


**Restrições Dinâmicas em Bancos  
de Dados Ativos Orientados a  
Objetos**

**Pedro Rafael Falcone Sampaio**

# Restrições Dinâmicas em Bancos de Dados Ativos Orientados a Objetos

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. Pedro Rafael Falcone Sampaio e aprovada pela Comissão Julgadora.

Campinas, 15 de Dezembro de 1994.



Claudia Bauzer Medeiros  
*Orientadora*

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

# Restrições Dinâmicas em Bancos de Dados Ativos Orientados a Objetos<sup>1</sup>

Pedro Rafael Falcone Sampaio<sup>2</sup>

Departamento de Ciência da Computação  
IMECC – UNICAMP

Banca Examinadora:

- Geovane Cayres Magalhães (Suplente)<sup>3</sup>
- Claudia Bauzer Medeiros(Orientadora)<sup>1</sup>
- Ivan Luiz Marques Ricarte<sup>4</sup>
- Jacques Wainer<sup>3</sup>

---

<sup>1</sup>Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação da UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

<sup>2</sup>O autor é Bacharel em Ciência da Computação pela Universidade Estadual do Ceará

<sup>3</sup>Professor do Departamento de Ciência da Computação – IMECC – UNICAMP.

<sup>1</sup>Professora do Departamento de Ciência da Computação – IMECC – UNICAMP.

<sup>4</sup>Professor do Departamento de Engenharia de Computação e Automação Industrial – FEE – UNICAMP.

# Dedicatória

Dedico a Deus,

São Judas Thadeu,  
São José,

meus pais e meus irmãos (Pedro Dario, Rafaela, Tadeu e Dariella)

minha noiva Sandra

e Lisa.

# Agradecimentos

Diversas pessoas físicas e jurídicas tiveram importância fundamental na execução deste trabalho. Os principais agradecimentos são destinados para:

- Minha orientadora Dra. Claudia Bauzer Medeiros, que me indicou o caminho das pedras e me orientou com competência durante esta jornada.
- A FAPESP e o CNPq que deram o apoio financeiro durante o mestrado.
- Aos professores que participaram da banca: Ivan Ricarte e Jacques Wainer, pelas sugestões e comentários.
- Ao DCC: professores, funcionários (em especial ao Luiz) e alunos, que compartilharam comigo suas experiências, conhecimento e solidariedade.
- Aos professores Tomasz, Claudia, Catto, Setubal, Rezende, Geovane, Cortes pela eficiência durante as disciplinas que fiz.
- Meu pai Pedro Dario, que sempre me apoiou moralmente e financeiramente durante o mestrado.
- Aos Meus Amigos do Ceará: Marcus Vinícius, Ronaldo, Flávio e Victor, que mostraram que amizade é um patrimônio maior do que qualquer valor material.
- Aos meus amigos da PÓS: Carlos, Kelner, Helaine, Helvio, Humberto, Jaime, Nuccio, Marques, Otávio, Rober, Juliano, Helena, Mário, Ronaldo Alves, Fábio, George, Parreira, Daniel, Inês, Lewis, Clevan, Mateus, Nabor, Anderson, Lima, Rosiane, Camponogara, Garcia, Carrilho, Evandro, Fileto, Tutumi, Cristina, Ricardo, Wesley, Luiz, Geraldo, Reginaldo, Horácio.
- Aos meus companheiros de sala: Maria, Ana, Márcio, Glienke.
- Ao meu amigo Mariano, pela amizade e por todas as discussões técnicas sobre banco de dados.
- Ao professor Pedro Sérgio e a turma da Combinatória, pelas cervejadas animadas.

- Aos professores do Ceará: Marcelino, Guy, Everardo, Raimundo Wilson, Tarcisio e Ellery, que me recomendaram e confiaram na minha capacidade.
- A galera do Volley.
- A família Melo Barros (Cris, Sandra, Dr. Adilson, Dra. Teresa) pelos momentos descontraídos e pela imensa cultura comigo compartilhada.
- Mais uma vez ao Marcus (Marcão) e a Andréia, pelo apoio, amizade e pelos incontáveis favores. Espero um dia poder retribuir pelo menos parte!
- Ao meu companheiro de residência e grande amigo Victor, pela educação, apoio, companheirismo e amizade durante esta longa jornada.

# Resumo

Esta dissertação aborda o problema da modelagem e manutenção de restrições gerais de integridade em sistemas de banco de dados. A solução se baseia no uso de gerenciadores de bancos de dados ativos orientados a objetos que suportam mecanismos de regras de produção.

O trabalho apresenta uma estratégia para o projeto de aplicações em sistemas de informação que leva em consideração as características comportamentais e ativas do gerenciador de banco de dados. O objetivo da estratégia é possibilitar a representação das restrições de forma declarativa e independente de modelo de dados durante o projeto conceitual e identificar mapeamentos em termos de regras de produção destinadas a manter a consistência. As restrições são especificadas em CDL, uma nova linguagem proposta na dissertação.

As principais contribuições são: uma taxonomia para restrições de integridade na modelagem de sistemas de informação; a linguagem — CDL — para especificar tais restrições; heurísticas gerais de mapeamento das restrições expressas em CDL para regras de produção no banco de dados ativo; e a especificação das características necessárias a um banco de dados ativo para que possa permitir a manutenção de quaisquer restrições de integridade em sistemas de informação.

Este trabalho estende as propostas anteriores encontradas na literatura ao incorporar a modelagem de restrições dinâmicas no projeto de sistemas de bancos de dados usando gerenciadores de bancos de dados orientados a objetos com capacidade ativa.

# Abstract

This dissertation addresses the problem of modelling and enforcing general integrity constraints in database systems. The solution is based on the use of active object-oriented DBMS that provide support to rule mechanisms.

The work proposes a strategy to be applied during application design. This strategy takes into consideration the behavior and active features of the DBMS. The strategy's goal is to represent the constraints in the conceptual design using CDL — a declarative and model independent language and to provide mappings in terms of production rules responsible for constraint enforcement.

The main contributions presented are: the proposal of a taxonomy for integrity constraints in modelling information systems; the specification of the CDL constraint language; general heuristics for mapping constraints expressed in CDL into production rules in the active database; and the specification of the characteristics needed from an active database in order to support general integrity constraints in information systems.

This dissertation extends previous proposals found in the literature, providing support to model dynamic constraints in database system design using active object-oriented database management systems.



# Conteúdo

Dedicatória	iv
Agradecimentos	v
Resumo	vii
Abstract	viii
<b>1 Introdução e Motivação</b>	<b>1</b>
1.1 Requisitos em Sistemas de Informação	1
1.2 Metodologias de Projeto	5
1.2.1 Metodologias de Projeto de Sistemas	6
1.2.2 Metodologias de Projeto de Sistemas de Informação	9
1.3 Projeto de Banco de Dados para Sistemas de Informação	10
1.3.1 Arquitetura ANSI/SPARC e o relacionamento com as Fases de Projeto de Banco de Dados	11
1.3.2 Projeto Conceitual	13
1.3.3 Modelagem Conceitual de Restrições	17
1.4 Incorporação de Regras em Banco de Dados	17
1.5 Organização da Tese	19
<b>2 Sistemas Ativos</b>	<b>21</b>
2.1 Introdução	21
2.2 Linguagens para Descrição de Regras Ativas	25
2.2.1 Regras E-C-A	25
2.2.2 Suporte a Mecanismos de Eventos	25
2.2.3 Tipos de Eventos	27
2.2.4 Álgebras de Eventos	29
2.2.5 Regras E-C-A Temporais	29

<b>3</b>	<b>Especificação e Controle de Restrições</b>	<b>31</b>
3.1	Introdução . . . . .	31
3.2	Tempora . . . . .	32
3.2.1	Modelo ERT . . . . .	32
3.2.2	Modelo de Regras . . . . .	35
3.3	$T_{ROLL}$ . . . . .	36
3.4	Controle de Restrições via Programação por Contrato . . . . .	38
3.4.1	Especificação de Restrições . . . . .	38
3.4.2	Mapeamento . . . . .	40
3.5	Abordagem de Urban et al . . . . .	41
<b>4</b>	<b>Taxonomia de Restrições na Modelagem Conceitual</b>	<b>43</b>
4.1	Introdução . . . . .	43
4.2	Restrições Sobre Estados . . . . .	46
4.3	Restrições sobre Comportamento . . . . .	48
4.3.1	Permissões . . . . .	50
4.3.2	Obrigações . . . . .	51
4.3.3	Comportamento X Restrições Sobre Comportamento . . . . .	52
4.4	Relacionamentos Intervalares Definindo Restrições . . . . .	53
4.5	Restrições de Transição . . . . .	53
<b>5</b>	<b>Projeto Conceitual de Banco de Dados em Camadas</b>	<b>55</b>
5.1	Introdução . . . . .	55
5.2	Descrição da Estratégia Proposta . . . . .	59
<b>6</b>	<b>Linguagem de Definição de Restrições (CDL)</b>	<b>61</b>
6.1	Introdução . . . . .	61
6.2	Linguagem de Definição de Restrições (CDL) . . . . .	63
6.3	Estudo de Caso . . . . .	65
6.4	Descrição do Nível de Objetos . . . . .	66
6.4.1	Especificação de Restrições de Integridade Estáticas . . . . .	68
6.5	Descrição do Nível de Restrições – Restrições Dinâmicas e de Comportamento	68
<b>7</b>	<b>Mapeamento de Restrições em CDL</b>	<b>70</b>
7.1	Ambiente Ativo Alvo . . . . .	70
7.2	Mapeamento de restrições . . . . .	71
7.2.1	Mapeamento de Restrições sobre Estados . . . . .	73
7.2.2	Mapeamento de Restrições sobre Comportamento . . . . .	74
7.3	Exemplos . . . . .	78

---

<b>8</b>	<b>Conclusões e Extensões</b>	<b>84</b>
<b>A</b>	<b>Linguagem de Definição de Restrições (CDL)</b>	<b>86</b>
	<b>Bibliografia</b>	<b>91</b>

# Lista de Tabelas

1.1	Problemas, Características e Alternativas de Implementação . . . . .	4
1.2	Exemplos de Aplicações . . . . .	5

# Lista de Figuras

1.1	Visão Quadrimensional dos Requisitos de Aplicações em Sistemas de Informação . . . . .	3
1.2	Princípios, Técnicas, Estratégias e Metodologias . . . . .	6
1.3	Terminologia para Projeto de Banco de Dados . . . . .	12
1.4	Terminologia Moderna para Projeto de Banco de Dados . . . . .	12
1.5	Projeto de Banco de Dados e Funções para S.I . . . . .	14
2.1	Eixo do Tempo Associado ao Sistema . . . . .	26
2.2	Taxonomia de Eventos em Sistemas Ativos. . . . .	27
3.1	Ilustração do Modelo ERT . . . . .	34
3.2	Geração de Regras Usando Análise de Restrições . . . . .	42
4.1	Taxonomia de Restrições . . . . .	45
4.2	Diagrama de Transição de Estado . . . . .	52
4.3	Relacionamentos Intervalares . . . . .	54
5.1	Camadas Para a Descrição do Modelo Conceitual . . . . .	57
5.2	Estratégia de Projeto Usando BDOO Ativo . . . . .	60
6.1	Exemplo Expresso Usando Diagrama de Objetos de OMT . . . . .	67

# Capítulo 1

## Introdução e Motivação

Restrições Externas ou Explícitas são regras usadas para definir critérios de consistência estáticos e dinâmicos que não são adequadamente representados nos modelos de dados das aplicações. Estas restrições precisam ser incorporadas às diferentes aplicações, o que encarece sua manutenção e gerenciamento.

Uma solução alternativa encontrada para esse problema é o uso de mecanismos de regras de produção acoplados a gerenciadores de bancos de dados conhecidos como *Bancos de Dados Ativos*. Com o suporte provido pelo banco de dados ativo, o código associado ao tratamento das restrições é deslocado das aplicações para as regras de produção destinadas ao monitoramento do ambiente de execução, simplificando o gerenciamento e a manutenção das restrições.

Esta dissertação aborda o problema da modelagem e manutenção de restrições explícitas no projeto de sistemas de bancos de dados para ambientes orientados a objetos com capacidade ativa. É dedicada especial atenção ao tratamento das restrições dinâmicas, não abordadas na maioria dos trabalhos publicados na literatura.

O problema é abordado de forma integrada, partindo da modelagem das aplicações e chegando à especificação das restrições como regras incorporadas ao banco de dados. Este capítulo fornece a base conceitual e a motivação para o trabalho desenvolvido.

### 1.1 Requisitos em Sistemas de Informação

Durante a modelagem de um sistema, podemos identificar cinco tipos de informação associados ao domínio da aplicação a ser automatizada[Oea91]:

- **Informação Estrutural:** representando fatos associados aos objetos do domínio conceitual da aplicação como Entidades, Relacionamentos e Atributos.
- **Informação Funcional:** descrevendo a lógica dos processos que modificam o estado dos objetos da aplicação.

- **Informação Temporal:** identificando o momento da manipulação dos objetos, noções como calendários e datas relevantes a aplicação, e a validade no mundo real dos objetos da aplicação tomando por base diversos pontos de uma escala temporal.
- **Informação Comportamental ou de Controle:** indicando qual o comportamento ou quais ações são desempenhadas pelas entidades por ocasião de um evento ocorrendo no sistema.
- **Informação Restritiva** caracterizando regras pertencentes ao domínio da aplicação que restringem e governam os outros tipos de informação supra citados. Podem estar associadas aos estados (integridade estática e dinâmica) ou ao comportamento (restrições comportamentais).

Estas perspectivas constituem um espectro das informações manipuladas nos diversos domínios de problemas existentes e que são traduzidas em termos de requisitos de dados, funções, tempo e controle. Após o levantamento dos requisitos, inicia-se a modelagem dos diversos aspectos relevantes à construção das soluções.

As informações funcionais, de controle, restrições sobre o comportamento e restrições dinâmicas de integridade constituem o aspecto dinâmico da modelagem enquanto que as informações estruturais e restrições de integridade estáticas constituem o aspecto estático.

A informação associada a aspectos temporais, além de representar requisitos estruturais, pode ter porções representadas por intermédio das restrições sobre o comportamento e integridade dinâmica, respectivamente associadas à definição do momento de execução ou ordem relativa das ações e às seqüências de estados admissíveis no decorrer do tempo.

A Figura 1.1 ilustra as quatro dimensões que representam os requisitos dos diversos domínios de problemas em Sistemas de Informação. Cada problema representa uma combinação das quatro dimensões, sendo que determinadas dimensões podem ter uma importância maior na elaboração de uma solução. A dimensão temporal estática identifica requisitos estruturais (histórico de entidades, relacionamentos e atributos) e está representada pela região sombreada saindo do eixo associado aos requisitos estruturais, enquanto que a região sombreada saindo do cubo representa a dimensão temporal expressa sob a forma de restrições temporais. Requisitos identificando restrições não temporais e restrições temporais expressas através de restrições sobre as outras três dimensões, estão imersos no cubo.

Cada ponto negro representa um domínio de problema onde há uma predominância das dimensões assinaladas. Uma metodologia destinada a representar e desenvolver soluções neste domínio deve considerar tais características.

A tabela 1.1 ilustra os pontos de interseção, as principais características das aplicações pertencentes aos domínios assinalados e possíveis tecnologias usadas no desenvolvimento da solução.

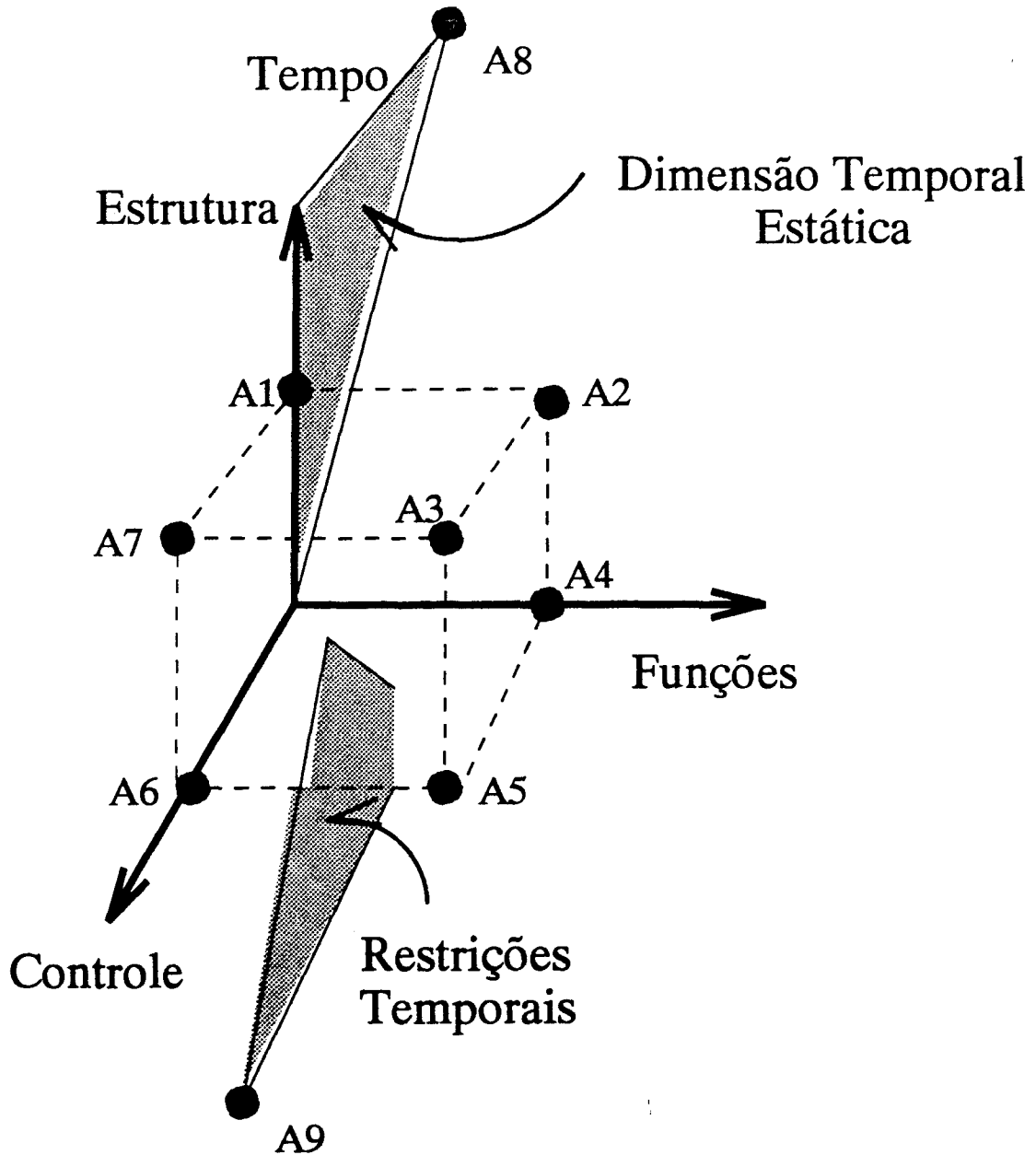


Figura 1.1: Visão Quadridimensional dos Requisitos de Aplicações em Sistemas de Informação



Ponto	Características	Ferramentas
A1	Problemas associados ao armazenamento de dados	Sistemas de Arquivos
A2	Problemas associados a manipulação de informações e processamento de transações	SGBD convencionais
A3	Problemas associados a manipulação de informações, processamento de transações e controle de eventos associados aos dois itens anteriores	SGBD Ativo
A4	Problemas altamente transacionais	Linguagens de Alto Nível
A5	Problemas associados ao monitoramento de transações	Monitores de Transações
A6	Problemas associados ao monitoramento de eventos	Sensores
A7	Problemas dirigidos por dados e controle	Linguagens Baseadas em Regras de Produção
A8	Problemas com manipulação de informação histórica e processamento de transações	SGBD com dimensão temporal, Mecanismos de Controle de Versões
A9	Problemas associados a representação e manutenção de regras temporais pertencentes ao domínio das aplicações	SGBD Ativo Temporal, Sistemas com Representação de Conhecimento Temporal

Tabela 1.1: Problemas, Características e Alternativas de Implementação

A tabela 1.2 apresenta alguns exemplos de aplicações pertencentes aos domínios identificados.

Ponto	Aplicações
A1	Sistemas de Mala Direta, Cadastros
A2	Sistemas Bancários, Sistemas Contábeis
A3	Sistemas de Tráfego Aéreo, Sistemas de Automação de Escritórios, Gerenciamento de Redes
A4	Sistemas para reserva de recursos
A5	Controle de Mísseis
A6	Sistemas Embutidos (Caldeiras, Controle de Processos)
A7	Sistemas Especialistas (Diagnóstico e Projeto)
A8	<i>Data Mining</i> , Sistemas Hospitalares,
A9	Sistemas de Previsão de Tempo, Controle de Ações, Previsão de Investimento Financeiro, <i>Planning</i>

Tabela 1.2: Exemplos de Aplicações

## 1.2 Metodologias de Projeto

Metodologias de projeto<sup>1</sup> são utilizadas para representar os requisitos e permitir que o *Gap Semântico* existente entre a análise e a implementação da solução do problema seja gradativamente reduzido através de fases intermediárias de projeto.

Qualquer metodologia normalmente versa sobre dois aspectos ortogonais a serem considerados no desenvolvimento da solução:

- **Processo de Projeto:** Descreve o *Modelo de Ciclo de Vida* adotado, determinando as principais etapas do projeto e o *Planejamento*, que define o cronograma a ser respeitado durante o projeto, a alocação de pessoas a cada fase e os critérios de qualidade.
- **Estratégia de Projeto:** Identifica os principais modelos e notações empregadas durante o projeto, além das técnicas de modelagem e regras de mapeamento e integração entre níveis distintos de abstração.

---

<sup>1</sup>Neste texto, a palavra PROJETO pode ser empregada para referir-se a todas as etapas do ciclo de vida de construção com exceção da fase de Manutenção, ou alternativamente, para referir-se a etapa posterior a análise de requisitos. Cabe ao leitor a identificação do contexto apropriado.

As metodologias representam paradigmas para desenvolvimento de soluções adotados por determinados grupos de desenvolvimento. Metodologias são construídas tendo como base certos princípios usados durante a construção do produto. Alguns destes princípios são[GMJ91]:

- Dividir para conquistar
- Abstração
- Modularidade

Modelos e técnicas de modelagem servem como ferramentas para a construção das diversas representações (Espaço do Problema e Espaço da Solução), de acordo com os princípios básicos existentes. Uma metodologia agrupa os modelos e técnicas de modelagem usadas durante o desenvolvimento da solução e indica como são mapeadas as diversas representações do sistema, desde o espaço de problemas até a solução.

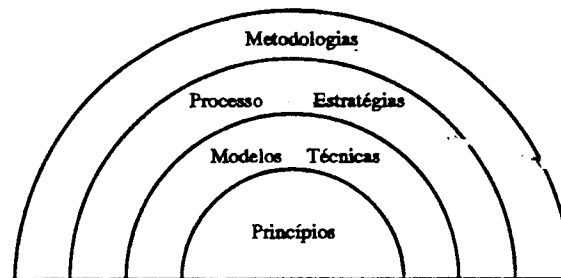


Figura 1.2: Princípios, Técnicas, Estratégias e Metodologias

Metodologias também apresentam heurísticas acerca do processo de desenvolvimento, dividindo-o em etapas que constituem o ciclo de vida do desenvolvimento da solução. As métricas usadas como base para o planejamento são derivadas a partir de experiências passadas de desenvolvimento usando a mesma metodologia.

Embora normalmente as metodologias apresentem modelos e técnicas endereçando as diversas dimensões de problemas existentes, algumas metodologias mostram-se mais adequadas do que outras para o desenvolvimento de soluções dentro de um determinado domínio de problemas. A Figura 1.2 ilustra os principais aspectos envolvidos na definição de metodologias para a solução de problemas.

### 1.2.1 Metodologias de Projeto de Sistemas

Metodologias de Projeto de Sistemas são adequadas para a construção de software. Dentre os principais domínios de sistemas existentes podemos destacar[GMJ91]

- Sistemas Automatizados pertencentes a Sistemas de Informação Organizacionais<sup>2</sup>

<sup>2</sup>Sistemas automatizados construídos sobre o banco de dados do empreendimento

- Sistemas de Tempo Real
- Sistemas Distribuídos
- Sistemas Reativos (Interfaces)
- Software Numérico e Software para Otimização

Cada domínio apresenta requisitos específicos que definem a funcionalidade exigida do sistema a ser implementado e que devem ser atendidos de forma a alcançar os padrões de qualidade exigidos.

### Classes de Metodologias de Projeto de Sistemas

As diversas metodologias propostas para o desenvolvimento de sistemas podem ser agrupadas em duas classes principais:

- Metodologias Estruturadas
- Metodologias Orientadas a Objetos

Metodologias de uma mesma classe apresentam pequenas variações entre si, seguem os mesmos princípios gerais, adotam técnicas e estratégias similares no desenvolvimento da solução e normalmente empregam a mesma tecnologia (Linguagens e Banco de Dados).

As metodologias de **Projeto Estruturado**[YC79] empregam o Diagrama de Fluxo de Dados juntamente com a técnica de modelagem funcional durante a fase de Análise de requisitos. Adotam diagramas de estrutura, Normalização de Depósitos de Dados e técnicas de projeto estruturado na fase de Projeto e utilizam o modelo e técnicas de programação estruturada[Dij76] (Construções *If Then Else*, *Repeat Until*, *While Do*, *atribuição e seqüência*), na fase de Implementação.

As Metodologias de **Projeto Orientado a Objetos**[Boo93, RBP+91, Ner92, CY91] podem ser caracterizadas pelo modo como incorporam idéias de outros paradigmas de desenvolvimento de sistemas. As principais abordagens são[MP92]:

- **Combinada:** nesta abordagem, técnicas e notações orientadas a objetos, de especificação de controle e de especificação de funções são usadas independentemente e combinadas na produção de um modelo integrado do sistema. A metodologia OMT[RBP+91] é um exemplo desta abordagem.
- **Adaptativa:** nesta abordagem, notações e técnicas existentes são adaptadas incorporando conceitos de orientação a objetos. Como exemplo, a notação proposta por Kappel[Kap91], estende um diagrama de estados produzindo uma versão orientada a objetos.

- **Pura:** nesta abordagem, notações e técnicas orientadas a objetos são utilizadas para modelar tanto os requisitos estáticos quanto os requisitos dinâmicos (funções e controle). Exemplos desta abordagem são: A metodologia de projeto proposta por Meyer[Mey88] e a proposta de Booch[Boo93].

As propostas consideradas dentro das duas classes de metodologias anteriormente apresentadas são bastante adequadas para o projeto de sistemas. Contudo, tratam de forma insatisfatória os requisitos inerentes ao projeto de sistemas de banco de dados para sistemas de informação. Dentre os pontos fracos apresentados podemos destacar:

- Ausência de modelos conceituais apropriados para descrever a semântica do empreendimento
- Subutilização das funções providas pelo SGBD
- Representação insatisfatória de Restrições
- Tratamento insuficiente do conceito de *Persistência*

A partir do surgimento do modelo E/R[Che76], foram propostas várias metodologias de projeto de banco de dados para sistemas de informação utilizando modelos conceituais de dados. As principais características destas metodologias são:

- Aplicação da técnica de Modelagem Conceitual das estruturas de informação usando modelos de dados semânticos
- Elaboração de estratégias para mapeamento do modelo conceitual em termos do modelo operacional do gerenciador de banco de dados
- Existência de uma fase onde são escolhidas as melhores estruturas de dados e os métodos de acesso destinados a atender os requisitos de desempenho durante a manipulação das informações.
- Utilização do conceito de **Independência de Dados**, que permite a modificação das estruturas de dados e métodos de acesso sem afetar as aplicações (Independência Física). além de permitir alterações a nível do modelo operacional do SGBD, deixando intactas as aplicações que não utilizam a porção alterada do modelo de dados (Independência Lógica).

## 1.2.2 Metodologias de Projeto de Sistemas de Informação

Um sistema de informação é uma unidade funcional de processamento de informações de uma organização composta de elementos (pessoas, subsistemas e dispositivos de processamento) desempenhando atividades, sujeitos a restrições e comunicando-se através de protocolos de comunicação. A comunicação entre os agentes (pessoas e subsistemas) ocorre através da passagem de mensagens sendo que o conhecimento (procedural, estrutural e normativo) é compartilhado por todos os elementos. As atividades podem ser desempenhadas de modo concorrente e assíncrono tendo porções estruturadas (automatizáveis) e pouco estruturadas.

Para a concepção e construção do sistema de informação de uma organização, várias técnicas são empregadas para descrever os aspectos manuais do sistema. Além disso, também são utilizadas metodologias de projeto de sistemas e metodologias de projeto de banco de dados para o desenvolvimento dos vários sistemas automatizados dentro do SI.

Ainda não há um consenso quanto a uma metodologia para o projeto de um sistema de informações para uma organização. A primeira tentativa de integração tanto dos aspectos automatizados quanto dos aspectos manuais foi feita na proposta de Engenharia da Informação de Martin[Mar91].

Sistemas de informação são sistemas reativos construídos em torno de um banco de dados[EJDS92]. Os avanços tecnológicos possibilitaram o desenvolvimento de SI baseados em grandes bancos de dados distribuídos, contendo informações não formatadas e objetos complexos multimídia[EdO94]. Para a construção do banco de dados sobre o qual estarão implementadas as aplicações, aplica-se a técnica de modelagem conceitual de dados e posteriormente desenvolve-se o projeto lógico e o projeto físico do banco de dados. Para a construção dos aplicativos sobre o banco de dados, empregam-se as técnicas de análise e projeto de sistemas.

Dentre as principais técnicas de modelagem aplicadas na construção de sistemas de informação podemos destacar[Ede94]:

- **Técnicas baseadas na modelagem de Processos.** Nesta abordagem, as atividades sendo desempenhadas possuem importância principal e a finalidade primordial da especificação gerada é representar de forma integrada todas as funções executadas no empreendimento, caracterizando o aspecto dinâmico das organizações.
- **Técnicas baseadas na modelagem de Agentes.** Nesta abordagem, as atividades realizadas na organização são associadas aos elementos responsáveis pela execução (agentes), caracterizando os diversos papéis cabíveis a um agente.
- **Técnicas baseadas na modelagem do Controle.** Nesta perspectiva, o empreendimento é visto como uma máquina de estados finitos, procurando-se representar

os estados e transições de estados ocasionadas por eventos no domínio do empreendimento

- **Técnicas de Modelagem Conceitual.** Nesta abordagem, as estruturas de informação do empreendimento são modeladas em termos de entidades, atributos, relacionamentos e restrições. A modelagem prossegue até a fase de projeto de banco de dados, onde os recursos de gerenciamento e manipulação providos pelos SGBD são usados visando simplificar o projeto das aplicações
- **Técnicas de Análise e Projeto de Sistemas.** Neste contexto, tanto metodologias estruturadas quanto as metodologias orientadas a objetos são usadas para a construção das funções automatizadas que fazem parte do S.I.

Em determinados sistemas de informação algumas das técnicas anteriormente descritas não são utilizadas. Nos Sistemas de Informação de Escritórios, várias das técnicas são aplicadas por se tratarem de sistemas sócio-técnicos. A especificação do SIE, além dos aspectos usuais de um sistema como aplicações e banco de dados, também deverá comportar os seguintes aspectos[EdO94]:

1. Estrutura organizacional do escritório
2. Hierarquia de composição das atividades
3. Alocação de recursos
4. Coordenação das atividades concorrentes
5. Troca de informações entre pessoas e tarefas
6. Tomada de decisões

## 1.3 Projeto de Banco de Dados para Sistemas de Informação

Algumas das questões fundamentais que devem ser consideradas por ocasião do desenvolvimento de banco de dados e aplicações para sistemas de informação são:

- **Integridade dos Dados**
- **Segurança e Controle de Acessos**
- **Representação Adequada da Realidade**

- **Desempenho das Transações**

Para simplificar o projeto considerando as questões acima e a interação com o banco de dados, níveis de abstração são definidos, identificando interfaces através das quais os diversos usuários, analistas de banco de dados e projetistas de aplicações terão acesso aos recursos providos pelo SGBD. A Integridade dos Dados é considerada principalmente nas fases de projeto conceitual e projeto lógico; a Segurança é abordada no projeto lógico; a Representação da Realidade é a principal preocupação do projeto conceitual, enquanto que os critérios de desempenho de transações, por sua vez, são tratados durante o projeto físico.

### 1.3.1 Arquitetura ANSI/SPARC e o relacionamento com as Fases de Projeto de Banco de Dados

A arquitetura ANSI-X3-SPARC para banco de dados é organizada em termos de três níveis de abstração, onde cada nível é representado por intermédio de um esquema. Os níveis especificados são:

- **Nível Interno ou Físico:** este nível define a forma como os arquivos de dados das aplicações são organizados, juntamente com os métodos e estruturas auxiliares de acesso definidas sobre os dados. O esquema interno é gerado mediante o uso da linguagem de definição de dados interna (DDL interna).
- **Nível Conceitual:** este nível constitui a visão lógica global dos dados sob a ótica do modelo operacional de dados do gerenciador (hierárquico, rede, relacional ou orientado a objetos). O esquema Lógico, Operacional ou Conceitual<sup>3</sup> é definido usando os comandos da linguagem de definição de dados conceitual (DDL conceitual).
- **Nível Externo ou Visão:** este nível é formado pelas visões que usuários ou grupos de usuários têm do banco de dados. Cada esquema externo ou visão é definido usando o subconjunto de comandos da linguagem de definição de dados associados a concepção de visões (DDL externa).

Na literatura de banco de dados, muitas vezes os termos Projeto Conceitual, Projeto Lógico e Projeto Físico vêm sendo empregados para denotar conceitos distintos. Em algumas abordagens, o **projeto lógico** refere-se à etapa onde é aplicada a técnica de modelagem conceitual enquanto que o **projeto físico** refere-se à etapa onde é produzido o esquema correspondente ao nível conceitual da arquitetura ANSI/SPARC[Set86]. A Figura 1.3 ilustra este ponto de vista.

---

<sup>3</sup>O termo **Esquema Conceitual**, embora seja empregado para descrever o nível conceitual da arquitetura ANSI/SPARC, será empregado nesta dissertação para descrever o esquema gerado pela modelagem conceitual. O termo **Esquema Lógico** será empregado para descrever o nível conceitual de ANSI/SPARC.



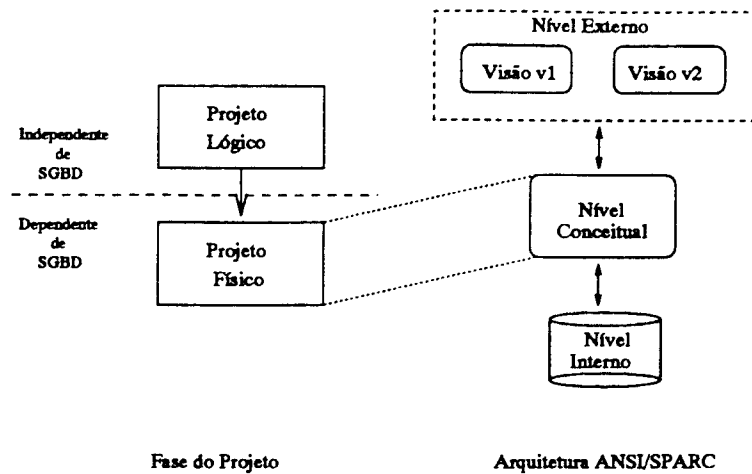


Figura 1.3: Terminologia para Projeto de Banco de Dados

Em trabalhos mais recentes[EN89, BCN92, Tan92, JSH91], o termo **projeto conceitual** é empregado para caracterizar a etapa na qual, dentre outras técnicas, é aplicada a modelagem conceitual, onde um modelo de dados semântico é empregado resultando na elaboração do esquema conceitual das estruturas de informação. Ainda nesta nova terminologia, o termo **projeto lógico** é usado denotando a fase onde ocorre o mapeamento do esquema conceitual em termos do esquema lógico do nível conceitual da arquitetura ANSI/SPARC. Técnicas como a *Normalização* também são aplicadas durante esta fase. Por último, adota-se o termo **projeto físico** para identificar a fase onde são escolhidas as organizações de arquivos e os métodos de acesso usados para descrever o nível interno da arquitetura ANSI/SPARC. Esta dissertação adota esta nova terminologia que é ilustrada na Figura 1.4.

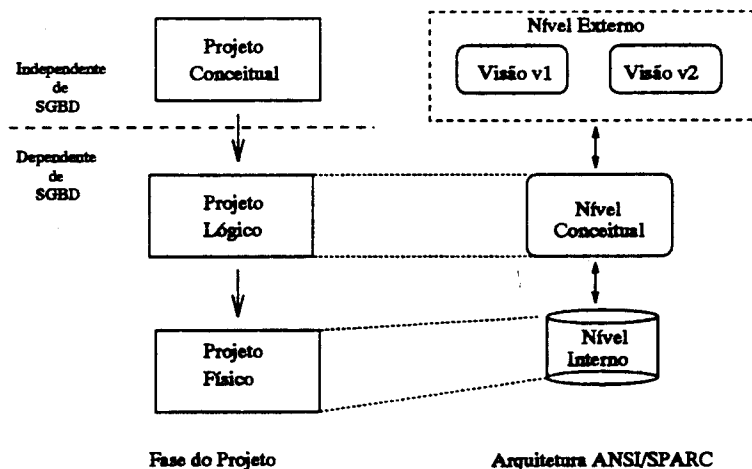


Figura 1.4: Terminologia Moderna para Projeto de Banco de Dados

O projeto de sistemas automatizados no domínio de sistemas de informação normalmente procede separando dados de processos, dando seqüência ao projeto de cada vertente separadamente — Projeto Conceitual, Projeto Lógico e Projeto Físico do banco de dados e Análise e Projeto para as funções.

Esta separação entre dados e aplicações segue até a implementação, onde dados são armazenados em banco de dados e gerenciados pelo SGBD enquanto que programas de aplicação que acessam o banco de dados são implementados em linguagens de programação. A Figura 1.5 ilustra tal ponto de vista.

### 1.3.2 Projeto Conceitual

Os recursos principais de um S.I são as informações, operações de manipulação das informações e restrições pertencentes ao domínio do empreendimento. A modelagem conceitual do empreendimento representa a técnica mais importante aplicada durante a concepção do S.I. A modelagem ocorre durante o projeto conceitual e funciona como ponto de partida para o projeto do banco de dados.

O resultado da fase de projeto conceitual é uma representação contendo dois componentes:

- **Esquema Conceitual**, representando os objetos, atributos e relacionamentos entre objetos, juntamente com as restrições estáticas de integridade.
- **Especificação de Propriedades Dinâmicas**, descrevendo as restrições dinâmicas (integridade e comportamento), possíveis consultas e transações.

### Modelagem de Dados

Um *Modelo de Dados* constitui uma coleção de conceitos matematicamente bem definidos que permitem a análise e a representação das propriedades estruturais e dinâmicas das aplicações[Bro84]. Um *Modelo Conceitual* é um modelo de dados usado para descrever os principais aspectos da realidade sendo modelada de uma maneira independente de implementação. Um *Modelo Lógico ou Operacional de Dados* proporciona descrições de dados que podem ser processados pelo software de gerenciamento de banco de dados. *Modelos Semânticos de Dados* são modelos conceituais que procuram incorporar abstrações de forma a representar a semântica da aplicação de uma maneira próxima a realidade.

A necessidade de representar a semântica das aplicações motivou o desenvolvimento de vários modelos semânticos dentre os quais E/R[Che76], SDM[HM81],  $E^2R$ [EN89], ERC+[Spa93]. Modelos de dados semânticos[PM88, HK87] foram inicialmente concebidos para permitir que domínios de aplicação complexos como CAD/CAM, CASE e automação de escritórios pudessem ser representados adequadamente sob o ponto de vista estrutural. Abstrações como especialização e agregação, mecanismos de herança e construtores de

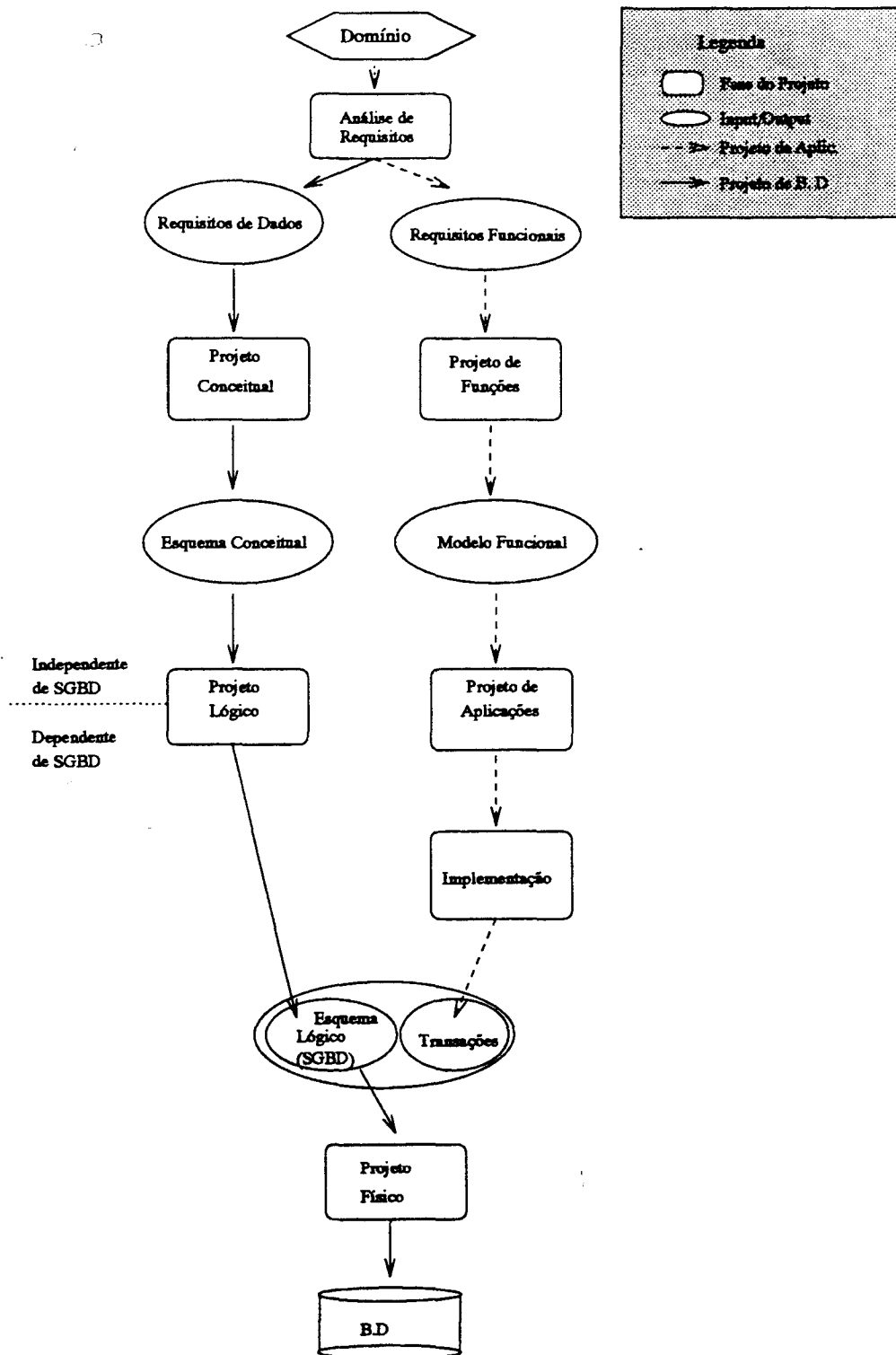


Figura 1.5: Projeto de Banco de Dados e Funções para S.I

tipos foram sucessivamente incorporados aos modelos semânticos permitindo uma melhor representação dos aspectos semânticos das aplicações em questão.

Modelos semânticos mostram-se bastante apropriados para a etapa de projeto conceitual. Contudo, ainda persiste a necessidade que o usuário domine outros paradigmas de modelagem de forma a efetuar a manipulação, já que o esquema conceitual expresso usando o modelo semântico deve ser mapeado em termos do esquema lógico do modelo operacional do gerenciador (tipicamente relacional ou orientado a objetos). Em determinados modelos conceituais, como por exemplo o modelo ERC+, operadores de manipulação são suportados, permitindo que ferramentas gráficas para consulta e manipulação sobre o modelo conceitual sejam projetadas.

Em uma etapa posterior, modelos semânticos como TAXIS[MBW80] e Galileo[ACO85] consideraram a integração dos aspectos comportamentais das aplicações aos aspectos estruturais, representando as primeiras idéias no contexto de modelagem conceitual do comportamento.

Embora o modelo orientado a objetos represente um avanço no projeto de aplicações e software básico, dadas as vantagens proporcionadas por características como reusabilidade, extensibilidade e modularidade, quando considerado no projeto de banco de dados para sistemas de informação, este modelo apresenta dificuldades para representar a semântica dos aspectos do domínio da aplicação. Dentre as principais deficiências do modelo orientado a objetos para a modelagem conceitual de sistemas de informação estão[Spa93]:

- Descrição inadequada de associações
- Ausência de operadores genéricos de manipulação de dados
- Ausência de conceitos para a descrição do comportamento global da aplicação
- Noções como cardinalidade não estão declarativamente representadas
- Semântica das Restrições inseridas na codificação dos métodos

O modelo orientado a objetos apresenta as noções de composição e herança como mecanismos para efetuar relacionamentos. A composição neste modelo funciona como uma noção de implementação (Ponteiro), trazendo aspectos de implementação à tona na etapa de modelagem conceitual. Como exemplo, a associação entre os tipos CARRO e PESSOA representando pertinência pode ser representada de diversas maneiras:

1. Como uma referência de Pessoa para Carro
2. Como referência de Carro para Pessoa
3. Como referências cruzadas entre Pessoas e Carros

4. Como um novo objeto em uma classe PERTENCE ligado por referências cruzadas com objetos das classes CARRO e PESSOA

Portanto, a decisão acerca de como será representada a associação, irrelevante do ponto de vista conceitual, deve ser levada em consideração prematuramente. Além disso, a semântica da associação fica espalhada pelas definições do objetos participantes.

Para suprir tais deficiências, modelos orientados a objetos foram estendidos no sentido de representar satisfatoriamente as associações a nível de modelagem conceitual. O modelo de objetos proposto na metodologia OMT[RBP+91] inclui abstrações de Generalização, Agregação e Associações Genéricas, além das abstrações básicas (composição e herança) já presentes no modelo orientado a objetos. Outro modelo concebido nesta linha é o modelo ERC+[Spa93], que evoluiu a partir do modelo E/R incorporando facilidades para expressão de Objetos Complexos e o conceito de Identidade de Objetos[KC86]. Tais modelos são classificados como *Modelos Objeto Relacionamentos*[Spa93].

A representação de aspectos temporais, por sua vez, vem sendo basicamente desenvolvida tanto no sentido de estender modelos operacionais de dados existentes com aspectos associados às diversas noções temporais (tempo do usuário, tempo de transação e tempo válido), quanto no sentido de definir e formular um modelo temporal consensual[PSE+94], que contemple todos os requisitos temporais inerentes aos diversos domínios de aplicação.

Na maioria das aplicações em sistemas de informação, o conhecimento temporal não é expresso e mantido de forma integrada, estando distribuído entre os programas, o banco de dados e os usuários do sistema, cada qual possuindo parte do conhecimento à disposição. Além disso, as porções que não estão formalmente representadas nas aplicações constituem uma espécie de “senso comum” temporal, sob a responsabilidade das pessoas da organização e isto dificulta sobremaneira a manutenção das aplicações e a uniformização de aspectos temporais em aplicações, como granularidade, calendários e nomenclatura.

Pouco foi feito quanto a modelagem conceitual de aspectos temporais, no sentido de elaborar estruturas e abstrações destinadas a descrever conceitualmente as aplicações com características temporais[Elm91, TL91]. Dentre as propostas existentes nesta direção estão[TLW91, SS88].

A evolução da modelagem conceitual, no entanto, vem focalizando principalmente a captura e especificação dos requisitos que possuem metáforas correspondentes no sistema computacional alvo: estrutura (dados), processos (procedimentos e funções) e controle (eventos), mas continua precária no que tange a representação de algumas características ligadas ao domínio do empreendimento como restrições semânticas, evolução e exceções[LK92].

### 1.3.3 Modelagem Conceitual de Restrições

Restrições são especificações a serem obedecidas pelos estados do banco de dados e pelas operações, delimitando quais estados são possíveis, o que as operações podem fazer, quando e em que condições podem fazê-lo[ACM80]. Restrições caracterizam os estados ou seqüências de estados válidas, além dos critérios de consistência acerca do ciclo de vida dos objetos.

*Restrições Semânticas* são restrições que aparecem no domínio sendo modelado e são classificadas como *Restrições Explícitas ou Externas* quando não puderem ser expressas por intermédio das estruturas do modelo de dados utilizado na modelagem, enquanto que *Restrições Implícitas* são restrições semânticas que podem ser expressas usando o modelo.

*Restrições Internas ou Inerentes* são restrições que definem regras acerca da construção de esquemas usando o modelo de dados. As restrições inerentes devem ser respeitadas, para que os esquemas gerados usando as operações e abstrações providas pelo modelo sejam consistentes. Dentre as principais vantagens da especificação declarativa das restrições durante a modelagem conceitual estão:

- Utilização do conhecimento exposto na otimização de acessos
- Utilização para fins de demonstração da corretude dos procedimentos
- Possibilidade de um controle centralizado da consistência
- Simplificação dos programas de aplicação<sup>4</sup>

Apesar dos avanços no sentido de incorporar a semântica das aplicações durante a modelagem, a representação das informações sobre restrições ainda representa campo aberto na modelagem de sistemas de informação. Nesta direção, notações baseadas em lógica são empregadas possibilitando uma descrição precisa de aspectos que não são normalmente capturados nas estruturas dos modelos de dados semânticos tradicionais.

## 1.4 Incorporação de Regras em Banco de Dados

Uma solução para a incorporação de restrições em sistemas é a utilização de regras de produção associadas ao banco de dados. *Regras de produção* representam formas declarativas de representação de conhecimento e são capazes de descrever relacionamentos entre estados sucessivos do sistema. Tais relacionamentos podem expressar derivação (incorporação de novos fatos), estratificação (retirada de fatos) e desencadeamento de ações.

---

<sup>4</sup>Este ponto será abordado juntamente com as discussões sobre incorporação de regras a banco de dados.

Regras foram inicialmente empregadas como formalismos para programação de sistemas *Rule Based*, tendo alcançado grande êxito na construção de sistemas especialistas. Regras são normalmente declaradas da seguinte forma:

```
IF    Condition
THEN Result
```

Em sistemas especialistas, *Condition* normalmente representa um casamento de padrões sobre as estruturas de conhecimento situadas na memória principal. Caso a condição seja satisfeita, uma modificação sobre o conhecimento armazenado na memória é feita, de acordo com o que está expresso em *Result*.

Em sistemas de banco de dados, o conceito de regras de produção foi adaptado considerando o imenso volume de dados situados em memória secundária. O modelo de regras utilizado neste contexto procura identificar situações de disparo das regras, evitando a necessidade da utilização de casamento de padrões<sup>5</sup>. Surge assim o conceito de bancos de dados *ativos*, correspondendo a sistemas de banco de dados que comportam mecanismos de gerenciamento de regras de produção.

Mecanismos de regras acoplados a gerenciadores de banco de dados podem ser utilizados tanto no desempenho de atividades do gerenciador como Materialização de Visões, Controle de Acessos, Dedução e Manutenção de Restrições (inerente e implícita), como também podem ser empregados em tarefas de suporte a aplicações como Manutenção de Restrições Explícitas e Monitoramento de Situações.

Com a incorporação de regras, surge a necessidade de estender as técnicas e ferramentas associadas ao projeto do banco de dados para tomar proveito desta funcionalidade. Um exemplo é a tese de Tanaka[Tan92], que incorpora as noções de evento e regras às estruturas de um modelo Entidade Relacionamentos estendido, obtendo um modelo conceitual denominado  $E^2R^2$ . A estratégia apresentada utiliza ferramentas para gerar automaticamente as regras destinadas a manter a integridade no contexto de um ambiente de banco de dados relacional.

A representação declarativa das restrições durante a modelagem conceitual também constitui um passo em direção à utilização da capacidade ativa, dado que estratégias de mapeamento de restrições para regras podem ser aplicadas, retirando das aplicações a responsabilidade de manter a consistência.

Diferentes estratégias de mapeamento de restrições podem ser aplicadas, considerando a notação de regras usadas pelo gerenciador alvo ou considerando uma notação genérica para regras, representando as principais características e mecanismos identificadas nas linguagens de regras dos sistemas ativos existentes.

---

<sup>5</sup>Algumas implementações de SGBD que incorporam regras de produção utilizam o casamento de padrões, sendo que as propostas mais recentes empregam a detecção de eventos.

A segunda abordagem apresenta algumas vantagens do ponto de vista que a tecnologia de SGBD ativos ainda não alcançou um consenso quanto a forma e expressividade das regras, tanto no modelo relacional quanto no modelo orientado a objetos, permitindo que a estratégia sirva como base para a adaptação para um ambiente específico. Esta abordagem foi empregada no trabalho de [TPC94], no qual as restrições, expressas usando um modelo semântico que estende o modelo IFO[AH87], são mapeadas em termos de uma notação para regras próxima, mas não equivalente, à notação da proposta de HiPAC[DBB+88].

## 1.5 Organização da Tese

A literatura sobre projeto de banco de dados para aplicações, como foi vista nas seções anteriores, cobre diferentes aspectos. No entanto, na maior parte das vezes não existe uma preocupação em integrar a especificação e a manutenção da consistência das aplicações ao projeto. Além disso, nos casos em que restrições são mencionadas, a principal preocupação é a manutenção de restrições de integridade estáticas.

Esta dissertação apresenta esta integração, estando centrada no problema de incorporação de restrições dinâmicas (integridade e comportamento) ao projeto de banco de dados para aplicações em sistemas de informação.

Este trabalho estende as propostas anteriores encontradas na literatura ao incorporar a modelagem de restrições dinâmicas no projeto de bancos de dados usando sistemas gerenciadores de bancos de dados orientados a objetos com capacidade ativa.

A opção por sistemas gerenciadores de banco de dados orientados a objetos é adotada pois estes incorporam o comportamento além dos aspectos estruturais das aplicações. A opção por um sistema com capacidade ativa é justificada pela facilidade de processamento e gerenciamento das regras que são de relevância fundamental para o controle das restrições.

As principais contribuições do trabalho são:

- Propor uma taxonomia de restrições na modelagem conceitual de sistemas de informação, caracterizando tanto a consistência estática quanto a consistência dinâmica das aplicações. A taxonomia serve como ponto de partida para a escolha de uma notação para a modelagem de restrições e para identificar os principais requisitos sobre a expressividade das regras de forma que estas sirvam para manter as restrições envolvidas
- Apresentar uma estratégia de projeto que leva em consideração as características comportamentais e ativas do gerenciador de banco de dados orientado a objetos
- Servir como apoio à modelagem conceitual, permitindo a representação declarativa das restrições, incluindo a captura dos aspectos dinâmicos das aplicações



- Identificar mapeamentos em termos de regras de produção para a manutenção da consistência dinâmica (Integridade dinâmica e Restrições sobre o comportamento)
- Simplificar a tarefa de manutenção de restrições, possibilitando a evolução ordenada dos requisitos das aplicações
- Permitir que aplicações como Automação de Escritórios, Sistemas de Informação Geográfica, Planejamento e Apoio a Decisão onde vários requisitos são expressos através de restrições, possam ser adequadamente modeladas e ter sua consistência assegurada

A dissertação está organizada da seguinte forma. O capítulo dois apresenta os principais conceitos sobre banco de dados ativos. O capítulo três faz uma revisão bibliográfica acerca de propostas para especificação e controle de restrições. O capítulo quatro apresenta a taxonomia proposta para classificar restrições na modelagem conceitual de sistemas de informação. O capítulo cinco apresenta os principais aspectos do projeto conceitual em camadas para bancos de dados ativos orientados a objetos, concentrando-se nos requisitos a serem observados por uma estratégia de modelagem conceitual. O capítulo seis apresenta a Linguagem de Definição de Restrições (*CDL*) ilustrando sua aplicação no projeto conceitual em camadas. O capítulo sete apresenta a estratégia de mapeamento de restrições em termos de regras e finalmente, no capítulo oito são apresentadas as conclusões e possíveis extensões ao trabalho. O apêndice traz a gramática da Linguagem de Definição de Restrições (*CDL*) proposta na dissertação.

# Capítulo 2

## Sistemas Ativos

Este capítulo discute os principais conceitos relativos a bancos de dados ativos, que são usados no mapeamento de restrições de integridade.

### 2.1 Introdução

Sistemas Gerenciadores de Banco de Dados tradicionais (SGBD) vêm sendo empregados como repositórios de dados em diversos tipos de sistemas de informação, tanto para aplicações convencionais quanto para aplicações não convencionais como sistemas de informação geográfica, controle de processos, CAD/CAM, automação de escritórios, controle de tráfego aéreo dentre outras.

O constante monitoramento dos estados individuais e seqüências de estados ocupadas pelos objetos de forma a manter restrições de integridade e controle constitui um requisito crítico em aplicações não convencionais. A responsabilidade de manter as restrições pode estar dispersa em vários níveis de abstração, tomando por base a arquitetura ANSI/SPARC:

- **Aplicações**
- **Interface entre os níveis externo e conceitual do SGBD**
- **Nível conceitual do SGBD**
- **Módulo independente mediador entre aplicações e banco de dados**

De acordo com os níveis acima descritos, restrições podem ser mantidas de diversos modos:

1. pelas aplicações acessando o banco de dados

2. por monitores que freqüentemente realizam uma operação de *polling* sobre o estado do banco de dados
3. pela camada que implementa o mapeamento entre o nível externo e o nível conceitual ANSI/SPARC
4. pela implementação do modelo conceitual do SGBD
5. por intermédio de regras de produção acopladas ao banco de dados destinadas a desempenhar o monitoramento das atividades e estados

O monitoramento, quando delegado às aplicações, constitui um empecilho ao controle centralizado das ações, além de exigir a replicação do código destinado a controlar as restrições a cada nova aplicação, aumentando o esforço de projeto. Quando feito por monitores que realizam a operação de *Polling*, há uma queda de desempenho associada ao tempo em que o sistema de banco de dados fica ocioso enquanto o *Polling* é executado.

Com relação aos mecanismos para definição e controle de visões existentes, pouco foi feito em termos de permitir novas possibilidades para formulação de restrições. Em geral, os mecanismos de visões funcionam como filtros através dos quais porções do SGBD podem ser consultadas e manipuladas por grupos de usuários, sem acrescentar semântica adicional.

Quanto à manutenção das restrições no nível conceitual, apesar das inúmeras propostas de modelos semânticos existentes, poucos foram implementados como modelos operacionais de SGBD's. A maior parte dos modelos operacionais de banco de dados oferece pouco suporte a restrições semânticas genéricas.

A quarta alternativa para o controle de restrições consiste em utilizar um banco de dados capaz de desencadear ações independentemente de requisições das aplicações definido como *Banco de Dados Ativo*. A capacidade ativa é viabilizada mediante o acoplamento de sistemas de regras de produção a SGBD's, e pela incorporação de um subsistema para gerenciamento destas regras.

O emprego de regras para aumentar a funcionalidade de SGBD's representa um tema de pesquisa em expansão[SSU91, Nav92]. A incorporação do mecanismo de regras tornando o SGBD ativo facilita a implementação de diversos mediadores[Wie92], permitindo o suporte a diversos domínios de aplicações.

Os esforços nesta área estão concentrados em quatro aspectos principais[vdVK93, ACC+93]:

- **Modelo de Regras.** Dentro desta perspectiva, grande parte das propostas adota um modelo baseado em regras E-C-A[DBM88]. São identificados os diversos tipos de eventos suportados, regras para especificar eventos compostos, definição dos tipos de consultas possíveis na avaliação da condição e ações aplicáveis.

- **Modelo de Execução.** Neste contexto, são definidos os possíveis modos de acoplamento, além do relacionamento existente entre a ativação das regras e o modelo de transação do SGBD.
- **Implementação e Otimização.** Dentre os principais trabalhos neste domínio estão as técnicas de otimização de consultas adaptadas para avaliar as condições das regras, utilização de redes de discriminação para organizar as diversas regras associadas a uma entidade[Sto92], e o emprego de marcadores em instâncias de forma a acelerar a identificação da regra a ser ativada em caso de atualização[Sto92].
- **Estratégias de Projeto.** Neste aspecto, os requisitos e características das aplicações são considerados visando encontrar formalismos e ferramentas para especificação, validação e análise do comportamento ativo necessário.

Regras de produção podem ser aplicadas para desempenhar diversos tipos de funções pertinentes às quatro categorias enumeradas a seguir:

1. **Desenvolvimento:** Não obstante o fato de regras constituírem uma primitiva de linguagem de baixo nível, é possível sua utilização para a codificação direta de diversos sistemas *event driven* e *data driven*.
2. **Suporte à Interação:** Algumas funções podem ser desempenhadas mediante o uso de regras, visando facilitar a interação dos usuários com o sistema de informação. Exemplos são funções de alerta ou notificação do usuário, inicialização de *defaults*, comunicação entre aplicações e controle do ambiente de interação
3. **Modelo de Dados:** Regras podem servir para dar suporte a restrições de integridade que não são diretamente expressas nos esquemas do modelo operacional do banco de dados, mecanismos de versões, funções de segurança e autorização.
4. **Funções do SGBD:** Funções do gerenciador podem ser implementadas com o auxílio de regras. Dentre as principais podemos destacar: materialização e controle de visões, otimização de estruturas de acesso via geração dinâmica de índices e manutenção de estatísticas.

Para alcançar a funcionalidade desejada, vários aspectos devem ser considerados no projeto de um mecanismo de controle ativo. Os principais aspectos são[DBB<sup>+</sup>88]:

- **Análise das aplicações alvo:** Neste esforço, são consideradas a natureza das tarefas<sup>1</sup>, as condições a serem monitoradas e as ações executadas. A natureza das

---

<sup>1</sup>Requisitos temporais, duração, granularidade, concorrência e cooperação.

tarefas determinará as características do modelo de execução e dos algoritmos de escalonamento: as possíveis condições terão grande impacto sobre o modelo de conhecimento e as ações executadas juntamente com os tempos requisitados para reação às situações também causarão impacto sobre o modelo de execução. Nesta etapa, será determinado o modelo de regras adequado para representar os requisitos temporais das aplicações, os eventos a serem suportados e os dados associados aos eventos e condições.

- **Especificação do Modelo de Conhecimento:** Nesta etapa, são definidas as primitivas para especificar regras, a álgebra para composição de eventos e as primitivas da linguagem para descrever as reações.
- **Especificação do Modelo de Execução:** Nesta etapa, o modelo considera as transações do usuário juntamente com gatilhos disparados. Modelos de transações são projetados de forma a garantir a corretude da execução concorrente de transações e gatilhos. Além disso, são definidos os possíveis modos de acoplamento entre a avaliação da condição e o disparo das respectivas ações.

Em sistemas relacionais, regras são implementadas por intermédio de uma camada mediadora (gerenciador de regras), requerendo estruturas e mecanismos adicionais para permitir o gerenciamento. Em sistemas orientados a objetos, três estratégias podem ser implementadas para dar suporte às regras[DPG91]:

- Pré-compilar as regras e posicioná-las em cada ponto do código onde há possibilidade de ativação
- Estender a definição de classes indicando que regra invocar sempre que um envio de mensagem é feito
- Utilizar uma camada adicional de gerenciamento de regras (mediador)

A primeira abordagem esconde a semântica das regras dentro dos métodos, dificultando a modularidade, a extensibilidade e a compreensão do comportamento ativo. A segunda abordagem, empregada por Diaz[DPG91], trata as regras de uma maneira uniforme, considerando-as como *First Class Objects* e implementa as operações de gerenciamento como métodos, sem a necessidade de prover uma camada externa de gerenciamento. A terceira alternativa emprega um módulo gerenciador de regras, definindo estruturas auxiliares e mecanismos para dar suporte ao gerenciamento de regras. Esta estratégia é empregada por Medeiros[MP91].

Devido à natureza primitiva da linguagem de regras, a codificação destinada a desempenhar o comportamento ativo torna-se bastante confusa e sujeita a erros. Uma alternativa

para superar tais dificuldades consiste em considerar as regras como uma espécie de linguagem de montagem e associar formalismos declarativos de especificação para descrever as funções. Estratégias de mapeamento são usadas para transformar as especificações em regras.

Além do suporte provido por linguagens de especificação e ferramentas de análise e validação, há também a necessidade de estender as metodologias de projeto de sistemas de banco de dados visando incorporar o comportamento ativo em etapas iniciais de projeto. A prática atual é a de projetar o esquema das aplicações independente da definição das regras, o que pode causar a ocorrência de efeitos indesejáveis durante a execução do sistema.

## 2.2 Linguagens para Descrição de Regras Ativas

Um banco de dados ativo permite o controle das aplicações mediante o uso de regras que monitoram as seqüências de estados e eventos ocorrendo no sistema, sinalizando cadeias de acontecimentos relevantes e reagindo de forma apropriada.

Dada a necessidade de relacionar eventos e estados existentes em diversos pontos do eixo temporal, é importante que uma linguagem de regras temporal seja utilizada. Segue uma descrição acerca dos principais conceitos e definições do modelo E-C-A de regras e principais extensões para tornar as regras temporais.

### 2.2.1 Regras E-C-A

Uma regra ativa está dividida em três partes: evento, condição e ação recebendo o nome de **Regra E-C-A**[DBM88]. A sintaxe intuitiva da regra é a seguinte:

```
WHEN (E)vent  
IF (C)ondition  
THEN (A)ction
```

Tendo a seguinte semântica: “Quando o evento  $E$  ocorrer, avalie a condição  $C$  e caso seja verdadeira, dispare a ação  $A$ ”. Regras funcionam como objetos podendo ser inseridas, consultadas, removidas e modificadas em uma base de regras[DBM88, MD89]. Além disso, atributos podem ser atrelados às regras servindo para determinar propriedades como o intervalo de validade temporal, estatísticas e escopo de aplicabilidade.

### 2.2.2 Suporte a Mecanismos de Eventos

Uma banco de dados ativo (BDA) precisa detectar a ocorrência de qualquer evento definido para poder iniciar a ativação das regras. O responsável por tal tarefa é o detector de eventos. Para que diversas situações reais possam ser monitoradas, uma linguagem

expressiva de definição de eventos deve ser empregada permitindo a modelagem das situações complexas. Exemplos de linguagens para expressão de eventos são as linguagens dos sistemas ODE[GJS92b], SAMOS[GGD91, GD93], COMPOSE[GJS92a] e a notação *Snoop*[CM93].

Um evento representa uma ocorrência atômica no eixo do tempo associada ao momento em que o BDA deve reagir a um acontecimento interno ou externo ao SGBD. O eixo do tempo constitui um domínio discreto representado pelo intervalo semi-aberto  $[0, 1, \dots, \infty)$ , demonstrado na Figura 2.1. O evento é representado como uma tupla  $(e_T, event - id)$ , onde  $e$  representa uma instância de um tipo básico de evento  $T$  e  $event - id$  representa a *Time-Stamp* associado ao ponto da linha do tempo no qual ocorreu o evento, identificando unicamente cada evento.

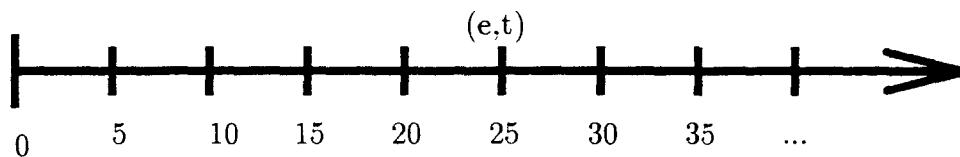


Figura 2.1: Eixo do Tempo Associado ao Sistema

Atributos são atrelados a eventos representando propriedades associadas ao sistema como o identificador da transação, o identificador do usuário que disparou a transação e o ponto no tempo representando o identificador do evento. Em sistemas orientados a objetos, eventos gerados pela invocação de métodos podem comportar como atributos os parâmetros do método invocado além de informações associadas ao estado do banco de dados existente por ocasião da ocorrência do evento.

A granularidade do tempo adotada para representar eventos pode variar em quantidade de unidades ou *chronons* no eixo do tempo. Intervalos são representados por expressões de eventos (EE), modelando fenômenos que levam uma duração finita. Um ponto absoluto no eixo do tempo deve ser identificado dentro do intervalo especificado pela EE denotando a ocorrência do evento. Para isso, modificadores de eventos como **end-of** e **begin-of** são empregados.

Eventos podem preceder, seguir ou não ter relações com outros eventos. Além disso, dois tipos de relação de ordem entre eventos podem ser considerados:

- **Ordenação Causal:** relaciona eventos de diferentes tipos (ex: o evento “o elevador partiu” deve estar precedido do evento “porta fechou”).
- **Ordenação Temporal:** relaciona eventos de diferentes tipos e eventos do mesmo tipo com base no tempo de ocorrência associado aos eventos.

Eventos podem ser lógicos (begin-of TRANSACTION) ou eventos físicos (ponto no código imediatamente anterior ao início da transação). Eventos lógicos estão em nível conceitual e são mapeados em termos de eventos físicos (detectáveis pelo SGBD ativo) pelos modificadores. Ocorrências de eventos são mapeadas pelos modificadores de acordo com a granularidade de tempo adotada pela aplicação. Os passos envolvidos na detecção de um evento são os seguintes:

- **Ocorrência do evento:** quando ocorre um evento, os parâmetros formais do evento são instanciados pelos valores dos parâmetros atuais.
- **Deteção do Evento:** quando o evento é detectado, os parâmetros são coletados pelo detector.
- **Sinalização:** o evento é sinalizado por intermédio do envio de uma mensagem para o avaliador de condições.

### 2.2.3 Tipos de Eventos

Os diversos tipos de eventos que ocorrem em sistemas ativos podem ser classificados de acordo com a seguinte taxonomia:

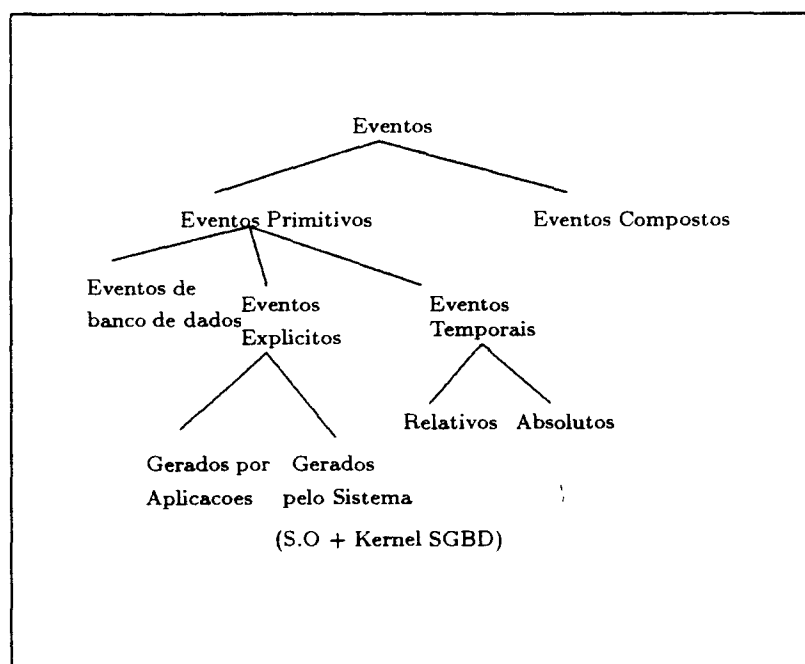


Figura 2.2: Taxonomia de Eventos em Sistemas Ativos.



**Eventos Primitivos** são registrados no sistema que é capaz de realizar a detecção. **Eventos Compostos** são formados a partir da aplicação de operadores de composição sobre eventos primitivos e/ou outros eventos compostos.

**Eventos de banco de dados** são gerados a partir das operações de manipulação definidas no modelo operacional do banco de dados (INSERT, RETRIEVE, UPDATE e MODIFY) para o caso do modelo relacional e chamada de métodos para o caso do modelo Orientado a Objetos ou por transações sobre o banco de dados.

**Eventos Explícitos** são definidos por aplicações ou pelo software de sistema (Sist. Operacional), registrados internamente ao BDA e destinados a sinalizar eventos ocorrendo externamente ao BDA que precisam ser observados pelo sistema ativo. A detecção é efetuada externamente ao BDA, juntamente com o processamento dos parâmetros, sendo posteriormente enviados ao BDA por ocasião da sinalização. Alguns exemplos são descritos a seguir:

- “Pressão da caldeira acima do valor normal”  
Parâmetros: Temperatura, Id-Caldeira  
Origem: Aplicação de controle de processos industriais
- “Queda de um servidor”  
Parâmetros: Id-Servidor  
Origem: Sistema Operacional Distribuído

**Eventos Temporais** definem pontos no eixo temporal onde eventos devem acontecer. A especificação pode ser associada a um tempo absoluto ( $\langle TimeSpec \rangle$ ), ou a um instante temporal relativo a ocorrência de um evento antecessor ( $E + \langle TimeSpec \rangle$ ), onde *TimeSpec* representa a especificação de um valor absoluto no eixo do tempo.

Os principais eventos existentes em bancos de dados orientados a objetos são descritos abaixo e podem ser enquadrados na taxonomia anteriormente apresentada:

- Eventos associados ao estado dos objetos
  - Imediatamente após a criação do objeto (supondo que construtores não são explicitamente especificados como funções membro)
  - Imediatamente antes da deleção do objeto (supondo que destrutores não são explicitamente especificados como funções membro)
  - Imediatamente antes ou depois que um objeto é atualizado, lido ou acessado através de uma função membro pública
- Eventos de execução de métodos (imediatamente antes ou depois que uma função membro é aplicada a um objeto)

- Eventos temporais
- Eventos de início, término e confirmação ou rejeição de transações
- Eventos externos (aplicações ou sistema)
- Eventos globais (modificação do esquema)

### 2.2.4 Álgebras de Eventos

Álgebras de eventos são incorporadas às linguagens de especificação de eventos para permitir a composição de eventos. Um evento composto ocorre no instante de tempo especificado pelos operadores de composição tomando por base o tempo associado à ocorrência do último evento componente.

Eventos compostos podem ser vistos como representantes de intervalos que denotam atividades compostas resultantes de uma seqüência de estados intermediários e subatividades no banco de dados.

Eventos compostos são avaliados através do *log* de eventos e representam formas simplificadas de consultas temporais. No entanto, os predicados booleanos do evento composto não referenciam objetos temporais armazenados no banco de dados, permitindo que características temporais sejam incorporadas em bancos de dados que não suportam as dimensões temporais de tempo válido e tempo de transação.

Os principais operadores encontrados em álgebras de composição como as de ODE, SAMOS, *Snoop* e COMPOSE são:

1. *Seqüência* indicando que o evento composto acontece quando os eventos que constituem a seqüência tiverem ocorrido na ordem determinada pelo operador
2. *Conjunção* indicando que o evento composto ocorre se ambos os eventos que formam a conjunção ocorrerem, independente da ordem relativa entre eles
3. *Disjunção* indicando que o evento composto ocorre se pelo menos um dos eventos da disjunção tiver acontecido
4. *Negação* indicando que o evento composto ocorre se os eventos denotados na expressão de negação não tiverem ocorrido

### 2.2.5 Regras E-C-A Temporais

Uma regra E-C-A é considerada temporal se obedecer a um ou mais dentre os critérios dispostos abaixo[PSE+94]:

1. O evento é composto e refere-se a eventos básicos ocorrendo em pontos do tempo distintos do ponto no qual a regra foi disparada.
2. O evento pertence à categoria temporal explícita (absoluto ou relativo)
3. A condição contém uma consulta temporal que não pode ser expressa em uma linguagem de consulta não temporal. Note que a linguagem não temporal pode fazer referências ao evento que causou o disparo da regra, e opera sobre um banco de dados que não armazena o histórico temporal[LS87].

Regras em um banco de dados não temporal só poderão ser consideradas temporais caso obedeam pelo menos um dentre os dois primeiros critérios anteriormente mencionados.

# Capítulo 3

## Especificação e Controle de Restrições

### 3.1 Introdução

O tratamento de restrições é um tema bastante pesquisado na área de bancos de dados. As diversas propostas apresentadas endereçam dois aspectos principais:

- Especificação da Restrições
- Controle das Restrições.

No entanto, a maior parte dos trabalhos não relaciona o tratamento das restrições com a metodologia usada para projetar o banco de dados, tornando implícito o papel das restrições na modelagem. Nestes trabalhos, em geral, elege-se um conjunto de restrições que se deseja manter, e apresenta-se uma notação para a especificação das restrições e mecanismos para que as restrições sejam mantidas.

Além de uma maior facilidade na tarefa de manter as restrições, outros benefícios de uma integração entre o tratamento das restrições e a abordagem de projeto são: manutibilidade, evolução dos requisitos e desempenho.

Este capítulo apresenta uma visão geral dos principais trabalhos onde o processamento das restrições está inserido no contexto de uma abordagem para projeto de banco de dados. A maior parte das propostas descritas utiliza como ambiente alvo um SGBD ativo, sendo que as demais reconhecem a importância do tratamento das restrições no projeto do banco de dados, mas não citam especificamente um sistema ativo como ambiente alvo. Nesta última perspectiva estão as abordagens adotadas pelo projeto *Tempora* e pela Linguagem *TROLL*.

## 3.2 *Tempora*

A abordagem promovida por *Tempora*[TLW91, LK92, Oel91, BNP+91] para o desenvolvimento de sistemas de informação procura tratar de forma uniforme aspectos inerentes ao domínio do empreendimento como estrutura, restrições, atividades, regras e exceções.

Para superar problemas como manutenção, evolução de requisitos e alinhamento entre políticas, estrutura e processos organizacionais com os sistemas informatizados, o projeto *Tempora* emprega três modelos na modelagem do domínio do problema:

- **Modelo ERT** usado para descrever as propriedades estruturais do domínio (conhecimento estrutural)
- **Modelo de Processos (PID)** usado para descrever o processamento da informação e o fluxo de informações pelos elementos do domínio (funções e controle)
- **Modelo de Regras** usado para representar restrições externas (sobre estados e comportamento) e regras de derivação de informação (restrições e dedução)

A arquitetura de *Tempora* para desenvolvimento de sistemas, além destes três modelos citados anteriormente, também suporta o nível de mapeamento e o nível de execução. Ela tem como ambiente, um banco de dados relacional com dimensão temporal, uma linguagem para programação de transações e ferramentas de apoio[LK92].

Outra característica do *Tempora* é a utilização de metamodelos formais para descrever o relacionamento existente entre as três perspectivas capturadas pelos modelos. Os metamodelos são armazenados explicitamente e usados na verificação da consistência das especificações.

### 3.2.1 Modelo ERT

O modelo ERT é um modelo construído a partir do modelo ER através da incorporação do tratamento da dimensão temporal de tempo válido. Além da incorporação do fator temporal, o modelo ERT difere de ER pois considera como relacionamento qualquer associação entre objetos<sup>1</sup> e também suporta o conceito de Objetos Complexos e as abstrações de Classificação, Agregação, Generalização e Agrupamento. Todo relacionamento em ERT é binário e é considerado como um conjunto contendo dois pares (*entidade ou valor*, *papel*), onde o *papel* expressa o modo como uma entidade ou valor está envolvida(o) no relacionamento.

O conceito de *Classe* é o conceito mais primitivo considerado em ERT. Em um esquema ERT somente classes de objetos são especificadas. O tempo é introduzido em ERT

---

<sup>1</sup>Não há a distinção entre relacionamentos entre entidades e o relacionamento de uma entidade com seus atributos.

como uma classe chamada de *Classe de Período de Tempo*. Cada conceito (Entidade ou Relacionamento) que varia no tempo recebe um marcador de tempo<sup>2</sup>, indicando o período de existência das entidades ou período de validade para relacionamentos. Para cada marcador de tempo, a granularidade é estabelecida permitindo que relações intervalares sejam computadas[All83].

Além de objetos com histórico armazenado, há também a possibilidade de definir objetos com duração instantânea (eventos), denotados através da definição do intervalo de duração como unitário.

O modelo ERT permite ainda que o usuário defina classes entidades simples e relacionamentos como derivados, sendo as instâncias determinadas dinamicamente de acordo com a necessidade. Se o componente definido como derivado tiver um marcador de tempo associado, então as regras de derivação somente são computadas no período especificado pelo marcador, senão são computadas em todos os instantes da granularidade mínima de execução do sistema.

As hierarquias formadas usando Generalização/Especialização possuem restrições associadas às subclasses definidas de acordo com o mesmo critério de especialização:

- *IS-A Parcial* estabelece que podem existir membros na classe mais geral que não pertencem a nenhuma das subclasses
- *IS-A Total* estabelece que todos os membros da classe mais geral pertencem também a algumas das subclasses
- *IS-A com Interseção*<sup>3</sup> estabelece que duas subclasses dentro do mesmo critério de especialização, podem ter instâncias em comum
- *IS-A Disjunto* estabelece que duas subclasses obtidas de acordo com o mesmo critério de especialização não podem ter instâncias em comum

As duas restrições iniciais servem para descrever relacionamentos semânticos entre classes e suas subclasses, enquanto que as duas últimas descrevem relacionamentos entre as subclasses de uma mesma classe. Outros axiomas válidos no modelo são:

- o período de validade de um relacionamento deve ser um subperíodo da interseção dos períodos de existência das duas<sup>4</sup> entidades envolvidas
- classes representando valores e relacionamentos IS-PART-OF (classes denotando composição) não podem variar no tempo

---

<sup>2</sup>“Timestamp.”

<sup>3</sup>“Overlapping.”

<sup>4</sup>o modelo somente permite relacionamentos binários.

- o relacionamento IS-A não varia no tempo

O modelo ERT permite a definição de classes de valores ou entidades complexas mediante o uso do operador de composição para representar agregações ou agrupamentos entre elementos. Os componentes de um objeto complexo comportam uma ou mais subestruturas hierarquicamente dispostas, cada qual relacionada através do relacionamento IS-PART-OF com a classe complexa. Outras restrições adicionais sobre IS-PART-OF são as restrições de dependência e exclusividade[TLW91].

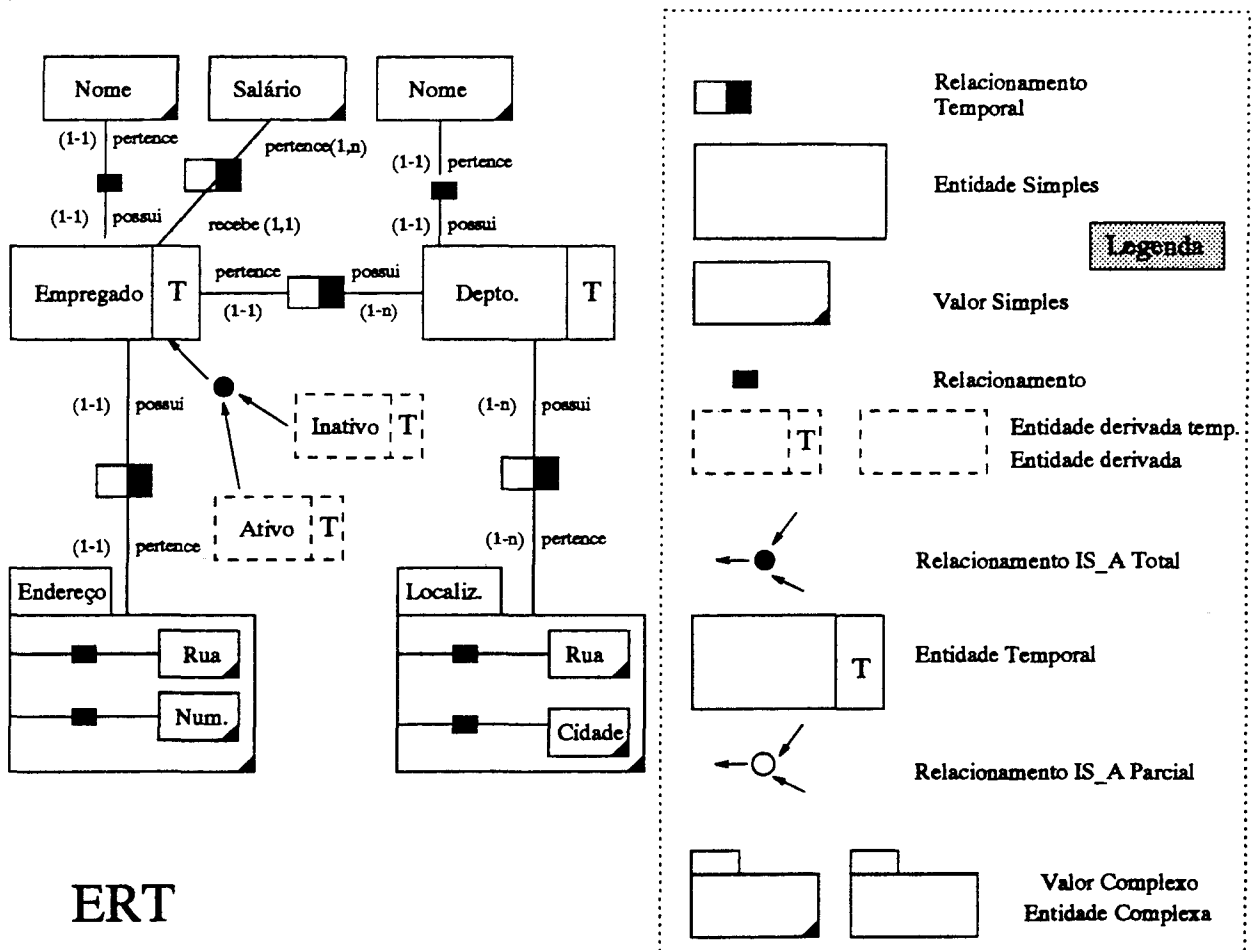


Figura 3.1: Ilustração do Modelo ERT

A Figura 3.1 mostra um exemplo de diagrama ERT. De acordo com a Figura, entidades como Empregado são marcadas indicando o interesse em registrar os períodos nos quais a entidade existe (significando que o empregado está ativo) e os períodos em que não existe (empregado inativo). Endereços de empregados podem variar no tempo assim como as localizações dos Departamentos.

### 3.2.2 Modelo de Regras

A Linguagem Externa de Regras (ERL)[BNP<sup>+</sup>91]<sup>5</sup>, permite a expressão de restrições sobre os estados e transições de estados dispostos nos esquemas ERT, regras de ação representando restrições sobre o comportamento das transações e regras de derivação de informações. As seguintes estruturas definem as alternativas no uso de ERL:

- *When* <event> *If* <condition> *Then* <consequence>
- *When* <event> *Then* <consequence>
- *If* <condition> *Then* <consequence>
- <condition>

A semântica da linguagem estabelece que, em qualquer instante da execução, se o <evento> ocorrer e a <condição> for válida então a <conseqüência> também deve ser assegurada.<sup>6</sup>

Operadores *Always*, *Sometime* denotando modalidades temporais podem ser associados a condições representando regras de derivação baseadas em predicados temporais ou restrições de integridade temporais.

Como exemplo, a informação acerca do estado ativo ou inativo da entidade *Empregado* que varia no tempo, pode ser caracterizada da seguinte forma: “se o empregado está presente no banco de dados no instante atual, então recebe a definição de ativo. Um empregado inativo será aquele que em algum momento do passado esteve presente no banco de dados, mas que agora não mais se encontra presente”. As regras expressando a derivação das informações são:

- *If* empregado(x) *Then* empregado-ativo(x)
- *If Not* empregado(x) *And Sometime-In-Past* empregado(x) *Then* empregado-inativo(x)

Outro exemplo, agora identificando uma regra de ação representando uma restrição sobre o comportamento desempenhado por uma transação é o seguinte:

- *If* empregado-ativo(x) *And Date-Is* 28/\*\*/\*\* *Then* calcular-salario(x)

Para concluir, algumas expressões em ERL denotando restrições sobre estados são:

<sup>5</sup>Em outro artigo sobre o projeto *Tempora*[LK92], a linguagem é denominada como Linguagem Conceitual de Regras (CRL)

<sup>6</sup>Embora a linguagem tenha uma semântica similar à semântica do modelo de regras E-C-A, o ambiente de execução não menciona a utilização de sistemas ativos



- *salario pertence empregado pertence departamento possui nome Where nome = "computação" >= 1000*
- *Always-In-Future salario pertence empregado > 500*

A especificação das restrições e derivações em ERL procura obedecer a dois critérios conflitantes: de um lado está a necessidade de legibilidade e inteligibilidade das sentenças, que devem ser compreendidas tanto pelos especialistas do domínio (usuários) quanto pelos projetistas do sistema; do outro, está a necessidade de representar de forma precisa e não ambígua o conhecimento. Baseado nestas questões, a notação ERL mostra-se bastante intuitiva e próxima à linguagem natural.

### 3.3 $T_{ROLL}$

A metodologia de modelagem conceitual  $T_{ROLL}$ [JSH91, JSHS91]<sup>7</sup> utiliza uma notação formal orientada a objetos para a descrição de aspectos estáticos e dinâmicos de sistemas de informação.  $T_{ROLL}$  é baseada em quatro sublinguagens integradas em uma notação uniforme: sublinguagem para descrição de dados, assertivas de primeira ordem, assertivas temporais e sublinguagem para a descrição de processos.

O princípio fundamental de  $T_{ROLL}$  é o de considerar o universo de discurso como uma coleção de objetos interagentes. Os aspectos estáticos e dinâmicos são encapsulados no interior dos objetos e são expressos através de descrições formais. As descrições de objetos representam os blocos de construção das especificações e são denominadas de *Templates*. Uma descrição associada a um identificador dá origem a uma instância de descrição enquanto que uma instância de descrição associada a um estado corrente dá origem a um objeto.

Um template pode ter as seguintes seções:

**template** *nome*

**data types** assinaturas de tipos de dados importadas

**attributes** nome e tipo dos atributos

**events** nome dos eventos e parâmetros

**derivation** regras para derivação de atributos e eventos

**constraints** restrições de integridade sobre estados

**valuation** efeitos de eventos sobre atributos

**behavior**

**permissions** condições para que eventos ocorram

**obligations** requisitos de completude para ciclos comportamentais

---

<sup>7</sup>Textual Representation of an Object Logic Language

**patterns** transações e scripts

**end template** *nome* <sub>3</sub>

As seções de tipos de dados, atributos e eventos constituem a assinatura de uma definição de template e formam um alfabeto que pode ser usado pelas sublinguagens do template. O exemplo de definição seguinte ilustra a especificação de um template LIVRO, onde os tipos básicos *nat*, *string* e os tipos abstratos *PESSOA* e *list(PESSOA)* são importados participando do alfabeto do template.

**template** *Empregado*

**data types** *nat*, *string*, —PESSOA—, *list*(—PESSOA—);

**attributes** *Dpto*: *list*(—Empregado—);

*Editora*: *string*;

*Ano*: *nat*;

**derived** *Idade*: *nat*;

**events** *birth* *contratado*;

**constraints** *Salario* > 1500;

**derivation** *Num-Dependentes* = **length**(*Dependentes*);

**end template** *Empregado*

O evento *contratado* está associado a criação de objetos descritos pelo template *Empregado*. *Num-Dependentes* é um atributo derivado, cuja regra de derivação está na seção **derivation** do template. Parâmetros podem ser associados a eventos, representando, por exemplo, valores para inicialização durante a construção (*birth*) de objetos ou ainda, podem ser usados para permitir que dados sejam trocados na comunicação entre objetos.

Na seção **constraints**, restrições de integridade estáticas e dinâmicas podem ser especificadas, usando respectivamente as sublinguagens para definição de assertivas de primeira ordem e assertivas temporais. As declarações seguintes ilustram a definição de um objeto e da classe *Empregado* usando o template.

**object** João

**template** *Empregado*

**end object**

**object class** *Empregado*

**identification**

**data types** *string*, —PESSOA—;

*Nome*: *string*;

*Salario*: *float*;

**template** *Empregado*;

**end object class**

O paradigma  $T_{ROLL}$  permite ainda a especificação de relacionamentos de interação entre objetos comunicantes[JHS92] e especificação da lógica de processos. Um exemplo completo contendo a modelagem conceitual dos aspectos estáticos e dinâmicos de uma aplicação de controle de caixas registradoras pode ser encontrado no manual da linguagem[JS91].

A abordagem  $T_{ROLL}$  procura integrar os aspectos estáticos e dinâmicos usando uma notação formal orientada a objetos empregando conceitos advindos de áreas como modelagem semântica, especificação algébrica e especificação de sistemas reativos. Uma desvantagem encontrada está no fato de não existir uma notação gráfica associada ao paradigma.

### 3.4 Controle de Restrições via Programação por Contrato

O trabalho de Geppert e Dittrich[GD94b] descreve uma abordagem para especificação e controle de restrições usando o gerenciador de banco de dados ativo orientado a objetos SAMOS. A estratégia é baseada na técnica de Programação por Contrato[Mey88].

Nesta abordagem, précondições são fórmulas sobre o estado do receptor e sobre os argumentos da mensagem. Póscondições restringem o estado do objeto receptor da mensagem após a execução do método e são denotadas por fórmulas relacionando o estado final, o estado inicial e os parâmetros de saída. Invariantes são restrições de integridade que podem ser definidas sobre todos os objetos individualmente, sobre a coleção de objetos que define a extensão de uma classe e sobre o banco de dados.

Précondições e póscondições definem um contrato, onde as précondições estabelecem critérios a serem obedecidos pelo emissor da mensagem e póscondições estabelecem o que o receptor está obrigado a produzir.

A consistência é mantida em três níveis:

- Objetos
- Extensões das classes
- Banco de Dados e Aplicações

#### 3.4.1 Especificação de Restrições

Restrições definindo um contrato são especificadas de acordo com a seguinte notação:

- **Require** *nome* formula [reparo] {, *nome* formula [reparo]}; (précondição)
- **Ensure** *nome* formula [reparo] {, *nome* formula [reparo]}; (póscondição)

O [reparo] indica uma ação de reparo a ser especificada opcionalmente usando a linguagem de manipulação de dados do modelo. A definição das cláusulas contratuais é feita durante a definição da assinatura dos métodos que possuem restrições associadas.

Além de précondições e póscondições, invariantes são especificadas juntamente com a definição do comportamento e das propriedades estruturais das classes.

A seguinte notação é usada para descrever uma invariante:

- **Invariant** *nome* formula [reparo] {, *nome* formula [reparo]};

Fórmulas são escritas em forma conjuntiva sendo cada conjunção uma disjunção de predicados. As variáveis livres permitidas nas fórmulas representam instâncias das classes. Métodos também são permitidos em fórmulas desde que não executem modificação de valores.

Também é possível referenciar métodos e variáveis representando instâncias de classes distintas permitindo que restrições interobjetos (intraclasse e interclasse) sejam especificadas. Segue um exemplo de invariante relacionando dois objetos de uma mesma classe e a respectiva restrição especificada na notação:

- “Empregados de uma firma não podem ter dependentes em comum”
- **Invariant** *Dependentes* for all Empregados X, Empregados Y: not (X.dependentes in Y.dependentes);

O controle das restrições ocorre por ocasião do envio de uma mensagem *m* definida na classe do objeto *o*. A précondição de *m* é verificada e se nenhuma das fórmulas for falsa, o método é executado. Caso a précondição seja falsa, se nenhuma ação corretiva estiver associada à précondição, o método e conseqüentemente a transação são abortados. O mesmo procedimento de controle é aplicado às invariantes e às póscondições.

- Restrições sobre a extensão (todas as instâncias consideradas coletivamente) são formuladas como parte da invariante da extensão
- Restrições definindo critérios para que um objeto seja membro de uma classe são formuladas como précondições de construtores (métodos de inserção de objetos)
- Restrições definindo critérios para que um objeto possa ser removido de uma classe são formuladas como póscondições dos destruidores (métodos de remoção de objetos)

Um exemplo de restrição sobre a extensão de uma classe é apresentado a seguir. A restrição pode ser declarada como précondição do construtor que insere objetos na classe Empregado.

- “Todo empregado deve ser brasileiro nato”

- **Require** *Nato* new Empregado.nacionalidade == "brasileiro"

Por último, restrições sobre as aplicações são definidas considerando o banco de dados como um grande objeto complexo tendo as possíveis extensões como propriedades estruturais e as aplicações como métodos. Neste nível de abstração o controle das restrições possui as seguintes características:

- Restrições sobre os estados do banco de dados que ainda não foram formuladas como invariantes das extensões das classes são definidas como invariantes do objeto "banco de dados"
- Restrições que devem ser válidas para que as aplicações possam ser executadas são definidas como précondições sobre as transações
- Restrições que devem ser válidas ao final da execução dos programas são definidas como póscondições sobre as transações

### 3.4.2 Mapeamento

Hipóteses adotadas pela estratégia de mapeamento:

1. Cada restrição definida é traduzida para um regra de produção correspondente
2. As regras de produção ficam localizadas internamente às classes
3. Todo método que realiza modificações sobre o estado do receptor é definido como uma transação (subtransação do enviante da mensagem)

Para uma précondição sobre um método, o evento da regra de produção correspondente à restrição é composto pela seqüência formada pelo evento associado ao envio da mensagem e pelo evento BOT gerado pelo modificador Begin-Of aplicado ao método sobre o qual a restrição é definida. Para que o BOT da transação correta seja identificado, o evento de transação possui como parâmetro o nome da transação que executa o método, definido pela concatenação dos nomes da classe e do método. Neste exato momento, a transação que executa o corpo do método recém chamado ainda não começou e qualquer anormalidade causaria o ABORT da transação que enviou a mensagem. O evento associado ao envio da mensagem é necessário para que o receptor correto da mensagem seja determinado.

A condição é traduzida para uma consulta expressa na DML do modelo, que computa o valor negado da restrição. A ação pode ser um reparo ou o ABORT. O modo de acoplamento é imediato.

Para póscondições, o evento da regra de produção correspondente à restrição é composto pela seqüência formada pelo evento associado ao envio da mensagem e pelo evento

EOT gerado pelo modificador End-Of aplicado ao método sobre o qual a restrição é definida.

Regras de produção associadas a invariantes são habilitadas por cadeias de eventos associados a modificações de valores dos objetos. O modo de acoplamento definido para invariantes é o modo “deferred”, já que durante a execução do método as invariantes podem estar parcialmente inconsistentes mas ao final da execução readquirirem a consistência.

O tratamento de restrições de integridade dinâmicas não é feito diretamente pelo mecanismo ativo de SAMOS pelo fato deste não dar suporte a *Deltas* (diferenças entre estados antigos e estados correntes de objetos).

### 3.5 Abordagem de Urban et al

Na abordagem proposta por Urban[UD89, UD90, UKN92], regras de produção são geradas automaticamente a partir das restrições e armazenadas como extensões às operações das classes, tornando transparentes os detalhes de verificação de restrições e disparo das regras.

A modelagem dos aspectos estruturais das aplicações é feita utilizando um modelo semântico e é complementada pela especificação declarativa de restrições associadas ao modelo e restrições externas pertencentes à aplicação.

O processo de *Análise de Restrições*[UD90] é aplicado possibilitando a representação declarativa das restrições e a identificação das restrições que afetam as operações. Uma análise é feita para determinar como cada restrição pode ser satisfeita. A ferramenta CONTEXT é responsável pelo processo que é feito como descrito a seguir:

“O usuário da ferramenta CONTEXT (projetista de B.D ou aplicações) identifica as operações de manipulação que deseja analisar. Após a escolha da operação, cada restrição afetada pela operação é identificada, levantando-se as possíveis violações e ações de reparo. A medida que o usuário escolhe as reações, regras de produção são geradas, ficando atreladas às operações.”

O usuário interage com o banco de dados por intermédio de transações de alto nível que executam as operações de modificação de estados. No instante da confirmação das transações, a integridade é verificada. A Figura 3.2 ilustra o processo.

Todas as operações de alteração de estado são registradas num Log que funciona como interface entre as transações e o subsistema de controle de integridade. Quando a transação está prestes a ser confirmada, o subsistema de integridade examina o log e, para cada operação registrada, as regras de integridade associadas são verificadas.

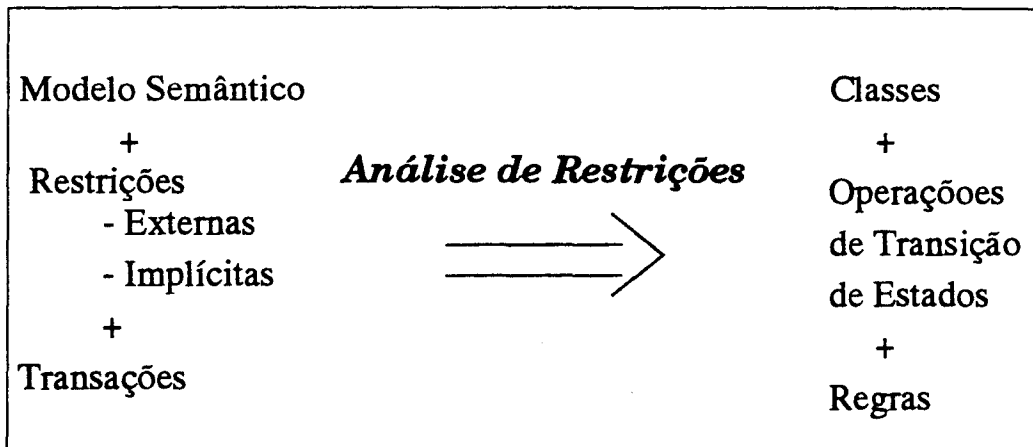


Figura 3.2: Geração de Regras Usando Análise de Restrições

# Capítulo 4

## Taxonomia de Restrições na Modelagem Conceitual

### 4.1 Introdução

Restrições definem critérios de consistência que devem ser satisfeitos em qualquer instante da execução de aplicações sobre o banco de dados. Restrições podem estar associadas aos estados do banco de dados ou às ações executadas pelas aplicações sobre o banco de dados.

O tratamento de restrições no projeto de aplicações para sistemas de informação, merece uma atenção especial pelo fato de que requisitos de informação representados sob a forma de restrições normalmente constituem uma fonte de informações semânticas acerca da empresa. Elas estão sujeitas a constante evolução, caracterizando a dinâmica dos sistemas de informação. A representação de forma explícita simplifica a tarefa de controle de restrições e possibilita que requisitos possam ser modificados e/ou incorporados sem afetar de maneira drástica o sistema. Restrições podem ser genericamente classificadas de acordo com três critérios ortogonais:

- Quanto à forma de representação da restrição
- Quanto ao aspecto da aplicação sendo focado
- Quanto à forma de manutenção da consistência

Restrições podem ser representadas declarativamente de forma gráfica ou textual, ou podem ser representadas proceduralmente pela incorporação de testes de consistência em programas que acessam o banco de dados.

A especificação declarativa de restrições apresenta inúmeras vantagens, quando comparada à representação procedural em programas. Dentre as principais vantagens, podemos observar:



- Permite que o projetista concentre-se nas políticas do domínio sendo modelado, identificando o que deve ser restringido, ao invés de preocupar-se com a questão de como será feito.
- Retira das aplicações a responsabilidade de prover boa parte do código associado à manutenção de restrições, simplificando os programas.
- Torna possível a modificação das restrições sem que seja necessária a modificação do esquema do banco de dados e/ou dos programas de aplicação.
- Serve como repositório de conhecimento declarativo para diversas atividades como otimização, dedução e segurança.

O uso de linguagens declarativas (como, por exemplo, a Lógica) permite ainda que cada restrição possa ser especificada independentemente como uma fórmula lógica e que a consistência do conjunto de restrições possa ser verificada usando os critérios de consistência de modelos na linguagem utilizada.

*Restrições sobre Estados* ocorrem quando a restrição é dada sobre os valores que compõem o(s) estado(s) do banco de dados. *Restrições sobre Comportamento* são restrições estabelecidas sobre as ações que atuam sobre os dados, sendo neste caso, o controle realizado para garantir a manutenção de critérios acerca de como, quando e em que condições as ações podem ser executadas.

Restrições sobre estados estão estreitamente associadas aos modelos de dados usados durante o projeto do banco de dados. Modelos de dados semânticos normalmente permitem a representação de um maior número de restrições enquanto que os modelos operacionais dos gerenciadores apresentam uma capacidade limitada para a expressão de restrições de integridade. Para os modelos orientado a objetos e relacional, várias classificações para restrições foram elaboradas [JMSS90, And92].

Este capítulo apresenta uma taxonomia de restrições na modelagem conceitual de banco de dados e aplicações para S.I (vide Figura 4.1). Esta classificação organiza as restrições em termos de duas classes principais: *Restrições sobre estados* e *Restrições sobre comportamento*. A partir desta taxonomia é possível identificar o aspecto do sistema associado à restrição, analisar possíveis acontecimentos que podem ocasionar a quebra da restrição e propor linguagens para expressão e mecanismos de detecção.

A taxonomia pode servir como base para a identificação de novas taxonomias de acordo com a estratégia de modelagem usada para descrever as aplicações do S.I. Por exemplo, o trabalho de [And92] apresenta uma classificação para restrições estáticas dentro do modelo orientado a objetos, correspondendo a um ramo (restrições estáticas) de uma possível instância (modelo orientado a objetos) da taxonomia aqui descrita.

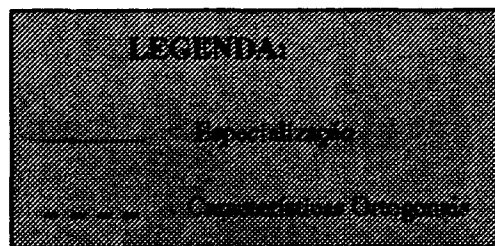
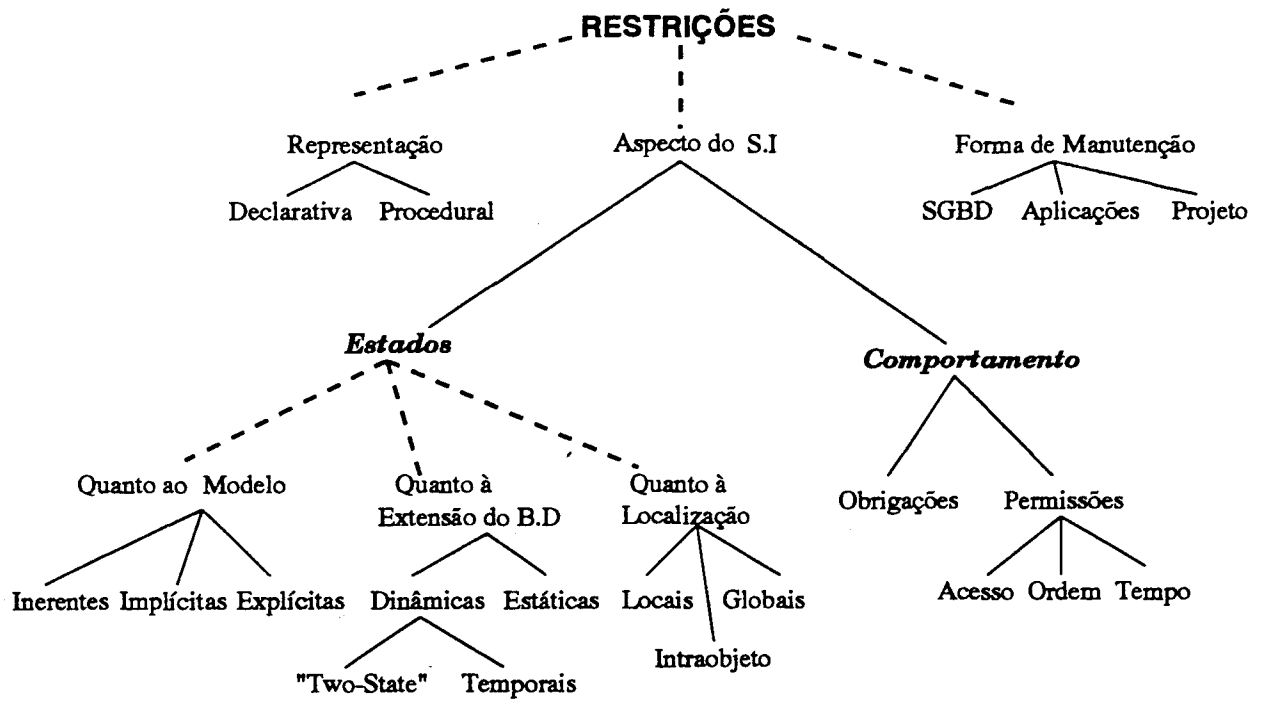


Figura 4.1: Taxonomia de Restrições

## 4.2 Restrições Sobre Estados

A manutenção da integridade dos estados constitui um fator determinante para o sucesso de um Sistema de Banco de Dados e está relacionada com a capacidade de permitir que o banco de dados somente armazene dados que representem estados válidos do mundo modelado e assuma seqüências de estados consistentes com o referido mundo[JMSS90].

Diversos fatores podem ocasionar a quebra da integridade em um banco de dados. Podemos agrupá-los em dois grupos principais. São eles:

- Fatores associados a inclusão, retirada, permanência<sup>1</sup> e alteração de informações no banco de dados. Como exemplos podemos identificar:
  - Falhas de dispositivos
  - Atualizações
  - Acessos não autorizados
  - Acessos concorrentes não seriais
  - Falhas durante o processamento de transações
- Fatores associados ao projeto do banco de dados. Dentre os principais podemos citar:
  - Incapacidade do esquema em representar os estados legítimos do universo modelado.
  - Esquemas em formas normais inapropriadas<sup>2</sup>

Idealmente, a integridade do banco de dados deveria estar atrelada à corretude de uma forma biunívoca, ou seja, ao garantir-se a integridade do banco de dados estar-se-ia também garantindo a corretude e vice-versa. Contudo, as dificuldades inerentes à prova de corretude de um esquema juntamente com a incapacidade de associar sensores que atestem a validade no mundo real dos dados armazenados, das modificações efetuadas e dos estados resultantes, torna a corretude total praticamente inviável de ser alcançada em sistemas de banco de dados[JMSS90].

Ao invés disso, procura-se assegurar a consistência dos dados, ou seja, a condição de que o banco de dados assuma somente estados possíveis do mundo real, sendo portanto uma pré-condição para a corretude ou fator determinante da corretude parcial. Os critérios de consistência descrevem as políticas do universo modelado e disciplinam as ações que

---

<sup>1</sup>Em sistemas de banco de dados destinados a aplicações com características temporais, o andamento do relógio do sistema pode ocasionar a quebra da integridade

<sup>2</sup>Aspecto relacionado com o projeto de banco de dados relacional

produzem ou modificam os dados dentro deste contexto[CCF82]. Neste sentido, a palavra *integridade*, quando empregada, estará sempre associada à noção de consistência.

O problema de manter a integridade é particionado em dois subconjuntos: – *Integridade Estática e Integridade Dinâmica*. Podemos definir o problema de manter a integridade estática como sendo o de assegurar que os dados armazenados em um banco de dados constituam um estado válido do mundo representado em uma determinada instância da coordenada do tempo. Da mesma maneira, podemos considerar o problema da manutenção da integridade dinâmica como sendo o de garantir que o banco de dados somente atravesse seqüências de estados possíveis no mundo modelado durante o espaço de execução.

Cada um dos subproblemas necessita de formalismos lingüísticos apropriados e mecanismos de controle específicos. Os formalismos lingüísticos são usados para expressar as restrições, enquanto que os mecanismos de controle são responsáveis pela manutenção das restrições durante a execução do sistema.

Várias notações podem ser empregadas para descrever restrições sobre estados[FN86]. Dentre as principais podemos destacar, dentre outras, linguagens visuais, linguagem de definição de dados de um modelo de dados, predicados da lógica. A manutenção das restrições, por sua vez, pode ser efetuada pelas aplicações, pelo uso de transações seguras[SS89] ou pelo gerente de integridade de um SGBD ativo.

As restrições de integridade sobre estados podem ser caracterizadas de acordo com três aspectos ortogonais[EN89]:

- Quanto ao relacionamento com o modelo do banco de dados:
  - **Implícitas:** São representadas nos esquemas produzidos utilizando a linguagem de definição de dados (LDD) do modelo do banco de dados. Cada modelo inclui um conjunto diferente de restrições implícitas que podem ser diretamente representadas nos esquemas. Tal conjunto também pode variar de acordo com a implementação do sistema gerenciador de banco de dados (SGBD). Como exemplos podemos listar:
    1. Restrições de Chaves nos modelos E/R e relacional
    2. Cardinalidade no modelo E/R
    3. Integridade Referencial em algumas implementações do modelo relacional
    4. Restrições sobre domínios
  - **Inerentes:** São restrições impostas pelos próprios axiomas que compõem o modelo de dados, não necessitando ser especificadas nos esquemas visto que são válidas por definição. Dentre as principais podemos destacar:
    1. Atomicidade de Valor nos atributos de um esquema relacional

2. Restrições sobre relacionamentos no modelo E/R<sup>3</sup>
  3. Unicidade de tuplas dentro de uma relação
- **Explícitas:** São restrições não capturáveis nos esquemas originados usando a LDD do modelo de dados<sup>4</sup>. São normalmente expressas usando variantes da lógica de primeira ordem no caso das restrições estáticas ou usando lógica temporal para o caso das restrições dinâmicas. Requerem estruturas ou mecanismos auxiliares por parte do SGBD ou das aplicações de forma a garantir sua manutenção. Como exemplo podemos citar: “todo empregado que é ou foi gerente jamais ganhará abaixo de um determinado valor”, “todo funcionário já foi estagiário”, “o salário de qualquer funcionário é menor que o salário do gerente do seu departamento”.
- Quanto ao aspecto do mundo que está sujeito à restrição:
    - **Estáticas:** Expressam os critérios de consistência sobre o estado do banco de dados. São normalmente expressas mediante o uso de variantes da Lógica de Primeira Ordem[End72]
    - **Dinâmicas:** Expressam os critérios de consistência acerca das seqüências de estados admissíveis. A Lógica Temporal[RU71] representa o formalismo de especificação mais difundido
  - Quanto à localidade das restrições:
    - **Intraobjeto:** Estão associadas aos valores de dados de uma mesma instância.
    - **Locais:** Estão associadas às instâncias (extensão) das entidades (relação ou classe).
    - **Globais:** Estão associadas ao banco de dados.

### 4.3 Restrições sobre Comportamento

Para que a especificação de um banco de dados para sistema de informação seja completa é necessário que as restrições sobre comportamento sejam levadas em consideração. No campo do projeto de banco de dados, cuida-se em geral apenas dos aspectos estruturais (estrutura dos objetos e relacionamentos entre eles), sendo que o comportamento e restrições comportamentais ainda continuam pouco explorados.

---

<sup>3</sup>Toda instância de um tipo relacionamento n-ário  $R$  relaciona exatamente uma entidade de cada tipo entidade participante de  $R$

<sup>4</sup>Se for considerada uma lógica suficientemente expressiva como a linguagem de definição de um banco de dados, teremos todas as restrições inerente ou implicitamente definidas

Com a utilização de banco de dados orientados a objetos, tanto os dados quanto os métodos e transações que representam o comportamento são modelados de forma uniforme e mantidos pelo SGBD. Isto aumenta ainda mais a importância da especificação das restrições sobre o comportamento e a sua manutenção.

Restrições comportamentais podem versar explicitamente sobre o tempo ou sobre a evolução das ações<sup>5</sup> através do tempo. O conhecimento temporal expresso nas restrições comportamentais pode refletir a duração das ações, ordenação cronológica relativa e absoluta, momentos de início e término das ações, condições para que sejam ativadas e ações que devem acontecer em algum momento do ciclo de vida dos objetos em interação.

Restrições comportamentais são normalmente especificadas usando cláusulas de *PRÉ* / *PÓS* condições ou formalmente representadas como variantes de lógicas situacionais. Uma sintaxe geral para expressão de restrições comportamentais deve relacionar ações que ocorreram no passado e ações acontecendo no presente com ações que possivelmente ocorrerão no futuro, definindo o comportamento temporal das ações das aplicações sobre o banco de dados.

Restrições comportamentais podem ser de dois tipos: **Permissões** e **Obrigações**. Para a compreensão dos conceitos envolvidos, a seguinte especificação de classe é elaborada em uma notação hipotética:

```
class PERSON
{

    @atributos
    nome: String;
    com_fome: Boolean;

    @acoes
    construct entrar(n:String)
    {
        nome <-- n;
        com_fome <-- nao;
    };

    comer(f:{"peixe" | "carne"})
    {
        se (f=="peixe")
            entao com_fome <-- sim
    }
}
```

---

<sup>5</sup>A palavra ação pode estar representando um método pertencente a um objeto, um método pertencente à extensão de uma classe (métodos da classe) ou ainda uma transação

```
        senao com_fome <-- nao
    };

    trabalhar()
    { com_fome <-- sim};

    destruct sair();

@resticoes_de_permissao
    PRE comer(): com_fome==sim;
    PRE sair(): com_fome==nao;

@restricoes_de_obrigacao
    POS trabalhar() => EVENTUALMENTE  comer();
}
```

Os atributos **nome** e **com-fome** definem a estrutura dos objetos da classe, enquanto que os métodos **entrar**, **comer**, **trabalhar**, **sair** definem o comportamento inerente aos objetos da classe. A seguir está a descrição do comportamento dos objetos.

Depois que uma pessoa é criada, o seu nome é registrado e ela ainda não está com fome. Após trabalhar, ela fica com fome e tem como opções peixe ou carne para comer. Após comer peixe, ela ainda não se dá por satisfeita e somente depois de comer carne ela fica satisfeita. A pessoa pode comer somente se estiver com fome e pode sair de cena somente se não estiver com fome. Caso a pessoa esteja com fome, em algum momento do tempo ela deve se alimentar.

### 4.3.1 Permissões

Restrições de permissão definem condições sobre a execução de ações. Estas restrições são estabelecidas usando cláusulas de PRÉ (BEFORE) e PÓS (AFTER) condições e estão relacionadas à premissa de execução da ação, a ordem de execução e ao momento de execução da ação, sendo respectivamente denominadas de restrições comportamentais de *Acesso*, *Ordem* e *Tempo*.

#### Restrições Comportamentais de Permissão de Acesso

Restrições comportamentais de permissão de acesso estão associadas às condições que devem ser satisfeitas pelo banco de dados para que a ação correspondente possa ser ativada. Em relação ao acesso, somente as restrições estabelecidas como PRÉ condição têm signi-

ficado apropriado. No exemplo apresentado anteriormente, o método *comer()* só poderá ser ativado se a condição “com-fome==sim” for satisfeita pelo banco de dados.

### Restrições Comportamentais de Permissão associadas à Ordem

Restrições comportamentais de permissão associadas à ordem dos acontecimentos indicam quais ações devem ter sido ativada(s) anteriormente e em qual seqüência para que uma outra ação possa ser ativada. Restrições de ordem podem ser estabelecidas em termos de PRÉ e PÓS condições. Como exemplo de restrição estabelecida como précondição podemos destacar: “Um método M3 somente será ativado após a ativação dos métodos M1 e M2, não importando a ordem relativa entre M1 e M2”. Um exemplo de restrição estabelecida como póscondição é a seguinte: “O método M1 ao final de sua execução deve ativar o método M2”.

### Restrições Comportamentais de Permissão associadas ao Tempo

Restrições comportamentais de permissão associadas ao tempo indicam o momento absoluto ou relativo para que uma determinada ação possa ocorrer. A condição normalmente indica précondições sobre o momento da ativação de uma determinada ação. Como exemplos podemos citar:

- O método M1 só pode ser ativado às 08 horas de todos os sábados
- O método M2 só pode ser ativado no dia 20

As restrições indicam os momentos absolutos do tempo nos quais são permitidas as ativações dos métodos M1 e M2.

#### 4.3.2 Obrigações

Obrigações estabelecem condições de completude para ciclos de vida de objetos, ou seja, eventos que devem acontecer se uma determinada condição for válida[JSH91].

Os requisitos acerca dos ciclos de vida dos objetos da aplicação são representados pelas restrições comportamentais de obrigação. Antes de serem destruídos os objetos, as obrigações devem ser cumpridas para garantir a consistência comportamental. No exemplo apresentado na página 50, após o objeto desempenhar o comportamento *trabalhar*, em algum momento do futuro o objeto terá de desempenhar o comportamento *comer*, antes de ser destruído.

Outro exemplo de restrição comportamental indicando obrigação é o seguinte:

- O método M3 deverá ser ativado 30 minutos após a ativação de M2



É importante que se diferencie as restrições comportamentais de obrigação cujo momento preciso da ocorrência da ação está definido, das restrições comportamentais de obrigação cujo instante futuro da ocorrência da ação não está determinado (Eventually). As primeiras, descrevem um critério preciso acerca do comportamento a ser desempenhado em seqüência pelo objeto, enquanto que as últimas indicam um comportamento que em algum momento será desempenhado, não estabelecendo exatamente o momento preciso em que ocorrerá.

### 4.3.3 Comportamento X Restrições Sobre Comportamento

Em diversos trabalhos que consideram a modelagem do comportamento ou controle, é bastante comum o emprego de estruturas no modelo dinâmico para incorporar restrições ao comportamento sem no entanto fazer a distinção entre comportamento e restrições comportamentais. Isto pode ser observado, por exemplo, no modelo dinâmico de OMT, onde *Condições de Guarda* sobre as transições de estados são incorporadas, representando um mecanismo limitado para expressar restrições comportamentais.

É importante que a distinção seja feita:

*O comportamento ou controle é a porção dinâmica que representa as decisões acerca de ações que mudam o valor dos objetos e invocam funções. São normalmente representadas mediante Diagramas de Transição de Estados, indicando qual ação é disparada quando o sistema estando em um dado estado recebe um evento. A Figura 4.2 ilustra tal ponto de vista:*

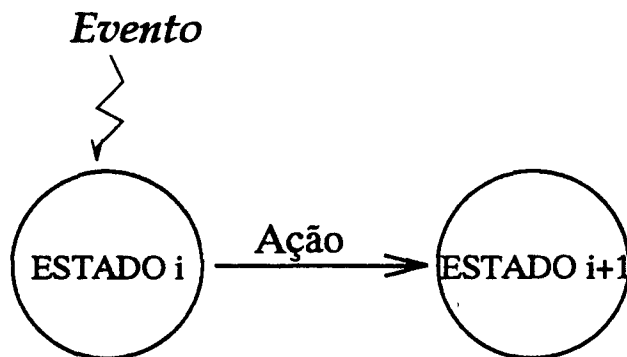


Figura 4.2: Diagrama de Transição de Estado

*Restrições sobre comportamento expressam condições para que a ação seja disparada. A condição pode estar baseada em estados ou comportamentos passados no sistema.*

## 4.4 Relacionamentos Intervalares Definindo Restrições

Outro tipo de restrição (presente em sistemas de representação de conhecimento) é o conhecimento acerca do relacionamento existente entre a duração de atividades[EdO94, All83].

Considerando-se uma representação discreta do eixo do tempo associado ao sistema, atividades são representadas por eventos que denotam fatos de duração instantânea ocorrendo no sistema (representam um ponto no eixo do tempo). As atividades ocorrem durante intervalos de dois ou mais pontos (eventos) da granularidade mínima e portanto podem representar acontecimentos de duração finita no universo modelado.

Boa parte das ações executadas sobre o banco de dados está sujeita a restrições que representam relacionamentos de precedência ou sucessão. Restrições comportamentais de ordem e de tempo são restrições que servem para expressar conhecimento temporal relativo ou absoluto representando precedência comportamental acerca das ações sendo executadas. O conhecimento temporal sobre o relacionamento entre os intervalos de duração das atividades também pode ser empregado para descrever restrições comportamentais. Como exemplos de relacionamentos intervalares, podemos destacar:

- “Uma ação representando o início de uma atividade não poderá ser executada caso exista outra atividade em execução”
- “Uma atividade A deverá ocorrer como subatividade da atividade B”

Restrições de duração são representadas através de operadores como STARTS, EQUAL, OVERLAP, END, MEET e DURING de uma álgebra intervalar[All83].

A Figura 4.3 apresenta possíveis relacionamentos entre ações representadas por intervalos de acordo com Allen[All83].

## 4.5 Restrições de Transição

Grande parte dos trabalhos associados à manutenção de restrições dinâmicas utiliza a terminologia *Restrições de Transição*, para representar restrições sobre a transição de estados no banco de dados. Uma restrição de transição pode ser especificada de duas formas:

- Como uma cláusula de précondição sobre a execução da ação (Restrição Comportamental)
- Como uma expressão em lógica temporal usando os operadores PREVIOUS ou NEXT para relacionar dois estados sucessivos (Restrição de Integridade Dinâmica “Two-State”)

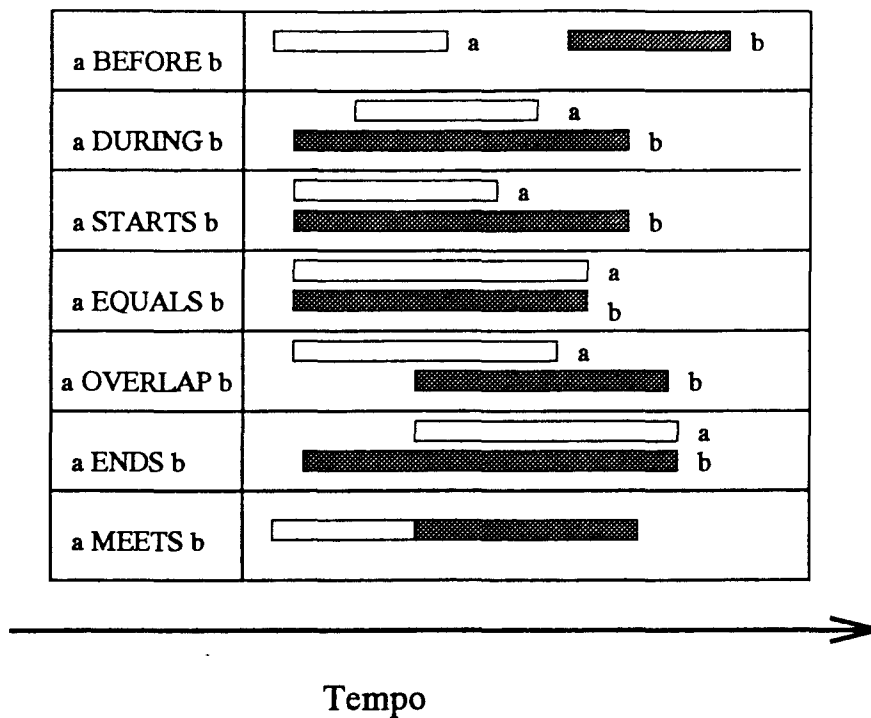


Figura 4.3: Relacionamentos Intervalares

Neste tipo de restrição, tanto o aspecto comportamental quanto o aspecto estático da modelagem podem ser considerados na expressão das restrições, sendo a escolha da forma de representação normalmente influenciada pelos mecanismos disponíveis no ambiente de execução.

## Capítulo 5

# Projeto Conceitual de Banco de Dados em Camadas

Este capítulo apresenta uma proposta de projeto conceitual de bancos de dados em camadas, de forma a incorporar o tratamento de restrições de integridade.

### 5.1 Introdução

Embora existam notações e técnicas estabelecidas tanto para o projeto conceitual de banco de dados quanto para o projeto de funções, ainda há uma carência de abordagens integradas para o projeto de sistemas de banco de dados.

A modelagem conceitual de sistemas sobre o banco de dados idealmente deveria integrar duas tarefas básicas[SS87, Saa91, JSH91, JS91]:

- **Modelagem Estrutural** que pode ser vista como a modelagem conceitual dos dados e restrições de integridade estáticas, tradicionalmente feita usando modelos semânticos
- **Modelagem Dinâmica** que corresponde a modelagem das restrições dinâmicas (integridade dinâmica e restrições sobre o comportamento), transações e o comportamento

Nesta dissertação, supõe-se uma abordagem baseada em camadas[CCF82, FN86, CS88] para a etapa de projeto conceitual de sistemas de banco de dados orientado a objetos com capacidade ativa. A elaboração do modelo conceitual procede em camadas, cada qual descrevendo aspectos específicos do banco de dados, levando em consideração o aspecto dinâmico das aplicações.

A descrição de cada camada é consistente com camadas inferiores sendo que a ordem natural de elaboração da descrição consiste em projetar as classes de objetos existentes em

aplicações, as restrições de integridade estáticas sobre o estado dos objetos, restrições de integridade estáticas sobre as extensões das classes, restrições comportamentais definindo o ciclo de vida dos objetos, restrições de integridade dinâmicas sobre os estados dos objetos e extensões das classes e finalmente, restrições comportamentais sobre métodos e transações.

Dentre as principais vantagens do desenvolvimento em camadas podemos destacar:

- Melhor controle dos aspectos restritivos que definem políticas das organizações sendo modeladas
- Modelagem incremental dos diferentes aspectos do sistema
- Especificação declarativa e formal das restrições, possibilitando o uso de ferramentas para obtenção automática de gatilhos a partir das descrições.
- Maior modularidade simplificando a manutenção
- Integração dos aspectos estáticos e dinâmicos

Adotando o enfoque proposto por Saake[Saa91], quatro níveis são usados para descrever o modelo conceitual de acordo com a Figura 5.1: nível de tipos de dados, nível de objetos, nível de restrições e nível de aplicações.

As regiões sombreadas da figura representam aspectos da modelagem conceitual cuja implementação ou controle podem ser beneficiados pelo uso de regras de produção. No nível de aplicações, aspectos do controle das aplicações como manutenção de estatísticas, inicialização de parâmetros globais e políticas de segurança[BP85] podem ser mapeados para regras.

Restrições estáticas no nível de objetos; restrições de comportamento e restrições de integridade dinâmica no nível de restrições também são mantidas usando regras. Tanto para os aspectos de controle, quanto para as restrições o modelo E-C-A é adequado:

```
ON    event_expression
IF    condition
THEN  action
```

A expressão de eventos descreve acontecimentos (eventos externos e eventos do banco de dados) que devem ser monitorados no decorrer da execução do sistema, a condição representa uma consulta sobre o banco de dados e a ação representa uma operação suportada pelo modelo do banco de dados.

A camada inferior (tipos de dados) descreve as estruturas de dados que representam os tipos de dados básicos e complexos utilizados pela segunda camada (camada de objetos). A especificação dos tipos abstratos de dados que dão suporte às camadas superiores não será tratada visto que o trabalho partirá de um modelo de dados, dos tipos básicos presentes

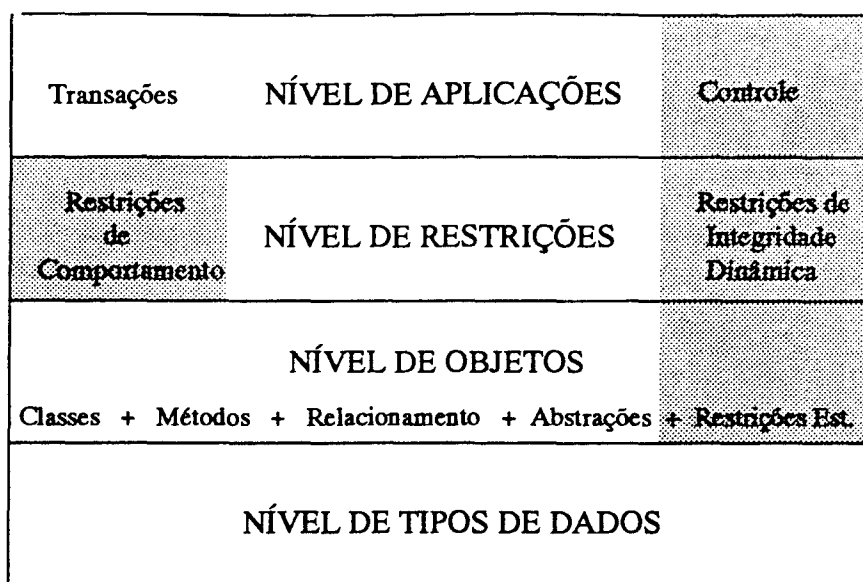


Figura 5.1: Camadas Para a Descrição do Modelo Conceitual

e tipos abstratos que podem ser projetados usando os construtores de tipos providos pelo modelo.

Para uma especificação apropriada da camada de objetos, algumas primitivas de modelagem são essenciais de acordo com Saake[Saa91]:

1. A noção de *Entidades ou Objetos*, descritos por intermédio de propriedades (atributos e métodos).
2. A noção de *Tipos de Entidades ou Objetos*, representando objetos que possuem as mesmas propriedades.
3. O conceito de *Identidade de Objetos*[KC86], identificando unicamente cada objeto independentemente do valor de suas propriedades.
4. Estruturas de *Classes*, associadas aos tipos de objetos, agrupando as instâncias de objetos do referido tipo.
5. *Operações de Abstração*, servindo para formar classes a partir de outras classes e descrever relacionamentos e associações entre classes.

A camada de objetos utiliza tipos básicos e construídos juntamente com operações de abstração dentre as quais Agregação, Associação, Generalização e Classificação. A interface de métodos pertencente a cada Classe também é descrita. Além disso, uma linguagem para especificação de restrições estáticas é empregada, completando a descrição dos estados consistentes.

A terceira camada (nível de restrições) descreve a evolução temporal dos objetos persistentes. Restrições dinâmicas são especificadas de forma a determinar seqüências válidas de estados (integridade dinâmica) além de ciclos comportamentais válidos para os objetos da aplicação (restrições de comportamento).

Como exemplo de restrições dinâmicas, podemos observar:

- “Antes de receber o primeiro salário, um funcionário precisa ser contratado por algum departamento.”
- “Um funcionário que inicia um projeto deve eventualmente terminá-lo”
- “O salário do funcionário jamais pode diminuir”

As duas primeiras representam restrições comportamentais acerca da ordem dos acontecimentos no sistema. A terceira representa uma restrição dinâmica de integridade acerca do valor do salário dos funcionários durante o decorrer do tempo. Neste nível de especificação, os principais formalismos empregados são: – lógica temporal[RU71, Ser80], lógica situacional de templates[EJDS92] Redes de Petri[Rei85] e Autômatos[LS87].

Na estratégia adotada por Saake, esta terceira camada é dividida em dois módulos complementares: módulo de evolução e módulo de ações. O módulo de evolução corresponde à definição de restrições de integridade dinâmica adotada na dissertação, enquanto que o módulo de ações corresponde à descrição de *PRÉ* e *PÓS* condições sobre as ações destinadas a garantir a consistência comportamental.

A quarta camada representa o nível onde ocorrerá a modelagem dos processos e do controle global do sistema sobre o banco de dados. Este nível de especificação conceitual pode ser feito usando técnicas de análise de sistemas e ferramentas de engenharia de software. Os requisitos de controle global podem ser traduzidos em termos de regras E-C-A, simplificando o trabalho de programação de aspectos como controle de acessos, controle de estatísticas, inicialização de defaults, etc. Este nível não será considerado nesta dissertação.

Para descrever os dois primeiros níveis a dissertação adotará o modelo de objetos da metodologia OMT proposto por Rumbaugh et al[RBP+91] e o subconjunto de regras estáticas da Linguagem de Definição de Restrições (*CDL*) apresentada nesta dissertação. Para a terceira camada, a porção dinâmica de *CDL* é empregada para capturar a especificação de restrições sobre o comportamento e restrições de integridade dinâmica.

A metodologia OMT foi escolhida por satisfazer os requisitos impostos à camada de objetos pela estratégia de projeto em camadas. Estes requisitos são: relacionamentos em nível semântico, abordagem orientada a objetos e notação gráfica simplificada. Além disso, esta metodologia é amplamente utilizada no projeto de sistemas de informação.

## 5.2 Descrição da Estratégia Proposta

Esta dissertação segue a linha dos trabalhos de Tanaka[Tan92] e Grotehen[GD94c], propondo uma extensão para o projeto de sistemas de banco de dados para sistemas de informação utilizando mecanismos de banco de dados ativos. Tanaka adota um modelo E/R estendido como modelo conceitual e o modelo operacional utilizado é o modelo relacional. Ainda naquele trabalho, uma notação para descrição de restrições de integridade é elaborada para complementar a descrição do esquema conceitual e uma ferramenta é projetada permitindo o mapeamento de restrições para regras.

Nesta dissertação, o modelo orientado a objetos é adotado como modelo operacional do gerenciador ativo. Devido a falta de padronização das diversas propostas em orientação a objetos, sugere-se a adoção de um modelo orientado a objetos que suporte as características básicas apresentadas em [BDK92]. A extensão é feita a nível de projeto para permitir uma utilização efetiva do modelo ativo como ferramenta para manter tanto restrições de integridade quanto restrições sobre o comportamento. O mapeamento de restrições para regras será feito de forma manual.

De acordo com a Figura 5.2, a etapa de Projeto Conceitual é destinada à elaboração de um modelo conceitual mediante a técnica de projeto conceitual em camadas descrita anteriormente.

A etapa de projeto conceitual inicia com a modelagem dos requisitos estruturais em termos de esquemas gerados usando o modelo de objetos da metodologia OMT, juntamente com restrições de integridade estáticas expressas usando a linguagem de definição de restrições. Em seguida, as restrições dinâmicas (integridade e comportamento) são expressas usando a variante da linguagem destinada à descrição dos aspectos dinâmicos.

A modelagem das transações, embora represente um aspecto fundamental na representação da dinâmica da aplicação, não será considerada neste trabalho.

A etapa de Mapeamento (projeto lógico) identifica os passos gerais para a obtenção do esquema lógico de regras do banco de dados destinadas à manutenção das restrições. Neste nível será apresentada uma estratégia manual. Contudo, dada a natureza formal das notações usadas durante a modelagem conceitual, é possível implementar ferramentas para automação de passos desta etapa, gerando automaticamente as regras para diversos sistemas alvo.



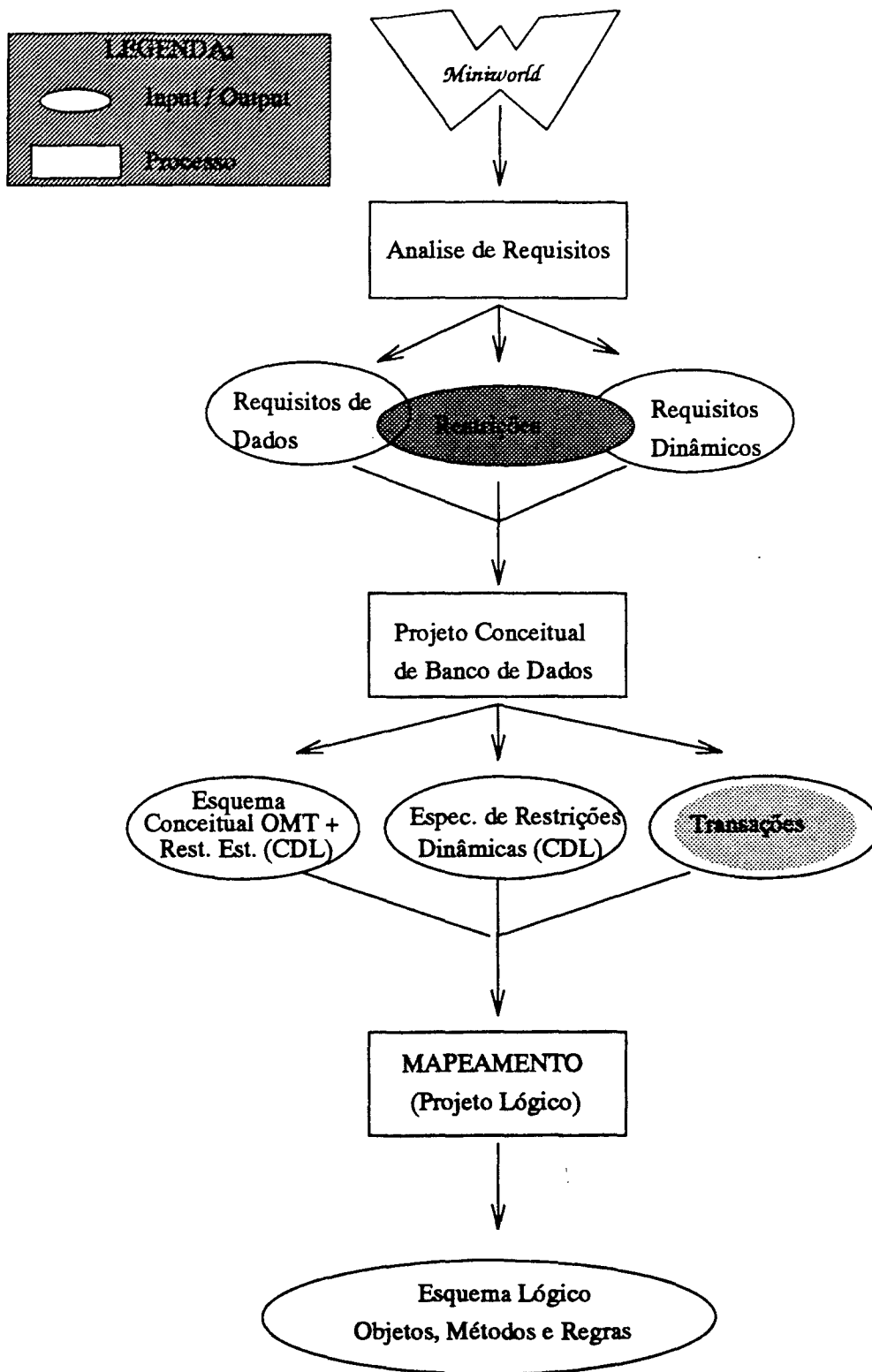


Figura 5.2: Estratégia de Projeto Usando BDOO Ativo

# Capítulo 6

## Linguagem de Definição de Restrições (CDL)

Este capítulo apresenta a Linguagem de Definição de Restrições (*CDL*) a ser usada durante o projeto conceitual em camadas descrito anteriormente. Esta linguagem cobre todos os aspectos da taxonomia proposta.

Os aspectos estáticos da aplicação constituindo o nível de objetos são modelados usando o modelo de objetos de OMT[RBP<sup>+</sup>91] e o subconjunto da notação *CDL* destinado à descrição das restrições estáticas de integridade.

Restrições dinâmicas (integridade e comportamento) são especificadas em *CDL* compondo o nível de restrições da abordagem em camadas.

### 6.1 Introdução

Para especificar restrições, há três considerações importantes a serem feitas:

- Problemas associados ao relativismo semântico
- Requisitos para a manutenção de restrições
- Escolha da notação

*Relativismo semântico* se refere ao fato de que determinadas restrições ocorrendo no domínio do problema podem ser especificadas tanto em termos de estados quanto de comportamento. Por exemplo a restrição “Um funcionário que foi demitido não pode ser readmitido” pode ser expressa das seguintes formas:

- através de uma restrição dinâmica de integridade que consulta o histórico da classe *Empregados*

- através de uma restrição comportamental que considera a seqüência de métodos *demitir()*, *admitir()* aplicados à mesma instância

Na maioria das notações para especificação de restrições, o projetista do banco de dados é forçado a seguir uma única alternativa dada a pouca flexibilidade existente no caso dinâmico, que normalmente permite somente a descrição de restrições dinâmicas sobre estados. No caso da CDL, o projetista pode optar pela forma que mais se adequa às suas necessidades.

*Requisitos para a manutenção de restrições* são os fatores que devem ser suportados pelo ambiente para que as restrições possam ser mantidas. Os principais fatores são:

1. *Dimensão temporal*: em sistemas de banco de dados temporais, regras podem ser acopladas tendo como condição uma consulta sobre o histórico do banco de dados. Um exemplo desta abordagem é o ambiente de OSAM/T[SC92].
2. *Estados estendidos do banco de dados*: em algumas abordagens para tratamento de restrições de integridade dinâmicas, o esquema do banco de dados é estendido para incorporar ao estado corrente do banco de dados a informação sobre o passado necessária para o controle das restrições dinâmicas, evitando o armazenamento do histórico. A proposta de Lipeck e Saake[LS87] e a proposta de Chomicki[Cho92] seguem este caminho.
3. *Histórico de eventos*: restrições também podem ser mantidas consultado o histórico de eventos de um SGBD ativo. Dependendo da expressividade da álgebra de eventos, algumas restrições dinâmicas podem ser mantidas.
4. *Calendários*: para que restrições envolvendo datas e intervalos de tempo sejam mantidas, sistemas calêndricos[CSS93] devem ser suportados, permitindo que regras e consultas sejam formuladas, envolvendo predicados como [ON prazo-fatal], onde prazo-fatal representa uma expressão do tipo: “terceiro dia útil do mês de novembro”.

Para efeito das restrições tratadas nesta dissertação, supõe-se a utilização de um ambiente alvo constituído por um banco de dados ativo com dimensão temporal. Além disso, o ambiente alvo apresenta um mecanismo para gerenciamento do histórico dos eventos permitindo que eventos complexos sejam formulados e supõe-se a existência de um mecanismo de calendário. Outros detalhes do sistema alvo são apresentados no próximo capítulo.

O terceiro aspecto da especificação de restrições está associado à escolha da notação, que deve ser expressiva, formal e intuitiva, permitindo uma flexibilidade na representação do domínio e sendo de fácil utilização.

## 6.2 Linguagem de Definição de Restrições (CDL)

A notação descrita nesta seção especifica restrições de integridade sobre estados (estáticas e dinâmicas) e restrições comportamentais. A notação serve de apoio à modelagem conceitual orientada a objetos e está fundamentada sobre conceitos de lógica de primeira ordem, lógica temporal com modalidades do passado e lógica situacional.

A notação foi elaborada com base na notação proposta por Ceri[CW90], que foi estendida em Andrade[And92] para suportar restrições estáticas em ambientes de banco de dados orientados a objetos. A extensão proposta nesta dissertação visa o suporte às restrições dinâmicas, ignoradas nas propostas anteriores. Para isso, modalidades da lógica temporal associadas ao passado foram acrescentadas, além da noção de restrições comportamentais expressas através de *PRÉ*, *PÓS* condições sobre a ocorrência de eventos.

As restrições expressas em CDL são:

- Restrições de integridade estáticas ou dinâmicas sobre dados expressas através de predicados sobre estados
- Restrições comportamentais de permissão de acesso, expressas através de *PRÉ* ou *PÓS* condições para a ocorrência de eventos, onde a condição é um predicado sobre estados
- Restrições comportamentais de permissão de tempo, formuladas através de *PRÉ* condições para a ocorrência de eventos, onde a condição é um predicado sobre o estado do relógio do sistema
- Restrições comportamentais de permissão de ordem, formuladas através de *PRÉ* condições para a ocorrência de eventos, onde a condição é um predicado denotando expressões de eventos sobre um *Log* de eventos (vide discussão no capítulo seguinte)
- Restrições comportamentais de obrigação, expressas através de *PÓS* condições para a ocorrência de eventos, onde a condição é um predicado indicando acontecimentos futuros

Estas restrições podem ser descritas através das duas primeiras regras gramaticais da linguagem *CDL* expressa utilizando a Forma Backus-Naur. As demais regras da gramática estão definidas no apêndice.

1. < restrição > ::= (< predicado >)
  - | **Pre** (< evento >) (< predicado >)
  - | **Pos** (< evento >) (< predicado >)
  - | **Pre** (< evento >) **Clock-Is** < comp-atributo > (< pred-clock >)
  - | **Pre** (< evento >) (< pred-ordem >)

| **Pos** (< evento >) (< pred-obriga >)

2. < pred-obriga > ::= **Eventually** (< evento >)  
| **At** (< pred-clock >) (< evento >)

*Eventos* são envios de mensagens, início/fim de transações ou eventos externos em geral (por exemplo, interrupção do sistema).

*Pred-clock* é um predicado sobre o estado do clock interno do sistema.

*Pred-ordem* é um predicado que avalia ordens de eventos.

As restrições são especificadas usando os elementos do esquema OMT (objetos, classes, papéis, relacionamentos e operações), tipos de dados fundamentais e eventos pertencentes ao domínio da aplicação. As principais construções suportadas pela notação são:

1. Operadores indicando modalidades temporais **Always Since** e **Sometime Since**
2. Operadores lógico **Implies**
3. Quantificadores **Exists, For-all**
4. Operadores de agregação **Sum, Min, Max, Card**
5. Operadores situacionais **Pre, Pos**
6. Operadores aritméticos +, -, /, \*
7. Operadores sobre conjuntos **Contains, Is-in, Is-Equal**
8. Operador indicativo do estado anterior **Previous**
9. Operadores booleanos **Or, And, Not**
10. Comparadores de identidade de objetos ==, !=
11. Comparadores de atributos =, <, >, <=, >=, <>
12. Operadores sobre eventos  $\wedge, \vee, \neg$ , **Before**
13. Operador indicativo de data e hora de um evento **TimeStamp**
14. Predicado associado ao calendário **Clock-Is**

É importante que se diferencie a semântica dos operadores situacionais **Pre**, **Pos** dos operadores sobre eventos. Estes últimos referem-se a eventos atômicos ocorridos ou não no passado enquanto que os operadores situacionais representam uma situação corrente da execução do sistema que possui uma restrição associada.

A notação de *CDL* encontra-se simplificada com relação as possíveis combinações de quantificadores existentes em uma lógica de primeira ordem temporal. Além disso, somente as construções associadas a restrições de acesso permitem a combinação de operadores temporais e situacionais. A simplificação reflete a pouca expressividade das linguagens de consultas temporais existentes na literatura, para as quais são traduzidas as restrições.

A seguir, um exemplo de aplicação é descrito e são mostrados os passos gerais para a concepção do modelo conceitual considerando a especificação das restrições de acordo com a estratégia de especificação em camadas.

### 6.3 Estudo de Caso

Nesta seção, um estudo de caso será feito ilustrando a aplicação da estratégia de especificação em camadas e a notação proposta. O exemplo pertence ao domínio de automação de escritórios.

A modelagem conceitual da aplicação é feita da seguinte forma:

- Descrição da camada de objetos: modelagem do esquema conceitual (modelo de objetos e restrições de integridade estáticas)
- Descrição da camada de restrições dinâmicas

A estratégia proposta nesta dissertação sugere o uso do modelo de objetos de OMT para a descrição do esquema da aplicação. A escolha do modelo de objetos de OMT decorre do fato deste observar os requisitos identificados no capítulo sobre projeto conceitual em camadas, impostos para a abordagem usada para descrever a camada de objetos. Outras notações capazes de descrever adequadamente os aspectos estruturais são: linguagem  $T_{ROLL}$  e o modelo ERC+.

Segue uma narrativa sobre o problema considerado:

Uma agência de prestação de serviços mantém um sistema de cadastro sobre pessoas disponíveis para trabalhar em empresas clientes no desenvolvimento de determinadas atividades. Cada pessoa cadastrada como empregado passa por períodos ativos, nos quais está prestando serviços a uma determinada empresa e períodos inativos, nos quais está à espera de um trabalho. Para que um empregado possa trabalhar (ser alocado a uma atividade) é necessário que ele passe por um período de treinamento dentro da agência não inferior a um ano, contado a partir da data em que foi incluído como empregado. Um empregado possui habilidades que são usadas na execução de uma atividade, podendo ser alocado para desempenhá-la caso possua todas as habilidades requisitadas pela atividade. Determinadas habilidades exigem uma seqüência de etapas de treinamento. Os empregados da agência são remunerados por hora de serviço prestado, sendo que o valor pago pela hora jamais poderá decrescer. A agência adota a política de somente manter empregados dentro da faixa etária de (25-65) anos.

## 6.4 Descrição do Nível de Objetos

A Figura 6.1 apresenta o esquema do exemplo, expresso usando a notação de OMT. As associações (links) representadas no modelo identificam relacionamentos entre objetos e servem para destacar que a informação não está subordinada a uma única classe. A escolha de como a associação será representada fica adiada para o projeto lógico.

As associações podem formar uma classe que pode ter atributos e operações próprias. Isto é útil quando os links precisam participar em associações com outros objetos ou quando estiverem sujeitos a operações. O exemplo demonstrado encontra-se bastante simplificado, não refletindo toda a funcionalidade de OMT.

Um *Papel* é um atributo derivado cujo valor é um conjunto de objetos relacionados. Ele é atrelado a uma extremidade da associação e permite que a associação seja percorrida sem que se mencione a associação.

Na metodologia OMT, o modelo de objetos é empregado para descrever os aspectos estruturais das aplicações. Dentre as características apresentadas pelo modelo podemos destacar:

- Permite um projeto conceitual independente de ambiente alvo
- Utiliza uma notação gráfica intuitiva usada tanto para o projeto conceitual quanto para o projeto lógico

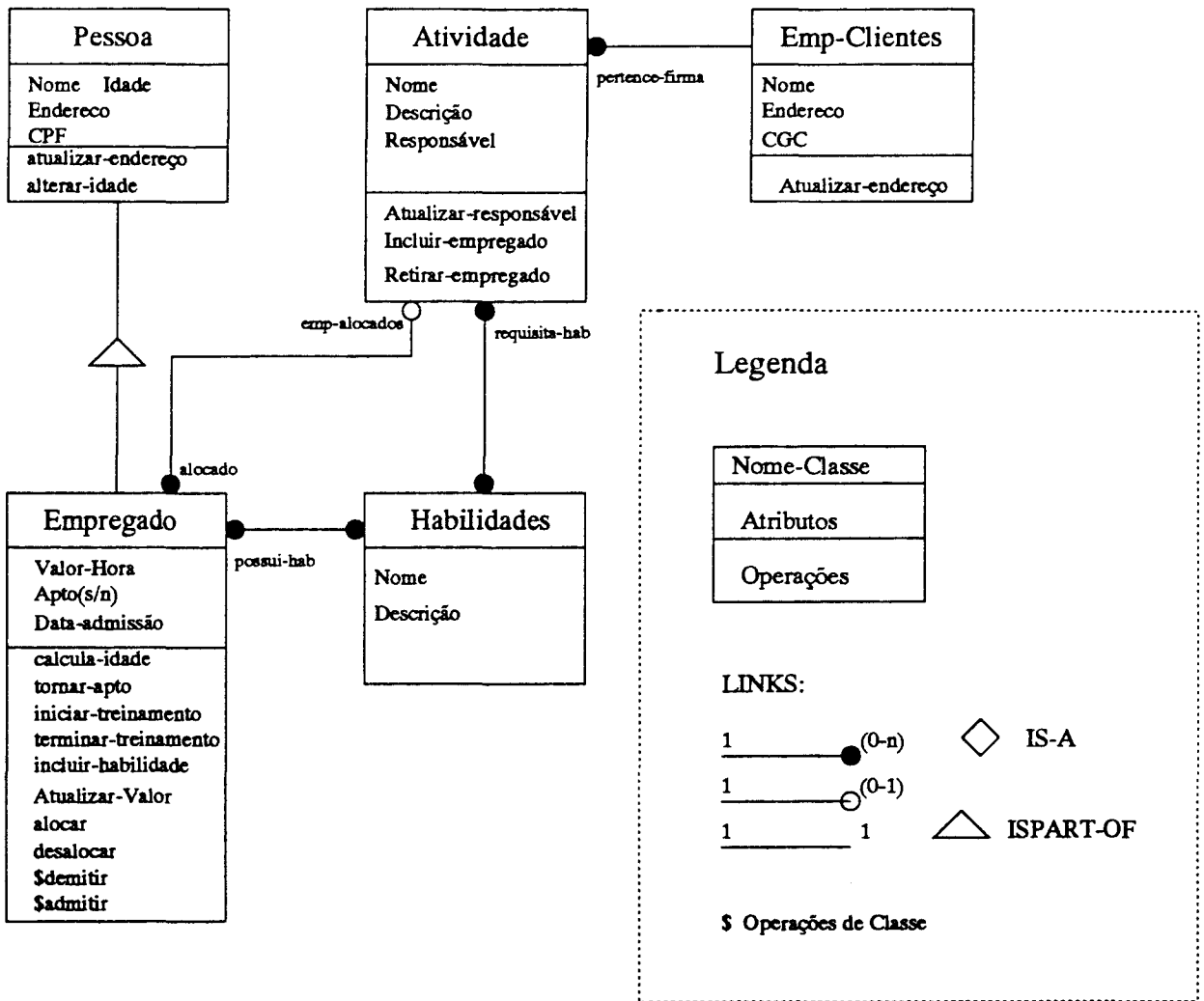


Figura 6.1: Exemplo Expresso Usando Diagrama de Objetos de OMT



- Os relacionamentos interobjetos são promovidos a nível semântico de classe, ao invés de serem escondidos como ponteiros internamente aos objetos

Uma das deficiências apresentadas na proposta de OMT está associada à ausência de uma notação precisa para a definição das restrições. As restrições são informalmente declaradas entre chaves em linguagem natural e estão dispersas ao longo dos três modelos usados para descrever as aplicações, impedindo uma abordagem integrada e uniforme para o controle das restrições.

Além disso, não há uma continuidade no tratamento destas restrições durante a etapa de mapeamento do modelo conceitual em termos de um ambiente de banco de dados alvo. A metodologia OMT não permite especificação gráfica de restrições em geral, nem uma notação para tal. Por este motivo, restrições precisam ser representadas proceduralmente pelos métodos da implementação, sendo que tal abordagem para transformação das restrições em métodos não é tratada satisfatoriamente pela metodologia. O uso da linguagem CDL complementa, portanto, uma lacuna nesta metodologia.

#### 6.4.1 Especificação de Restrições de Integridade Estáticas

Algumas das restrições estáticas ocorrendo no domínio do problema e as respectivas sentenças em *CDL* estão ilustradas a seguir. De acordo com a metodologia em camadas aqui proposta, estas restrições são especificadas no próprio nível de objetos.

- “Todo empregado alocado a uma atividade deve possuir todas as habilidades requisitadas pela atividade”  
(**For-all e In Empregados, For-all a In Atividades** : (e.alocado == a **Implies Is-equal** (e.possui-hab , a.requisita-hab)))
- “Todo empregado deve estar na faixa etária de 25 a 65 anos”  
(**For-all e In Empregados**: (e.idade >= 25 **And** e.idade <= 65))
- “Todo empregado só pode desempenhar no máximo uma atividade durante o mesmo intervalo de tempo”  
(**For-all e In Empregados**: (**Card** (e.alocado) <= 1 ))

### 6.5 Descrição do Nível de Restrições – Restrições Dinâmicas e de Comportamento

No nível de restrições as possíveis evoluções dos objetos através do tempo são caracterizadas pelas restrições dinâmicas. *CDL* permite tanto a descrição de restrições dinâmicas

intraobjeto, relacionando valores de um mesmo objeto em diferentes estados do banco de dados quanto a descrição de restrições dinâmicas relacionando objetos distintos.

Restrições comportamentais sobre a extensão das classes são estabelecidas sobre as operações de classe (precedidas pelo símbolo \$ no esquema OMT).

Algumas das restrições encontradas no exemplo são:

- “O valor pago por hora de trabalho de um empregado jamais poderá decrescer”  
(**For-all** e **In** *Empregados* : (**Always** e.Valor-Hora  $\geq$  **Previous** e.Valor-Hora))
- “Para que um empregado possa ser alocado a um projeto é necessário que ele passe por um período de treinamento de no mínimo um ano, contado a partir da data em que foi incluído como empregado”  
PRE (alocar  $\rightarrow$  e **In** *Empregados*)(**Clock-Is**  $\geq$  **Timestamp** ( \$admitir  $\rightarrow$  e **In** *Empregados*) + (00/12/00/00/00/00) )
- “Determinadas habilidades exigem seqüências de etapas de treinamento”  
Por exemplo, um empregado só pode ser considerado apto a ser Pesquisador após haver passado pelas etapas sucessivas de Graduado, Mestre e Doutor, nesta ordem.  
PRE (incluir-habilidade(Pesquisador)  $\rightarrow$  e **In** *Empregados*)  
(( incluir-habilidade(Graduado)  $\rightarrow$  e **In** *Empregados* **Before**  
incluir-habilidade(Mestre)  $\rightarrow$  e **In** *Empregados* ) **Before**  
incluir-habilidade(Doutor)  $\rightarrow$  e **In** *Empregados* )
- “Para que um empregado esteja apto para se tornar o responsável por alguma atividade, é necessário que ele tenha pelo menos três anos como empregado da agência”  
PRE ( tornar-apto  $\rightarrow$  e **In** *Empregados*) ( **Clock-Is**  $\geq$  **Timestamp** (\$admitir  $\rightarrow$  e **In** *Empregados*) + (00/00/03/00/00/00) )
- “Quando um empregado inicia um treinamento, em algum momento do futuro deve concluí-lo”  
POS ( iniciar-treinamento  $\rightarrow$  e **In** *Empregados*) **Eventually** (terminar-treinamento  $\rightarrow$  e **In** *Empregados*)
- “A política da agência é manter empregados na faixa de 25 a 65 anos”. Esta restrição foi anteriormente especificada através de uma restrição estática sobre estados. Sua verificação pode ser reforçada pela seguinte restrição de acesso:  
POS ( \$admitir  $\rightarrow$  e **In** *Empregados*) (**For-all** e **In** *Empregados*: ( e.idade  $\geq$  25 **And** e.idade  $\leq$  65))

# Capítulo 7

## Mapeamento de Restrições em CDL

Este capítulo define os requisitos do ambiente ativo alvo do mapeamento, especifica o modelo E-C-A de regras adotado para o ambiente e apresenta as heurísticas de mapeamento usadas para mapear restrições em *CDL* para regras no modelo E-C-A especificado.

Algumas das restrições existentes no estudo de caso apresentado no capítulo anterior são mapeadas para o modelo E-C-A, ilustrando a aplicabilidade das heurísticas propostas.

### 7.1 Ambiente Ativo Alvo

[MA92] mostrou que restrições dinâmicas de dois estados podem ser mapeadas em regras em sistemas ativos orientados a objetos. No entanto, a verificação de restrições dinâmicas gerais exige um sistema não apenas ativo, mas estruturas adicionais de controle. Para permitir a manutenção das restrições em CDL, é necessário adicionar às facilidades ativas:

- um registro de eventos (*Log*)
- facilidades de armazenamento de estados históricos do banco de dados (ou seja, uma dimensão temporal)

Determinadas restrições temporais exigem a verificação de sequências de estados de bancos de dados. Por este motivo, sua manutenção só pode ser realizada se o sistema tiver acesso ao histórico dos diferentes estados. Desta forma, qualquer sistema ativo que gerencie restrições dinâmicas gerais deve obrigatoriamente incluir a dimensão temporal.

O outro aspecto a ser considerado é o gerenciamento de restrições comportamentais, que são baseadas na composição de eventos. Ocorrências de eventos são denotadas por pares (*Evento básico, Identificador*) e podem comportar os seguintes atributos:

- Transaction-ID
- Identificador (Time-Stamp)

- User-Id
- Identificador da classe envolvida
- Identificador do objeto manipulado
- Atributos inerentes ao evento

Como eventos básicos o ambiente deve suportar eventos externos tais como “clock ticks”, eventos abstratos (gerados pelo núcleo do SGBD ou por entidades externas ao banco de dados e são sinalizados explicitamente), início (Begin-Of)<sup>1</sup> e término (End-Of) da execução de métodos<sup>23</sup> e eventos associados ao início e término de transações. Eventos básicos são combinados mediante os operadores da álgebra formando eventos compostos. O evento composto ocorre no instante da ocorrência do último evento constituinte. Para eventos compostos denotando eventos temporais, os atributos são derivados a partir dos atributos dos eventos básicos constituintes.

Propõe-se que o controle da ocorrência de eventos seja realizado através do uso de um log de eventos. Um *Log L* de eventos é representado por um conjunto finito de ocorrências distintas de eventos identificadas unicamente através do Time-Stamp. Uma expressão de eventos *EE* identificando um evento básico ou composto quando aplicada sobre o log *L*, dá origem a um log  $L' \subseteq L$  formado por eventos de *L* onde *EE* é válida.

Este *Log* deve ser acrescentado ao sistema ativo para permitir a manutenção de restrições comportamentais. Tais restrições podem exigir verificação de seqüências específicas ou alternâncias de eventos. Por este motivo, só podem ser verificadas se algum mecanismo de histórico de eventos é mantido.

Uma solução alternativa ao *Log* é proposta por Dittrich em [GD94a], que usa redes de Petri para esta verificação. Esta dissertação prefere a solução do *Log* porque pode ser naturalmente incorporada à dimensão temporal preconizada.

## 7.2 Mapeamento de restrições

Esta seção mostra como as restrições especificadas em CDL nas camadas de restrições e de objetos podem ser mapeadas para regras de produção no ambiente alvo tendo como base os requisitos descritos anteriormente.

---

<sup>1</sup>Na notação de SAMOS os modificadores são **BEFORE** e **AFTER**. Na notação de Snoop os comandos **Begin-Of** e **End-Of** são empregados. A escolha de Begin-Of e End-Of foi feita para realçar a distinção entre *CDL* e o nível operacional.

<sup>2</sup>Todo método que executa modificações sobre o estado do receptor da mensagem e que possui restrições associadas é realizado por uma subtransação da transação chamadora. .

<sup>3</sup>A chamada de um método também gera um evento permitindo que o receptor correto da mensagem seja identificado.

O modelo E-C-A alvo de mapeamento apresenta as principais características dos modelos de Ode, HiPAC e SAMOS, sendo fortemente influenciado pelos operadores e notação do último.

### Modelo E-C-A alvo

- **E**: identifica um evento básico ou uma expressão de eventos sobre o *log*. Eventos primitivos e compostos são definidos de acordo com o comando **DEFINE EVENT** especificado a seguir. O histórico inicia no instante em que o primeiro evento é definido e termina quando todas as definições de eventos são removidas<sup>4</sup>. O histórico persiste por várias sessões e transações permitindo que eventos compostos baseados em eventos de aplicações distintas sejam detectados. Os eventos primitivos registrados no *log* são:
  - **temporal absoluto** **DEFINE EVENT** *nome-ev*  
(dd.mm.aa : hh.mm.ss)
  - **temporal relativo** **DEFINE EVENT** *nome-ev* do tipo (*e* + *t*), onde *e* representa o momento de ocorrência de um evento definido e *t* representa uma constante temporal ou um evento temporal absoluto definido.
  - **método** **DEFINE EVENT** *nome-ev*  
{ nome-classe, nome-obj }.*nome-método*
  - **transação** **DEFINE EVENT** *nome-ev* { EOT, BOT, ABORT } *nome-transação*
  - **abstratos** **DEFINE EVENT** *nome-ev*

Eventos abstratos são sinalizados explicitamente pelas aplicações, sistema operacional ou núcleo do SGBD usando o comando **RAISE EVENT** *nome-ev*. Os operadores de composição de eventos adotados são:

- **disjunção**  $\vee$
- **conjunção**  $\wedge$
- **seqüência** ;
- **negação** !(E in *nome-I*)

Os modificadores de eventos usados no modelo são: **Begin-Of** e **End-Of**

---

<sup>4</sup>eventos são considerados como objetos.

Para a negação é necessário que um intervalo *nome-I* tenha sido previamente definido através do comando `DEFINE INTERVAL nome-I [et1 - et2]`, onde *et1* , *et2* são valores temporais. Os parâmetros dos eventos simples e compostos são acessados mediante as seguintes funções:

- **occ-point**[n](E) retorna o valor do relógio do sistema por ocasião da ocorrência do n-ésimo evento do evento composto E<sup>5</sup>
  - **occ-tid**[n](E) retorna o identificador da transação que gerou o n-ésimo evento de E
  - **object-id**[n](E) retorna o identificador do objeto associado ao n-ésimo evento de E
  - **user-id**[n](E) retorna o identificador do usuário associado ao n-ésimo evento de E
  - **method-par**[n][m](E) retorna o m-ésimo parâmetro do n-ésimo evento de E
- **C**: identifica uma consulta simples ou temporal sobre os estados do banco de dados ou sobre os parâmetros dos eventos. A condição será obtida pela negação dos predicados sobre valores das sentenças *CDL* e deve ser formulada usando a linguagem de manipulação do SGBD ativo, estendida para permitir consultas sobre os parâmetros dos eventos.
  - **A**: constitui a ação a ser disparada caso a condição seja verdadeira.

A sintaxe geral para a definição das regras no ambiente alvo é a seguinte:

```

DEFINE RULE nome-regra

ON nome-ev

IF consulta denotando a condição

THEN ação expressa na DML do SGBD

```

### 7.2.1 Mapeamento de Restrições sobre Estados

Para que restrições sobre estados sejam mapeadas, a expressão em *CDL* é analisada para identificar os eventos que podem violar as restrições. Os eventos identificados constituem o esquema de eventos do banco de dados que é definido usando comandos `DEFINE EVENT`.

<sup>5</sup>para eventos simples, o valor de n é 1.

Para identificar os pares (classe, método) que afetam a restrição, a seguinte heurística proposta por Medeiros[MP91] é usada:

1. Analisar a restrição *CDL* identificando as classes referenciadas na restrição
2. Os atributos da restrição são observados para verificar se os métodos pertencentes às classes identificadas anteriormente efetuam operações de alteração. Caso um método altere o atributo, o par (classe, método) deve ser definido no esquema de eventos do banco de dados ativo
3. Os pares (classe, método) obtidos no passo anterior formam os eventos-base. O esquema de entidades é percorrido para determinar outros eventos que podem violar a restrição através do grafo de herança

Utilizando a heurística de mapeamento, uma restrição sobre estados em *CDL* é mapeada para uma ou várias regras, de acordo com o seguinte critério:

- Definir os pares (classe, método) de acordo com a heurística anteriormente apresentada
- Definir cada evento  $E_i$  usando DEFINE EVENT
- Definir uma regra correspondente à restrição cujo evento é a disjunção de todos os eventos  $E_i$ , a condição representa a negação do predicado sobre estados formulada usando a linguagem de manipulação do banco de dados (consultas simples ou temporais) e a ação é um comando escolhido pelo usuário também expresso na DML.

### 7.2.2 Mapeamento de Restrições sobre Comportamento

Restrições comportamentais em *CDL* são mapeadas para regras de uma forma bastante uniforme dada a proximidade dos operadores de *CDL* dos operadores providos pelo modelo E-C-A adotado. Para o esquema de eventos, é necessário que todos os eventos primitivos envolvidos nas restrições sejam previamente definidos. A partir da definição dos eventos primitivos, as seguintes heurísticas definem o mapeamento das restrições em *CDL*:

- *Permissão de Acesso*: a restrição de permissão de acesso *CDL*<sup>6</sup> é mapeada para uma regra tendo como evento a expressão formada usando o comando:

```
DEFINE EVENT nome-ev {Begin-Of , End-Of }  $E_i$ 
```

---

<sup>6</sup> Pre (< evento >) (< predicado >) | Pos (< evento >) (< predicado >)

onde  $E_i$  representa a tradução de < evento > em termos dos eventos primitivos do modelo E-C-A adotado e Begin-Of ou End-Of<sup>7</sup> respectivamente para PRÉ ou PÓS condições. A regra correspondente à restrição *CDL* é a seguinte:

**ON** *nome-ev*

**IF** *consulta* verificando se os parâmetros de  $E_i$  estão de acordo com o < evento > correspondente em *CDL* sobre o qual a condição é formulada **and not** < predicado >  
**THEN** *ação* expressa na DML do SGBD

- *Permissão de Ordem*: a restrição de permissão de ordem em *CDL*<sup>8</sup> é mapeada em termos de duas regras no modelo E-C-A. A primeira tem como evento a expressão formada mediante a definição do evento composto:

```
DEFINE EVENT nome-ev1 !((EPO) in I) ; Begin-Of  $E_i$ 
```

onde EPO representa a tradução de < pred-ordem > em termos dos operadores de composição do modelo E-C-A,  $E_i$  corresponde à definição do < evento > sobre o qual existe uma PRÉ condição e I representa o intervalo de existência de eventos no banco de dados. Esta regra não necessita a avaliação dos parâmetros de EPO já que a seqüência de eventos EPO representando < pred-ordem > não ocorreu. A segunda regra tem como evento a expressão formada mediante a definição do evento composto:

```
DEFINE EVENT nome-ev2 (EPO) ; Begin-Of  $E_i$ 
```

cuja condição acessa os parâmetros do evento composto<sup>9</sup>. As regras correspondente às restrições em *CDL* são:

**ON** *nome-ev1*

**IF** *consulta* verificando se os parâmetros de  $E_i$  estão de acordo com o < evento > correspondente em *CDL* sobre o qual a PRÉ condição é formulada

<sup>7</sup>Quando um evento não tiver um modificador associado assume-se que o modificador *default* é End-Of.

<sup>8</sup> **Pre** (< evento >) (< pred-ordem >)

<sup>9</sup>note que não basta somente que a ordem dos eventos esteja correta mas também é necessário que os parâmetros estejam de acordo com a restrição *CDL*. Isto pode ser observado na restrição sobre a inclusão de habilidades apresentada no capítulo anterior, que impõe uma relação de ordem sobre eventos com parâmetros.



**THEN** ação expressa na DML do SGBD

**ON** nome-ev2

**IF** consulta verificando se os parâmetros de  $E_i$  estão de acordo com o < evento > correspondente em *CDL* sobre o qual a PRÉ condição é formulada **and not** < pred-ordem >, acessando os parâmetros dos eventos  $E_k$  que compõem a expressão de eventos correspondente à tradução do < pred-ordem > em *CDL*

**THEN** ação expressa na DML do SGBD

- *Permissão de Tempo*: restrições de permissão de tempo em *CDL*<sup>10</sup> podem ser de dois tipos: *absolutas*, quando < pred-clock > é do tipo < valor-clock > ou *relativas*, quando < pred-clock > for do tipo [ **TimeStamp** (< evento >) +] < valor-clock >. Para as absolutas, uma regra é derivadas no modelo E-C-A. A regra tem como evento uma expressão formada usando o comando:

DEFINE EVENT nome-ev Begin-Of  $E_i$

onde  $E_i$  representa a tradução de < evento > em termos dos eventos primitivos do modelo E-C-A adotado. A regra correspondente é a seguinte:

**ON** nome-ev

**IF** consulta verificando se os parâmetros de  $E_i$  estão de acordo com o < evento > correspondente em *CDL* sobre o qual a PRÉ condição é formulada **and not** ( **occ-point**[1](nome-ev)<sup>11</sup> *comparador*<sup>12</sup> *fator temporal*<sup>13</sup>)

**THEN** ação expressa na DML do SGBD

Para as restrições de permissão de tempo relativas, seja  $E_i$  a tradução do < evento > em *CDL* sobre o qual a pré-condição é formulada e  $E_j$  a tradução do < evento > em *CDL* sobre o qual é aplicada a função **TimeStamp**. Duas regras são derivadas no modelo E-C-A. A primeira tem como evento a expressão definida através do comando:

<sup>10</sup> **Pre** (< evento >) **Clock-Is** < comp-atributo > (< pred-clock >)

<sup>11</sup> instante de ocorrência do evento  $E_i$

<sup>12</sup> correspondente a < comp-atributo >

<sup>13</sup> constante ou evento temporal absoluto do modelo E-C-A correspondente a < valor-clock >

```
DEFINE EVENT nome-ev1  $E_j$  ; Begin-Of  $E_i$ 
```

□

e a segunda tem como evento a expressão definida pelo comando:

```
DEFINE EVENT nome-ev2 !( $E_j$  in I); Begin-Of  $E_i$ 
```

esta última indicando que o evento base do intervalo relativo não ocorreu, logo, se os parâmetros de  $E_i$  estiverem de acordo com os parâmetros do < evento > correspondente em *CDL*, a restrição foi quebrada. As regras correspondentes são:

```
ON nome-ev1
```

```
IF consulta verificando se os parâmetros de  $E_i$  estão de acordo com o < evento > correspondente em CDL sobre o qual a PRÉ condição é formulada and not ( occ-point[2](nome-ev1) comparador14 occ-point[1](nome-ev1)15 + fator temporal16)  
THEN ação expressa na DML do SGBD
```

```
ON nome-ev2
```

```
IF consulta verificando se os parâmetros de  $E_i$  estão de acordo com o < evento > correspondente em CDL sobre o qual a PRÉ condição é formulada. Se o evento não tiver parâmetros a condição é verdadeira  
THEN ação expressa na DML do SGBD
```

- *Obrigações*: a restrição de obrigação expressa em *CDL*<sup>17</sup> cujo momento futuro de execução da ação não é definido (**Eventually** < evento >) é mapeada para duas regras no modelo E-C-A. A primeira tem o evento definido pelo comando:

```
DEFINE EVENT nome-ev1 End-Of  $E_i$  ; !(  $E_k$  in I ) ; Begin-Of  $E_j$ 
```

onde  $E_i$  corresponde ao < evento > em *CDL* sobre o qual a póscondição é definida,  $E_j$  representa o evento de método gerado pelo destrutor da classe a qual pertence o objeto manipulado por  $E_i$ ,  $E_k$  representa o < evento > em *CDL* que deveria acontecer, e I o intervalo entre a ocorrência de  $E_i$  e a ocorrência de  $E_j$  (neste caso, deseja-se

<sup>14</sup>correspondente ao < comp-atributo > em *CDL*

<sup>15</sup>instante de ocorrência de  $E_j$

<sup>16</sup>constante ou evento temporal absoluto do modelo E-C-A correspondente a < valor-clock >

<sup>17</sup> **Pos** (< evento >) (< pred-obriga >)

< pred-obriga > ::= **Eventually** (< evento >) | **At** (< pred-clock >) (< evento >)

verificar se o objeto vai ser destruído antes que o comportamento obrigatório seja executado pelo objeto). A segunda regra tem como evento a expressão definida pelo comando:

```
DEFINE EVENT nome-ev2 End-Of  $E_i$  ;  $E_k$  ; Begin-Of  $E_j$ 
```

sendo  $E_i$  e  $E_k$  definidos analogamente ao caso anterior. (neste caso, ainda é necessário testar se os parâmetros dos eventos estão corretos). As regras correspondentes são:

**ON** *nome-ev1*

**IF** *consulta* verificando se os parâmetros de  $E_i$  e  $E_j$  estão de acordo com o < evento > correspondente em *CDL* sobre o qual a póscondição é formulada (neste caso deve ser verificado se trata-se do mesmo objeto recebendo as duas mensagens: mensagem indicando o < evento > sobre o qual a póscondição é definida e mensagem indicando a aplicação do destrutor)

**THEN** *ação* expressa na DML do SGBD

**ON** *nome-ev2*

**IF** *consulta* verificando se os parâmetros de  $E_i$  ,  $E_k$  e  $E_j$  estão de acordo com o < evento > correspondente em *CDL* sobre o qual a póscondição é formulada (neste caso deve ser verificado se trata-se do mesmo objeto recebendo as três mensagens: mensagem indicando o < evento > sobre o qual a póscondição é definida, mensagem indicando o comportamento obrigatório e mensagem indicando a aplicação do destrutor ao objeto, além da avaliação de possíveis parâmetros dos eventos)

**THEN** *ação* expressa na DML do SGBD

## 7.3 Exemplos

Esta seção apresenta o mapeamento das restrições expressas em *CDL* em termos do modelo E-C-A definido como ambiente alvo. O mapeamento das restrições segue as heurísticas de mapeamento propostas. Para as restrições sobre estados, a ação da regra por ocasião da quebra de integridade será simbolizada pelo comando NOTIFY, alertando o usuário, enquanto que restrições comportamentais possuem como ação o ABORT da transação que violou a restrição.

As consultas estão apresentadas numa notação hipotética, baseada em SQL com *path-expressions* e com operadores para acessar os atributos dos eventos, servindo para ilustrar

as consultas que representam a negação dos predicados.

Para a restrição: “Todo empregado só pode desempenhar no máximo uma atividade durante o mesmo intervalo de tempo” formulada em *CDL* como apresentado a seguir,

(For-all e In *Empregados*: (Card (e.alocado) <= 1 ))

Após a análise da restrição, os métodos { \$demitir, \$admitir, alocar } da classe *Empregados* e o método { incluir-empregado } da classe *Atividades* podem ocasionar a quebra da restrição.

As seguintes definições e regras representam o mapeamento da restrição para o modelo E-C-A alvo:

```
DEFINE EVENT e0 = Atividades.incluir-empregado
```

```
DEFINE EVENT e2 = Empregados.alocar
```

```
DEFINE EVENT e3 = Empregados.$admitir
```

```
DEFINE EVENT e4 = Empregados.$demitir
```

```
DEFINE EVENT ec0 = e0 ∨ e2 ∨ e3 ∨ e4
```

```
DEFINE RULE regra0
```

```
ON      ec0
```

```
IF      SELECT *
```

```
        FROM Empregados
```

```
        WHERE COUNT ( Empregados.alocado ) > 1
```

```
THEN    NOTIFY
```

Para a restrição: “Todo empregado alocado a uma atividade deve possuir todas as habilidades requisitadas pela atividade”, formulada em *CDL* como apresentado a seguir,

(For-all e In *Empregados*, For-all a In *Atividades* : (e.alocado == a Implies Is-equal (e.possui-hab , a.requisita-hab)))

Após a análise da restrição, os métodos { \$demitir, \$admitir, alocar } da classe *Empregados* e o método { incluir-empregado } da classe *Atividades* podem ocasionar a quebra da restrição.

As seguintes definições e regras representam o mapeamento da restrição para o modelo E-C-A alvo<sup>18</sup>:

```

DEFINE RULE regra3
ON      ec0
IF      SELECT *
        FROM  Empregados
        WHERE Empregados.alocado.requisita-hab NOT IN Empregados.possui-hab
THEN    NOTIFY

```

Para a restrição: “um empregado só pode ser considerado apto a ser Pesquisador após haver passado pelas etapas sucessivas de Graduado, Mestre e Doutor, nesta ordem” formulada em *CDL* como apresentado a seguir,

```

PRE (incluir-habilidade(Pesquisador) → e In Empregados)
(( incluir-habilidade(Graduado) → e In Empregados Before
incluir-habilidade(Mestre) → e In Empregados ) Before
incluir-habilidade(Doutor) → e In Empregados )

```

As seguintes definições e regras representam o mapeamento da restrição para o modelo E-C-A alvo:

```

DEFINE EVENT e1 = Empregados.incluir-habilidade

DEFINE EVENT e01 = Begin-Of Empregados.incluir-habilidade

DEFINE EVENT ec1 = !(e1 ; e1 ; e1 in I); e01

DEFINE EVENT ec2 = e1 ; e1 ; e1; e01

DEFINE RULE regra1
ON      ec1
IF      method-par[4][1](ec1) = "pesquisador"
THEN    ABORT

```

<sup>18</sup>note que os eventos primitivos e o evento composto foram definidos anteriormente

```

DEFINE RULE regra2
ON      ec2
IF      method-par[4][1](ec2) = "pesquisador" and
        not ( method-par[3][1](ec2) = "doutor" and
              method-par[2][1](ec2) = "mestre" and
              method-par[1][1](ec2) = "graduado" and
              object-id[1](ec2) = object-id[4](ec2) and
              object-id[2](ec2) = object-id[4](ec2) and
              object-id[3](ec2) = object-id[4](ec2)
        )
THEN ABORT

```

Para a restrição: “Para que um empregado possa ser alocado a um projeto é necessário que ele passe por um período de treinamento de no mínimo um ano, contado a partir da data em que foi incluído como empregado” formulada em *CDL* como apresentado a seguir:

```

PRE (alocar → e In Empregados) (Clock-Is >= Timestamp ( $admitir → e In
Empregados) + (00/12/00/00/00/00) )

```

As seguintes definições e regras representam o mapeamento da restrição para o modelo E-C-A alvo:

```

DEFINE EVENT e03 = Begin-Of Empregados.alocar

```

```

DEFINE EVENT ec3 = e3 ; e03

```

```

DEFINE EVENT ec4 = !(e3 in I) ; e03

```

```

DEFINE RULE regra3
ON      ec3
IF      object-id[1](ec3) = object-id[2](ec3) and
        not ( occ-point[2](ec3) >= occ-point[1](ec3)
              + (00.00.03:00.00.00)
        )
THEN ABORT

```

```

DEFINE RULE regra4

```

```

ON      ec4
IF      True
THEN    ABORT

```

Para a restrição: “Quando um empregado inicia um treinamento, em algum momento do futuro deve concluí-lo” formulada em *CDL* como apresentado a seguir:

POS ( iniciar-treinamento  $\rightarrow$  e In *Empregados*) **Eventually** (terminar-treinamento  $\rightarrow$  e In *Empregados*)

As seguintes definições e regras representam o mapeamento da restrição para o modelo E-C-A alvo:

```

DEFINE EVENT e04 = Begin-Of Empregados.$demitir
.
DEFINE EVENT e5  = Empregados.iniciar-treinamento
.
DEFINE EVENT e6  = Empregados.terminar-treinamento
.
DEFINE EVENT ec5 = e5 ; ! (e6 in I) ; e04
.
DEFINE EVENT ec6 = e5 ; e6 ; e04
.
.
DEFINE RULE regra5
ON      ec5
IF      object-id[1](ec5) = object-id[3](ec5)
THEN    ABORT
.
.
DEFINE RULE regra6
ON      ec6
IF      Object-id[1](ec6) = object-id[3](ec6)
        and not (object-id[2](ec6) = Object-id[3](ec6))
THEN    ABORT

```

Para a restrição: “A política da agência é manter empregados na faixa de 25 a 65 anos” formulada em *CDL* como apresentado a seguir:

POS ( \$admitir  $\rightarrow$  e In *Empregados*) (For-all e In *Empregados*: ( e.idade  $\geq$  25 And e.idade  $\leq$  65))

A seguinte regra representa o mapeamento da restrição para o modelo E-C-A alvo:

```
DEFINE RULE regra7
ON      e3
IF      SELECT *
        FROM  Empregados
        WHERE Empregados.idade < 25 OR Empregados.idade > 65
THEN    NOTIFY
```



# Capítulo 8

## Conclusões e Extensões

Esta dissertação discutiu o problema de restrições de integridade em bancos de dados, mostrando como o problema pode ser encarado de forma integrada, a partir da especificação da aplicação. Esta especificação é formulada em etapas, através de diversas camadas de um banco de dados, usando como base o modelo OMT estendido com a linguagem CDL, proposta para especificação das restrições.

O banco de dados subjacente é um sistema ativo orientado a objetos, que suporta a dimensão temporal e permite a manutenção de um log de eventos. Estas duas últimas facilidades são necessárias para que se possa manter restrições dinâmicas em geral.

Vários aspectos foram considerados no desenvolvimento da estratégia de mapeamento de restrições em termos de regras do banco de dados ativo. Dentre as principais contribuições, podemos destacar:

- **Identificação da Natureza das Restrições:** as restrições foram classificadas no contexto de uma taxonomia de restrições na modelagem de sistemas de informação, proposta na dissertação.
- **Especificação das Restrições:** a linguagem CDL foi proposta para especificar restrições no projeto conceitual.
- **Análise do Ambiente Alvo:** o ambiente do sistema ativo foi analisado, identificando-se os mecanismos necessários para manter restrições gerais. Um modelo E-C-A do ambiente alvo foi formulado.
- **Mapeamento:** heurísticas de mapeamento das restrições expressas na notação declarativa foram elaboradas.

As extensões à dissertação são tanto teóricas quanto práticas. Uma primeira extensão prática é a implementação do mecanismo de mapeamento e de um compilador para restrições expressas em CDL.

A estratégia de projeto proposta está bastante simplificada, não considerando alguns aspectos importantes em sistemas de regras. Um destes aspectos refere-se à incorporação de uma etapa posterior ao mapeamento visando analisar o comportamento das regras quando consideradas conjuntamente[SvdVK93]. O objetivo desta etapa é o de verificar conflitos, possibilidade de ciclos de disparo e regras contraditórias.

Outra etapa não considerada refere-se à validação das especificações lógicas e verificação da consistência dos esquemas. Trabalhos nesta direção podem ser encontrados em[Ara90, Ara92, Tan92].

O estudo não contemplou restrições relativas a tempo real, que podem ser incorporadas estendendo os operadores temporais de *CDL*. Outra extensão possível é a de permitir as restrições intervalares.

Ainda outras extensões se referem à implementação do ambiente alvo especificado. As várias propostas existentes para mecanismos ativos não consideram a possibilidade de incorporar a dimensão temporal. Assim sendo, é necessário explorar melhor as consequências de incluir a dimensão temporal para que um sistema ativo gerencie estados e comportamento ao longo do tempo. Esta nova dimensão pode acarretar dificuldades do ponto de vista de gerenciamento de regras (pois é concebível imaginar que determinadas regras só sejam válidas em determinados instantes históricos).

# Apêndice A

## Linguagem de Definição de Restrições (CDL)

1.  $\langle \text{restrição} \rangle ::= (\langle \text{predicado} \rangle)$

| **Pre** ( $\langle \text{evento} \rangle$ ) ( $\langle \text{predicado} \rangle$ )

| **Pos** ( $\langle \text{evento} \rangle$ ) ( $\langle \text{predicado} \rangle$ )

| **Pre** ( $\langle \text{evento} \rangle$ ) **Clock-Is**  $\langle \text{comp-atributo} \rangle$  ( $\langle \text{pred-clock} \rangle$ )

| **Pre** ( $\langle \text{evento} \rangle$ ) ( $\langle \text{pred-ordem} \rangle$ )

| **Pos** ( $\langle \text{evento} \rangle$ ) ( $\langle \text{pred-obriga} \rangle$ )

2.  $\langle \text{pred-obriga} \rangle ::= \text{Eventually } (\langle \text{evento} \rangle)$

| **At** ( $\langle \text{pred-clock} \rangle$ ) ( $\langle \text{evento} \rangle$ )

3.  $\langle \text{pred-ordem} \rangle ::= \langle \text{pred-ordem} \rangle \vee \langle \text{evento-1} \rangle$

|  $\langle \text{evento-1} \rangle$

- 
4.  $\langle \text{evento-1} \rangle ::= \langle \text{evento-1} \rangle \wedge \langle \text{evento-2} \rangle$   
 $| \langle \text{evento-2} \rangle$
5.  $\langle \text{evento-2} \rangle ::= \langle \text{evento-2} \rangle \mathbf{Before} \langle \text{evento} \rangle$   
 $| (\langle \text{pred-ordem} \rangle)$   
 $| \neg (\langle \text{pred-ordem} \rangle)$   
 $| \langle \text{evento} \rangle$
6.  $\langle \text{pred-clock} \rangle ::= [ \mathbf{TimeStamp} (\langle \text{evento} \rangle) + ] \langle \text{valor-clock} \rangle$
7.  $\langle \text{evento} \rangle ::= \langle \text{envio} \rangle$   
 $| \mathbf{Insert} \langle \text{var} \rangle \mathbf{In} \langle \text{dominio} \rangle$   
 $| \mathbf{Remove} \langle \text{var} \rangle \mathbf{In} \langle \text{dominio} \rangle$   
 $| \langle \text{nome-trans} \rangle | \langle \text{evento-externo} \rangle$
8.  $\langle \text{predicado} \rangle ::= \langle \text{quantificador} \rangle : (\langle \text{seleção} \rangle)$
9.  $\langle \text{quantificador} \rangle ::= [\mathbf{Not}] \mathbf{Exists} \langle \text{var} \rangle \{, \langle \text{var} \rangle\} \mathbf{In} \langle \text{domínio} \rangle [, \langle \text{quantificador} \rangle]$   
 $| \mathbf{For-all} \langle \text{var} \rangle \{, \langle \text{var} \rangle\} \mathbf{In} \langle \text{domínio} \rangle [, \langle \text{quantificador} \rangle]$
10.  $\langle \text{seleção} \rangle ::= \langle \text{seleção-2} \rangle \mathbf{Implies} \langle \text{seleção-2} \rangle$   
 $| \langle \text{seleção-2} \rangle$

- 
11.  $\langle \text{seleção-2} \rangle ::= \langle \text{seleção-2} \rangle \mathbf{Or} \langle \text{seleção-3} \rangle$   
|  $\langle \text{seleção-3} \rangle$
12.  $\langle \text{seleção-3} \rangle ::= \langle \text{seleção-3} \rangle \mathbf{And} \langle \text{condição} \rangle$   
|  $\mathbf{Always} \langle \text{seleção-3} \rangle [\mathbf{Since} \langle \text{condição} \rangle]$   
|  $\mathbf{Sometime} \langle \text{seleção-3} \rangle [\mathbf{Since} \langle \text{condição} \rangle]$   
|  $\mathbf{Always} \langle \text{seleção-3} \rangle \mathbf{Since} \langle \text{pred-clock} \rangle$   
|  $\mathbf{Sometime} \langle \text{seleção-3} \rangle \mathbf{Since} \langle \text{pred-clock} \rangle$   
|  $(\langle \text{seleção-2} \rangle)$   
|  $\langle \text{condição} \rangle$
13.  $\langle \text{condição} \rangle ::= \mathbf{Not} (\langle \text{expressão} \rangle)$   
|  $\langle \text{expressão} \rangle$   
|  $\mathbf{True} \mid \mathbf{False}$
14.  $\langle \text{expressão} \rangle ::= \langle \text{exp-simples} \rangle \langle \text{comparador} \rangle \langle \text{exp-simples} \rangle$   
|  $\langle \text{exp-simples} \rangle$
15.  $\langle \text{exp-simples} \rangle ::= [+ \mid -] \langle \text{termo} \rangle \{ (+ \mid -) \langle \text{termo} \rangle \}$   
|  $\mathbf{Previous} \langle \text{termo} \rangle$

- 
16.  $\langle \text{termo} \rangle ::= \langle \text{fator} \rangle \{ (* | /) \langle \text{fator} \rangle \}$
17.  $\langle \text{fator} \rangle ::= \langle \text{var-exp} \rangle$   
 $| \langle \text{string} \rangle | \langle \text{numero} \rangle | (\langle \text{expressao} \rangle)$   
 $| \langle \text{exp-agregacao} \rangle | \langle \text{exp-conjunto} \rangle$
18.  $\langle \text{exp-agregacao} \rangle ::= \langle \text{op-agrega} \rangle (\langle \text{var-exp} \rangle)$
19.  $\langle \text{exp-conjunto} \rangle ::= \langle \text{var} \rangle \mathbf{Is-in} \langle \text{dominio} \rangle$   
 $| \langle \text{op-pert} \rangle (\langle \text{var-exp} \rangle, \langle \text{var-exp} \rangle)$
20.  $\langle \text{var-exp} \rangle ::= \langle \text{var} \rangle . \langle \text{atributo} \rangle$
21.  $\langle \text{comparador} \rangle ::= \langle \text{comp-atributo} \rangle | \langle \text{comp-objeto} \rangle$
22.  $\langle \text{comp-atributo} \rangle ::= = | \langle \rangle | > | < | >= | <=$
23.  $\langle \text{comp-objeto} \rangle ::= == | !=$
24.  $\langle \text{op-agrega} \rangle ::= \mathbf{Sum} | \mathbf{Min} | \mathbf{Max} | \mathbf{Card}$
25.  $\langle \text{valor-clock} \rangle ::= \text{“valor indicando data e hora (DD/MM/AA/HH/MM/SS) ”}$
26.  $\langle \text{op-pert} \rangle ::= \mathbf{Contains} | \mathbf{Is-in} | \mathbf{Is-equal}$
27.  $\langle \text{envio} \rangle ::= \langle \text{método} \rangle \rightarrow \langle \text{var} \rangle \mathbf{In} \langle \text{domínio} \rangle$

28.  $\langle \text{var} \rangle ::= \text{“Identificador de Variável”}$
29.  $\langle \text{domínio} \rangle ::= \text{“Identificador de Classe”}$
30.  $\langle \text{atributo} \rangle ::= \text{“Identificador de Atributo”}$
31.  $\langle \text{método} \rangle ::= \text{“Identificador de método ”}$
32.  $\langle \text{nome-trans} \rangle ::= \text{“Identificador de uma transação”}$
33.  $\langle \text{evento-externo} \rangle ::= \text{“Identificador de Evento externo”}$

# Bibliografia

- [ACC<sup>+</sup>93] M. Adiba, C. Collet, T. Coupaye, P. Habraken, J. Machado, H. Martin, and C. Roncancio. Triggers systems: Different approaches. Technical Report SUR007, IMAG - LGI - BULL, June 1993.
- [ACM80] ACM. *Proceedings of the Workshop on Data Abstraction, Databases and Conceptual Modelling*, volume 11 of *SIGART, SIGMOD Record*. ACM Press, February 1980.
- [ACO85] A. Albano, L. Cardelli, and R. Orsini. Galileo; a strongly typed interactive conceptual language. *ACM TODS*, 10:230–260, 1985.
- [AH87] S. Abiteboul and R. Hull. Ifo - a formal semantic database model. *ACM TODS*, 12(4):525–565, 1987.
- [All83] James F. Allen. Maintaining knowledge about temporal intervals. *CACM*, 26(11):832–843, November 1983.
- [And92] Marcia Jacobina Brito Andrade. Manutenção de restrições de integridade em banco de dados orientado a objetos. Master's thesis, Universidade Estadual de Campinas, 1992.
- [Ara90] Constantin Arapis. Specifying object life-cycles. Technical report, Universite de Geneve, 1990.
- [Ara92] Constantin Arapis. Object behavior composition: A temporal logic based approach. Technical report, Universite de Geneva, 1992.
- [BCN92] Carlo Batini, Stefano Ceri, and Shamkant Navathe. *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin Cummings, 1992.
- [BDK92] F. Bancilhon, C. Delobel, and P. Kanellakis. *Building an Object-Oriented Database System - The Story of O2*, chapter 1. Morgan Kaufmann, 1992.



- [BNP<sup>+</sup>91] Peter Mc. Brien, Marc Niezette, Dionysios Pantazis, Anne Helga Seltveit, Ulf Sundin, Babis Theodoulidis, Gregoris Tziallas, and Rolf Wohed. A rule language to capture and model business policy specifications. In *Proc. 3rd Nordic Conference on Advanced Information Systems Engineering (CAiSE)*, 1991. also in LNCS.
- [Boo93] Grady Booch. *Object-Oriented Design*. Benjamin/Cummings, 2nd edition, 1993.
- [BP85] G. Bracchi and B. Pernici. Specification of control aspects in office information systems. In *Information Systems: Theoretical and Formal Aspects*, pages 211–224. North-Holland, 1985.
- [Bro84] Michael L. Brodie. On the development of data models. In *On Conceptual Modelling*. Springer-Verlag, 1984.
- [CCF82] J. Castilho, M. Casanova, and A. Furtado. A temporal framework for database specifications. In *Proceedings of the VLDB Conference*, pages 280–291, 1982.
- [Che76] P. P. Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
- [Cho92] Jan Chomicki. History-less checking of dynamic integrity constraints. In *Proceedings of the 8th IEEE Conference on Data Engineering*, Phoenix, Arizona, February 1992.
- [CM93] S. Chakravarthy and D. Mishra. Snoop; an expressive event specification language for active databases. Technical Report UF-CIS-TR-93-007, University of Florida, March 1993.
- [CS88] J. Carmo and A. Sernadas. A temporal logic framework for a layered approach to systems specification and verification. In C. Rolland, F. Bodart, and M. Leonard, editors, *Temporal Aspects in Information Systems*, pages 31–46. IFIP TC 8/WG, North-Holland, 1988. 8.1 Working Conference on Temporal Aspects in Information Systems.
- [CSS93] Rakesh Chandra, Arie Segev, and Michael Stonebraker. Implementing calendars and temporal rules in next generation databases. Technical report, Lawrence Berkeley Laboratory, 1993.
- [CW90] S. Ceri and J. Widom. Deriving production rules for constraint maintenance. In *Proceedings of the 16th Intl. Conference on VLDB*, 1990.

- [CY91] P. Coad and E. Yourdon. *Object-Oriented Design*. Prentice-Hall, 1991.
- [DBB+88] U. Dayal, B. Blaustein, A. Buchmann, M. Hsu, S. Chakravarthy, R. Ledin, D. McCarthy, A. Rosenthal, S. Sarin, M. Carey, M. Livny, and R. Jauhari. The hipac project: Combining active databases and timing constraints. *ACM SIGMOD RECORD*, 17(1):51-70, March 1988.
- [DBM88] U. Dayal, A. Buchmann, and D. McCarthy. Rules are objects too: A knowledge model for an active, object-oriented database system. In *Lecture Notes in Computer Science*, volume 334, pages 129-143. Springer-Verlag, 1988.
- [Dij76] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.
- [DPG91] O. Diaz, N. Paton, and P. Gray. Rule management in object-oriented databases; a uniform approach. In *Proceedings of the International VLDB Conference*, 1991.
- [Ede94] Nina Edelweiss. *Sistemas de Informação de Escritórios: Um Modelo Para Especificações Temporais*. PhD thesis, Universidade Federal do Rio Grande do Sul, 1994.
- [EdO94] Nina Edelweiss and José Palazzo M. de Oliveira. *Modelagem de Aspectos Temporais de Sistemas de Informação*. IX Escola de Computação, 1994.
- [EJDS92] Hans-Dieter Ehrich, Ralf Jungclaus, Grit Denker, and Amilcar Sernadas. Object-oriented design of information systems: Theoretical foundations. Technical report. Universität Braunschweig, 1992.
- [Elm91] Ramez Elmasri. Recent advances in temporal databases. In *Anais do VII Simpósio Brasileiro de Banco de Dados*, pages 17-26, 1991.
- [EN89] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Morgan Kaufman, 1989.
- [End72] H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.
- [FN86] Antonio L. Furtado and Erich J. Neuhold. *Formal Techniques For Data Base Design*. Springer-Verlag, 1986.
- [GD93] Stella Gatzui and Klaus Dittrich. Events in an active object oriented database system. In *Proceedings of the 1st Intl. Workshop on Rules in Database Systems*, August 1993.

- [GD94a] Stella Gatzju and Klaus R. Dittrich. Detecting composite events in active database systems using petri nets. In *4th Intl. Workshop on Research Issues in Data Engineering: Active Database Systems*, Houston, Texas, February 1994.
- [GD94b] Andreas Geppert and Klaus R. Dittrich. Specification and implementation of consistency constraints in object-oriented database systems: Applying programming-by-contract. Technical report, Institut Fur Informatik Universitaet Zuerich, October 1994.
- [GD94c] Thomas Grotehen and Klaus R. Dittrich. Erweiterung konzeptueller objektmodelle für den entwurf aktiver systeme. Technical report. Universität Zürich - IFI, 1994.
- [GGD91] S. Gatzju, A. Geppert, and K. Dittrich. Integrating active concepts into an object-oriented database system. In *Proceedings of the 3rd Intl. Workshop on Database Programming Languages*, Nafplion, Greece, August 1991.
- [GJS92a] N. Gehani, H. V. Jagadish, and O. Shmueli. Compose: a system for composite event specification and detection. Technical report, ATT Bell Laboratories, December 1992.
- [GJS92b] N. H. Gehani, H. V. Jagadish, and O. Shmueli. Event specification in an active object-oriented database. In *Proceedings of the SIGMOD International Conference*, pages 81-90, 1992.
- [GMJ91] Carlo Ghezzi, Dino Mandreoli, and M. Jazayeri. *Fundamentals of Software Engineering*. Prentice-Hal, 1991.
- [HK87] R. Hull and R. King. Semantic database modeling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3):201-260, 1987.
- [HM81] M. Hammer and D. McLeod. Database description with sdm: A semantic database model. *ACM TODS*, 6(3):351-381, 1981.
- [JHS92] Ralf Jungclaus, Thorsten Hartman, and Gunter Saake. Relationships between dynamic objects. Technical report, TU Braunschweig, 1992.
- [JMSS90] M. Jarke, S. Mazumdar, E. Simon, and D. Stemple. Assuring database integrity. *Journal of Database Adm.*, 1(1):391-400, 1990.
- [JS91] Ralf Jungclaus and Gunter Saake. Formal specification of object systems. In *Proceedings of the TAPSOFT 91*, 1991.

- [JSH91] Ralf Jungclaus, Gunter Saake, and Thorsten Hartmann. Language features for object-oriented conceptual modelling. In *Proceedings of the 10th Int. Conf. on the ER-Approach*. 1991.
- [JSHS91] Ralf Jungclaus, Gunter Saake, Thorsten Hartmann, and Cristina Sernadas. Object-oriented specification of information systems: The troll language. Technical Report 91-04, Technische Universität Braunschweig, December 1991.
- [Kap91] G. Kappel. Using an object-oriented diagram technique in the design of information systems. In *Dynamic Modelling of Information Systems*. Elsevier Science Publishers, 1991.
- [KC86] Setrag N. Khosafian and George P. Copeland. Object identity. In *OOPSLA '86 Proceedings*, pages 406–415, 1986.
- [LK92] P. Loucopoulos and E. Katsouli. Modelling business rules in an office environment. *SIGOIS*, 1:28–37, 1992.
- [LS87] U. W. Lipeck and G. Saake. Monitoring dynamic integrity constraints based on temporal logic. *Information Systems*, 12(3):255–269. 1987.
- [MA92] C. B. Medeiros and M. J. Andrade. Implementing Integrity Control in Active Databases. In *Proc. 1st International Conference on Information and Knowledge Management*, pages 310–317, Baltimore, November 1992.
- [Mar91] James Martin. *Engenharia da Informação*. Editora Campus, 1991.
- [MBW80] J. Mylopoulos, P. Bernstein, and H. Wong. A language facility for designing interactive database-intensive applications. *ACM TODS*. 5(2):185–207, 1980.
- [MD89] D. R. McCarthy and U. Dayal. The architecture of an active database management system. In *Proceedings of the ACM SIGMOD*, pages 215–224, Portland, Oregon, June 1989.
- [Mey88] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, 1988.
- [MP91] C. B. Medeiros and P. Pfeffer. Object integrity using rules. In *Proceedings of the ECOOP*, pages 219–230, 1991.
- [MP92] David E. Monarch and Gretchen I. Puhr. A research typology for object-oriented analysis and design. *CACM*, 35(9):35–47, September 1992.
- [Nav92] Shamkant Navathe. Evolution of data modelling for databases. *CACM*, September 1992.

- [Ner92] J. M. Nerson. Applying object-oriented analysis and design. *CACM*, September 1992.
- [Oea91] T. William Olle and et al. *Information Systems Methodologies*. Addison-Wesley, 2nd edition, 1991.
- [Oel91] Anders Oelmann. Representing a system specification with a temporal dimension in an object-oriented language. In *Proceedings of the CAISE' 91*, pages 540–560. Lecture Notes in Computer Science 436. 1991.
- [PM88] J. Peckham and F. Maryanski. Semantic data models. *ACM Computing Surveys*, 20(3):153–189. 1988.
- [PSE<sup>+</sup>94] Niki Pissinou, R. Snodgrass, R. Elmasri, I. Mumick, M. Ozsu, B. Pernici, A. Segev, B. Theodoulidis, and U. Dayal. Towards an infrastructure for temporal databases: Report of an invitational arpa/nsf workshop. *SIGMOD RECORD*, 23(1):35–50, March 1994.
- [RBP<sup>+</sup>91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorenzen. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.
- [Rei85] Wolfgang Reisig. *Petri Nets An Introduction*. Springer-Verlag, 1985.
- [RU71] N. Rescher and A. Urquhart. *Temporal Logic*. Springer-Verlag, 1971.
- [Saa91] Gunter Saake. Conceptual modelling of database applications. In *Proceedings of Information Systems and Artificial Intelligence: Integration Aspects*, LNCS 474, 1991.
- [SC92] Stanley Y. W. Su and Hsin-Hsing M. Chen. Temporal rule specification and management in object-oriented knowledge bases. Technical Report TR-92-026, University of Florida, CIS, 1992.
- [Ser80] Amilcar Sernadas. Temporal aspects of logical procedure definition. *Information Systems*, 5:167–187, 1980.
- [Set86] Waldemar Setzer. *Projeto L'ogico e Projeto F'isico de Banco de Dados*. Escola de Computação, 1986.
- [Spa93] Stefano Spaccapietra. Object orientation and conceptual modelling. Technical report, Ecole Polytechnique Federale - Lausanne, 1993.
- [SS87] Amilcar Sernadas and Cristina Sernadas. Object-oriented specification of databases: An algebraic approach. In *Proceedings of the 13th VLDB Conference*, pages 107–116, 1987.

- [SS88] Arie Segev and Arie Shoshani. Modeling temporal semantics. *Proceedings of the IFIP TC 8/WG Working Conference on Temporal Aspects in Information Systems*, 1988.
- [SS89] Tim Sheard and David Stemple. Automatic verification of database transaction safety. *ACM Transactions on Database Systems*, 14(3):322–368, September 1989.
- [SSU91] A. Silberschatz, M. Stonebraker, and J. Ullman. Database systems: achievements and opportunities. *CACM*, October 1991.
- [Sto92] Michael Stonebraker. The integration of rule systems and database systems. *IEEE Transactions on Knowledge and Data Engineering*, 4(5):415–423, October 1992.
- [SvdVK93] A. Siebes, M. van der Voort, and M. Kersten. Towards a design theory for database triggers. Technical Report CS-R9201, CWI, 1993.
- [Tan92] Asterio Kiyoshi Tanaka. *On Conceptual Design of Active Databases*. PhD thesis, Georgia Institute of Technology, 1992.
- [TL91] Charalampos I. Theodoulidis and Pericles Loucopoulos. The time dimension in conceptual modelling. *Information Systems*, 16(3):273–300, 1991.
- [TLW91] C. Theodoulidis, P. Loucopoulos, and B. Wangler. A conceptual modelling formalism for temporal database applications. *Information Systems*, 16(4):401–416, 1991.
- [TPC94] M. Teisseire, P. Poncelet, and R. Cicchetti. Towards event-driven modelling for database design. In *Proceedings of the 20th VLDB Conference*, pages 285–295, 1994.
- [UD89] Susan Darling Urban and Lois M. L. Delcambre. Constraint analysis for specifying perspectives of class objects. In *Proceedings of the IEEE Data Engineering Conference*, pages 10–17, 1989.
- [UD90] Susan D. Urban and Lois M. L. Delcambre. Constraint analysis: A design process for specifying operations on objects. *IEEE Transactions on Knowledge and Data Engineering*, 2(4):391–400, December 1990.
- [UKN92] S. Urban, A. Karadimce, and R. Nannapaneni. The implementation and evaluation of integrity maintenance rules in an object-oriented database. In *Proceedings of the IEEE*, pages 565–572, 1992.

- 
- [vdVK93] M. van der Voort and M. Kersten. Facets of database triggers. Technical Report CS-R9122, CWI, April 1993.
- [Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, March 1992.
- [YC79] E. Yourdon and L. Constantine. *Structured Design: Fundamentals of a Discipline of Computer Programming and Design*. Prentice-Hall, 2nd edition, 1979.