProcedimento INTEGER NOT NULL,
NumeroDente NUMBER(2) NULL,
NumeroFace NUMBER(2) NULL,
ValorTratamento FLOAT NULL
);

ALTER TABLE TRATAMENTO_PROCEDIMENTO
ADD (PRIMARY KEY (CodTratamento));

ALTER TABLE AGENDA
ADD (FOREIGN KEY (Cod_secretaria)
REFERENCES SECRETARIA);

ALTER TABLE AGENDA
ADD (FOREIGN KEY (CodTratamento)
REFERENCES TRATAMENTO);

ALTER TABLE Paciente
ADD (FOREIGN KEY (CodDentista)
REFERENCES DENTISTA);

```
ALTER TABLE PAGAMENTO
  ADD ( FOREIGN KEY (CodTratamento)
        REFERENCES TRATAMENTO );
```

```
ALTER TABLE PRONTUARIO
  ADD ( FOREIGN KEY (CodDentista)
        REFERENCES DENTISTA );
```

```
ALTER TABLE PRONTUARIO
  ADD ( FOREIGN KEY (CodPaciente)
        REFERENCES Paciente );
```

```
ALTER TABLE TRATAMENTO
  ADD ( FOREIGN KEY (CodPaciente)
        REFERENCES Paciente );
```

```
ALTER TABLE TRATAMENTO
  ADD ( FOREIGN KEY (CodDentista)
        REFERENCES DENTISTA );
```

```
ALTER TABLE TRATAMENTO_PROCEDIMENTO
  ADD ( FOREIGN KEY (CodTratamento)
        REFERENCES TRATAMENTO );
```

```
ALTER TABLE TRATAMENTO_PROCEDIMENTO
  ADD ( FOREIGN KEY (CodProcedimento)
        REFERENCES PROCEDIMENTO );
```

Apêndice B – Inserções dos dados das tabelas

```
INSERT INTO AGENDA
(CODAGENDA,CODTRATAMENTO,DATACONSULTA,COD_SECRETARIA,HORACONSULTA,
SITUACAO)
VALUES ( 1,1,'08/11/07',1,'15:00','MARCADO');
```

```
INSERT INTO AGENDA
(CODAGENDA, CODTRATAMENTO,DATACONSULTA,COD_SECRETARIA,HORACONSULTA,
SITUACAO)
VALUES ( 2,1,'9/11/07',1,'15:30','MARCADO');
```

```
INSERT INTO DENTISTA
(CODDENTISTA,NOMEDENTISTA,TIPODENTISTA,RUA,BAIRRO,CEP,CIDADE,UF)
VALUES(1,'ANDERSON','CIRURGIA','SÃO LUIZ','CENTRO',1751030,'MARILIA','SP');
```

```
INSERT INTO PACIENTE VALUES (1,'ROBERTO','28-11-1981', 'M', 350997822,22438819855,'S',
'RUA','BAIRRO','MARILIA',17516290, 34141959,34141957,'ESTUDANTE','NAO',
'NAO','NAO','NAO','SIM','SIM','SIM','NAO','SIM','NAO','NAO','NAO','SIM','NAO','SIM','NAO','NAO',
'NAO',1)
```

```
INSERT INTO PACIENTE VALUES (2,'ALINE','26-11-1981', 'F', 350997822,22438819855,'S','RUA
TAL','BAIRRO TAL','MARILIA', 17516290,34141959,34141957,'ESTUDANTE','NAO',
'NAO','NAO','NAO','SIM','SIM','SIM','NAO','SIM','NAO','NAO','NAO','SIM','NAO',
'SIM','NAO','NAO',1)
```

```
INSERT INTO PACIENTE VALUES (2,'ALINE','26-11-1981',NULL,350997822,22438819855,'S','RUA
TAL','BAIRRO TAL','MARILIA', 17516290, 34141959,34141957,'ESTUDANTE','
NAO','NAO','NAO','NAO','SIM','SIM','SIM','NAO','SIM','NAO','NAO','NAO','SIM','NAO','SIM',
'NAO','NAO',1)
```

```
INSERT INTO PAGAMENTO
(NUMPAGAMENTO,CODTRATAMENTO,DATAPAGAMENTO,DATAVENCIMENTO,VALORPAG
AMENTO) VALUES(1,1,'8-11-07','29-11-07',100);
```

```
INSERT INTO PAGAMENTO
(NUMPAGAMENTO,CODTRATAMENTO,DATAPAGAMENTO,DATAVENCIMENTO,VALORPAG
AMENTO) VALUES(2,1,'8-11-07','09-11-07',150);
```

```
INSERT INTO PROCEDIMENTO(CODPROCEDIMENTO,VALORPROCEDIMENTO,DESCRICAO)
VALUES(1,50,'FAZER CIRURGIA');
```

```
INSERT INTO PROCEDIMENTO(CODPROCEDIMENTO,VALORPROCEDIMENTO,DESCRICAO)
VALUES(2,35,'IMPLANTAR PROTESE');
```

```
INSERT INTO PRONTUARIO
VALUES(1,1,1,1,12,34,1,'NORMAL','NORMAL','NORMAL','NORMAL','NORMAL','NORMAL','NOR
MAL',
'NORMAL','NORMAL','NORMAL','NORMAL','NORMAL','15-11-07','NORMAL');
```

```
INSERT INTO SECRETARIA
(COD_SECRETARIA,NOMESECRETARIA,ENDERSECRETARIA,CPFSECRETARIA,TELSECRET
ARIA)
VALUES(1,'SANDRA','RUA CAMPUS',33452212411,34141954);
```

```
INSERT INTO TRATAMENTO
(CODTRATAMENTO,CODDENTISTA,DATAINICIO,CODPACIENTE,TIPOTRATAMENTO)
VALUES(1,1,'05-08-07',1,'CANAL');
```



```
INSERT INTO TRATAMENTO  
(CODTRATAMENTO,CODDENTISTA,DATAINICIO,CODPACIENTE,TIPOTRATAMENTO)  
VALUES(2,1,'05-08-07',2,'CIRURGIA');
```

```
INSERT INTO TRATAMENTO_PROCEDIMENTO  
(CODTRATAMENTO,CODPROCEDIMENTO,NUMERODENTE,  
NUMEROFACE,VALORTRATAMENTO) VALUES  
(1,1,13,12,100);
```

```
INSERT INTO TRATAMENTO_PROCEDIMENTO  
(CODTRATAMENTO,CODPROCEDIMENTO,NUMERODENTE,  
NUMEROFACE,VALORTRATAMENTO) VALUES  
(2,2,16,14,150);
```

Apêndice C – *Procedure* para calcular multa

```
CREATE OR REPLACE PROCEDURE MULTA(VDATAVENC IN DATE)
IS BEGIN
  DECLARE VPORCENTAGEM NUMBER(05,2);
  BEGIN
    VPORCENTAGEM:=2/100;
    IF (SYSDATE>VDATAVENC) THEN
      UPDATE PAGAMENTO SET VALORPAGAMENTO = VALORPAGAMENTO +
        (VALORPAGAMENTO * VPORCENTAGEM)
        WHERE (DATAVENCIMENTO = VDATAVENC);
    END IF;
  END;
END;
```