UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ROBSON LEONARDO FERREIRA CORDEIRO

Classificação e Especificação de Restrições de Integridade em Bancos de Dados Temporais de Versões

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Clesio Saraiva dos Santos
Orientador

Prof ª. Dra. Nina Edelweiss
Co-orientadora

Porto Alegre, abril de 2005.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Cordeiro, Robson Leonardo Ferreira

Classificação e Especificação de Restrições de Integridade em Bancos de Dados Temporais de Versões / Robson Leonardo Ferreira Cordeiro – Porto Alegre: Programa de Pós-Graduação em Computação, 2005.

127 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2005. Orientador: Clesio Saraiva dos Santos; Co-orientadora: Nina Edelweiss.

1. Classificação e Especificação de Restrições de Integridade 2. Modelagem de Dados 3. Tempo 4. Versões. I. Santos, Clesio Saraiva dos. II. Edelweiss, Nina. III. Título.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

SGBD	Sistema Gerenciador de Bancos de Dados
TVM	Temporal Versions Model
TVCL	Temporal Versioned Constraint Language
TVQL	Temporal Versioned Query Language
TVL/SE	Temporal Versioned Language for Schema Evolution
CAD	Computer-Aided Design
OID	Object Identifier
BNF	Backus Naur Form
ACID	Atomicidade, Consistência, Independência e Durabilidade
DML	Data Manipulation Language
SQL	Structured Query Language
DBA	Database Administrator
OQL	Object Query Language
ODMG	Object Data Management Group
UML	Unified Modeling Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Um Sistema gerenciador de Bancos de Dados (SGBD) possui como principal característica a capacidade de gerenciar bases de dados que representam parte do mundo real. Para que essa representação seja fiel, os dados presentes em uma base de dados devem obedecer a diversas regras conhecidas como restrições de integridade. Estas podem ser provenientes da realidade modelada, da implementação ou do modelo de dados utilizado.

O suporte oferecido por sistemas gerenciadores de bancos de dados tradicionais não é suficientemente adequado a certas aplicações com necessidades que vão além das convencionais. Diversas aplicações necessitam armazenar dados históricos em conjunto com seus períodos de validade. Outras precisam armazenar versões de conjuntos de dados, gerenciando suas agregações e formas de representação. Através do suporte aos conceitos de tempo e de versão, provido por um SGBD, grande parte dessas necessidades é suprida. Este tipo de banco de dados usa o conceito de tempo para armazenar e controlar dados históricos enquanto o conceito de versão permite a gerência de alternativas de projeto.

Existem atualmente diversos trabalhos e implementações relacionados à manutenção de restrições de integridade sobre bancos de dados tradicionais. Entretanto, restrições que consideram a gerência de tempo e de versões sobre dados ainda representam uma área de pesquisa praticamente inexplorada.

De acordo com essa realidade, o primeiro objetivo do presente trabalho consiste em definir uma classificação de restrições de integridade para bases de dados com suporte a tempo e versões, a fim de prover uma base para o desenvolvimento de pesquisas relacionadas à sua especificação e manutenção. O segundo objetivo consiste em agregar ao Modelo Temporal de Versões (TVM), que suporta os conceitos de tempo e de versão, uma linguagem que permita a especificação de restrições de integridade. Esta linguagem considera características relacionadas à temporalidade e ao versionamento dos dados e das próprias restrições.

Palavras-Chave: Classificação e Especificação de Restrições de Integridade, Modelagem de Dados, Tempo e Versões.

Integrity Constraints Classification and Specification for Temporal Versions Databases

ABSTRACT

The main feature of a Database Management System (DBMS) is to manage databases that represent some part of the real world. In order to make a faithful representation of reality, data stored in a database must obey several rules named integrity constraints. These constraints can be originated by the modeled reality, by the implementation or by the used data model.

The support offered by a snapshot database is not fully adequate for some applications with non-conventional needs. Many applications need to register historical data together with their validity periods. Other need to control data sets versions, managing their aggregations and representation forms. Through the support of time and version concepts, offered by a DBMS, many of these needs are supplied. This kind of database uses the time concept to control and store historical data while the version concept allows managing several project alternatives.

There are many researches and implementations on the integrity maintenance of snapshot databases. However, constraints that take into account time and versions management over data still represent an almost unexplored research area.

According to this reality, the first goal of this work is to define an integrity constraints classification for databases with time and versions support. This classification may provide a base for the development of researches on their specification and maintenance. The second goal is to increase the Temporal Versions Model (TVM), that supports time and version concepts, with an specification language for integrity constraints. This language gives support to the time and versions characteristics of data and of the constraints themselves.

Keywords: Integrity Constraints Classification and Specification, Data Modeling, Time and Versions.

1 INTRODUÇÃO

1.1 Motivação

A principal funcionalidade de um Sistema Gerenciador de Bancos de Dados (SGBD) é gerenciar bases de dados que representam parte do mundo real. A fim de possibilitar uma representação fiel da realidade modelada, os dados presentes em um banco de dados devem acatar a diversas restrições de integridade (DOUCET et al., 1997).

Bancos de dados tradicionais, conhecidos como *Snapshot Databases*, são suficientemente capazes de suprir as necessidades de certas aplicações. Entretanto, existem aplicações onde o armazenamento de um histórico de estados de sua base de dados se faz necessário, assim como a possibilidade de registro de dados válidos em períodos pré-determinados. Além disso, existem aplicações com a necessidade de armazenar versões de um mesmo conjunto de dados ou objetos, armazenar diferentes formas de representação de um mesmo objeto e controlar suas agregações.

A grande maioria dessas necessidades é suprida através de bancos de dados temporais com suporte a versões. Neste trabalho, referências a este tipo de banco de dados são feitas através do termo Banco de Dados Temporal de Versões. Estes bancos de dados utilizam o conceito de tempo para controlar e armazenar dados históricos, enquanto o conceito de versão é aplicado a fim de permitir alternativas de projeto.

Existem atualmente diversos trabalhos e implementações de técnicas para definição e verificação otimizada de restrições de integridade em bancos de dados tradicionais (WEN-CHI; ZHANG, 1995; FREEMAN-BENSON; BORNING, 1992; CASTILHO, 1987; SANTOS, 1980). Em relação a este tipo de banco de dados, pode-se dizer que as técnicas existentes são suficientes para as necessidades de suas aplicações.

Em relação a bancos de dados com suporte a tempo e versões, a manutenção de restrições de integridade é considerada mais complexa do que em bases tradicionais. Tradicionalmente, é necessário considerar um único estado (ou transição de estados) de versões únicas dos dados da base de dados. Em bancos de dados com suporte aos conceitos de tempo e de versão, deve-se considerar a possibilidade de que uma restrição abranja e/ou afete todos os estados da base desde sua criação e todas as versões de dados de cada um desses estados. Também devem ser considerados a temporalidade e o versionamento das próprias restrições. São raros os trabalhos relacionados à manutenção de restrições de integridade neste tipo de banco de dados. Além disso, a maioria dos trabalhos existentes não abrange todos os tipos possíveis de restrições e, muitas vezes, oferece suporte somente ao conceito de tempo ou somente ao conceito de versão. Dentre estes trabalhos, os de maior destaque são (BÖHLEN, 1994; CASTILHO; CASANOVA; FURTADO, 1982; CHOMICKI, 1992; CHOMICKI; TOMAN, 1995; COWLEY; PLEXOUSAKIS, 2000; DAYAL et al., 1988; DOUCET et al., 1997;

ESCOFET, 2001; MEDEIROS; JOMIER; CELLARY, 1992; PLEXOUSAKIS, 1993; RAM et al., 1997; SRINATH; RAMAKRISHNAN; RAM, 2000).

A realidade apresentada serve como maior motivação para o desenvolvimento deste trabalho. Acredita-se na grande necessidade de trabalhos que possibilitem formas completas de especificação e manutenção de restrições de integridade em bancos de dados com suporte a tempo e versões.

1.2 Objetivos e Contribuições

O principal objetivo deste trabalho é desenvolver técnicas que possibilitem a especificação de restrições de integridade e sirvam de base para o desenvolvimento de trabalhos relativos à sua manutenção em bases de dados com suporte a tempo e versões. Além de considerar o histórico de todas as versões dos dados da base durante seu tempo de vida, deve também ser considerada a utilização desses conceitos sobre as próprias restrições. Assim, é possível definir períodos onde uma determinada restrição deva ser verificada, trabalhar com diferentes versões de restrições e armazenar todo o seu histórico de alterações, de forma que os dados válidos em um determinado período sejam verificados pelas restrições existentes no mesmo período.

Em busca desse objetivo, é apresentada uma classificação completa de restrições de integridade em bancos de dados temporais de versões. Espera-se que essa classificação auxilie a definição, especificação e verificação de restrições, além de servir de base para processos de otimização em sua manutenção. Seu alvo central é o projetista do SGBD, pois com base na classificação, torna-se possível a criação de técnicas distintas para a manutenção de restrições com características comuns, obtendo-se vantagens sobre elas.

A fim de representar as restrições a serem verificadas sobre uma base de dados, são necessárias formas não ambíguas de especificação. Dessa forma, é apresentada uma linguagem de especificação de restrições de integridade, agregada ao Modelo Temporal de Versões (TVM) (MORO, 2001), um modelo de dados que utiliza, de uma forma uniforme, os conceitos de tempo e de versão. Depois de apresentada essa linguagem, seu poder de expressão é analisado com base na classificação proposta.

1.3 Organização do Texto

Esse trabalho está organizado da seguinte maneira. O segundo capítulo apresenta a base conceitual necessária ao entendimento dos conceitos de tempo e de versão, além de suas aplicações sobre bancos de dados. O terceiro capítulo apresenta uma introdução a restrições de integridade e suas formas de manutenção. O estado da arte referente a restrições de integridade com tempo e versões é apresentado através da exposição e comparação dos principais trabalhos relacionados ao tema, encontrados na literatura. O quarto capítulo apresenta uma classificação detalhada e sistemática de restrições de integridade em bases de dados temporais de versões. No capítulo seguinte, a abrangência da classificação é analisada, a fim de demonstrar que a mesma abrange todos os aspectos analisados por outras classificações, além de aspectos não considerados por elas. Com base nas classes de restrições definidas, o sexto capítulo define uma linguagem de especificação de restrições, agregada ao modelo de dados TVM. Finalmente, o sétimo capítulo apresenta um estudo de caso que ilustra a aplicação prática da classificação e da linguagem de especificação de restrições, enquanto o último capítulo apresenta conclusões e uma breve discussão sobre trabalhos futuros.

Além dos capítulos descritos, a BNF da linguagem de especificação de restrições é apresentada no Apêndice.

2 TEMPO E VERSÕES EM BASES DE DADOS

Este capítulo tem como principal objetivo apresentar a base conceitual necessária à classificação e à linguagem proposta para a especificação de restrições de integridade em bancos de dados temporais de versões. Serão apresentados os principais conceitos relacionados à utilização de tempo e de versões sobre bases de dados.

Deste modo, nas seções seguintes são apresentados: (i) o conceito de versão; (ii) o conceito de tempo; (iii) um modelo que apresenta conceitos de tempo e de versão, o TVM e sua linguagem de consulta, TVQL; e (iv) considerações finais.

2.1 O Conceito de Versão

Tem-se como versão de um objeto a descrição do mesmo em um determinado período de tempo ou sob um determinado ponto de vista. Segundo (GOLENDZINER, 1995), uma versão corresponde a um estado identificável de um objeto e deve ser tratada uniformemente em um modelo de dados. As diversas versões são geralmente organizadas através de um grafo acíclico dirigido, de acordo com a sua ordem de derivação.

Encontram-se na literatura diversos estudos relacionados ao conceito de versão, aplicados sobre modelos de dados relacionais como em (DADAM; LUM; WERNER, 1984) e orientados a objetos (AGRAWAL et al., 1991; BEECH; MAHBOD, 1988; BJÖRNERSTEDT; HULTÉN, 1989; CONRADI; WESTFECHTEL, 1998; GOLENDZINER, 1995; KATZ, 1990; KIM, 1989).

Em Golendziner (1995), o termo *objeto versionado* é definido e determina um objeto que possui diversas versões em um banco de dados. Todo objeto versionado possui uma *versão corrente*.

Normalmente, existe uma distinção entre dois tipos possíveis de operações de criação de versões: *criação para revisão* e *criação de alternativa* (também chamada de variante). Uma revisão corresponde a um avanço em um determinado conjunto de dados e sua criação depende sempre de um predecessor. Uma alternativa corresponde a uma variação funcionalmente similar de um determinado objeto, devida a diferentes pré-requisitos. Como por exemplo, têm-se pré-requisitos de desempenho. Na maioria dos casos, a escolha do tipo de criação de versões fica a cargo do usuário da base de dados (MEDEIROS; JOMIER; CELLARY, 1992).

O conceito de *configuração* é necessário quando o versionamento é aplicado a objetos compostos, ou seja, objetos formados pela união de diversos objetos mais simples chamados de módulos, e é permitido o versionamento desses módulos. Uma configuração de um objeto composto é uma coleção com uma versão de cada um de seus módulos. As versões que compõem uma configuração devem ser compatíveis e formarem juntas uma versão válida de um objeto composto. Uma nova versão ou configuração de um objeto composto não necessariamente inclui novas versões para

todos os módulos. Assim, certas versões de módulos que não tenham sido alteradas podem pertencer a mais de uma configuração do objeto composto.

Existem diversas aplicações para o conceito de versões em bancos de dados como: (i) manter a história de um objeto; (ii) tratar problemas de mudanças nas definições de tipos do banco de dados; (iii) manter a confiabilidade dos dados em ambientes onde existe concorrência; (iv) aumentar o desempenho em Sistemas Distribuídos; (v) tratar mudanças no esquema do banco de dados; (vi) representação de uma mesma informação em diferentes níveis, e (vii) oferecer suporte ao trabalho cooperativo (DITTRICH; LORIE, 1988).

2.2 O Conceito de Tempo

Um glossário de termos relacionados ao conceito de tempo, aplicado a bases de dados, foi proposto por (JENSEN et al., 1998) e complementado por (SNODGRASS, 2000). O conceito tem origem em uma tríade de triplas com elementos ortogonais, sendo descrita como: (i) tipos de dados temporais; (ii) tipos de tempo e, (iii) expressões temporais.

Segundo (EDELWEISS, 1998), os três tipos de dados temporais são:

- *instante no tempo* - o único tipo de dado temporal representado pela grande maioria dos sistemas gerenciadores de bancos de dados tradicionais. Ele representa um ponto no tempo. Um exemplo é o instante atual, representado pela palavra reservada *now*;
- *intervalo temporal* - representa o tempo decorrido entre dois instantes no tempo e, portanto, possui limites inferior e superior que podem ser abertos ou fechados;
- *elemento temporal* - representa uma união finita de intervalos no tempo. Esses intervalos podem ou não ser disjuntos, fato este, que enriquece muito o seu poder de expressão. O elemento temporal engloba todos os tipos de dados temporais anteriores.

Os três tipos de tempo existentes são:

- *tempo definido pelo usuário* - representando propriedades temporais definidas explicitamente pelo usuário e manipuladas pelos programas de aplicação;
- *tempo de transação* - o instante em que um dado foi inserido no banco de dados;
- *tempo de validade* - o tempo em que a informação é válida na realidade modelada.

Snodgrass (2000) define os três tipos básicos de afirmações ou expressões temporais: (i) *corrente:* que representa o estado atual dos dados. Um exemplo de expressão corrente é: *"Qual o salário de José agora ?"*; (ii) *seqüenciada:* que representa o estado de um determinado dado ou conjunto de dados em um instante no tempo definido. Como exemplo se tem: *"Qual era o salário de José em janeiro de 2000?"* e, (iii) *não seqüenciada:* que representa os dados ignorando-se o tempo. Por exemplo, *"Quais os valores do salário de José em qualquer tempo ?"*.

Jensen et al. (1998) ainda define diferentes formas de ordenação temporal:

- *tempo linearmente ordenado* - quando existe uma total ordenação entre quaisquer dois pontos no tempo;
- *tempo ramificado* (Figura 2.1) - onde dois ou mais pontos diferentes podem ser sucessores ou antecessores imediatos de um determinado ponto no tempo;

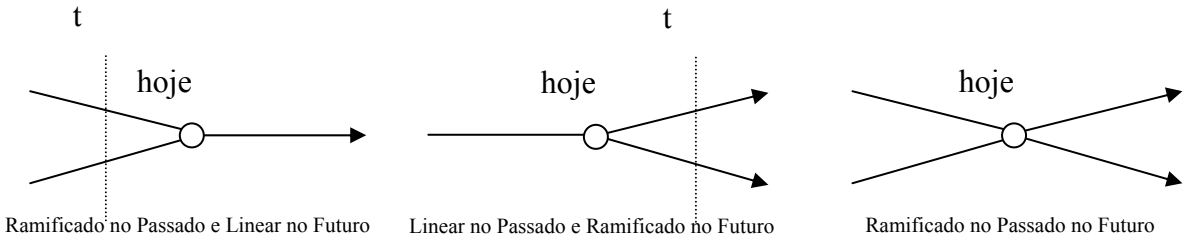


Figura 2.1: Exemplos de tempo ramificado

- *tempo circular* (Figura 2.2) - que é utilizado para modelar eventos recorrentes, ou seja, eventos que se repetem no tempo. Como por exemplo um mês.

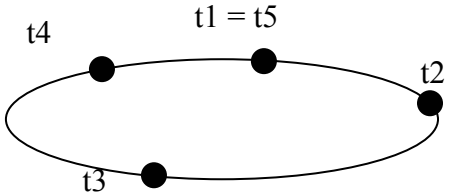


Figura 2.2: Exemplos de tempo circular

Nota-se ainda que, o tempo é contínuo por natureza. Isto significa que sempre existe um ponto no tempo entre dois outros pontos no tempo quaisquer. Para possibilitar a representação de tempo em informática, é necessária a introdução do conceito de *tempo discreto*, onde se tem uma linha composta por uma seqüência de intervalos temporais consecutivos que não podem ser decompostos. A esses intervalos dá-se o nome *chronons* e o valor temporal (segundo, minuto, dia, ano, etc.) relacionado a ele deve ser definido de forma a melhor atender as necessidades de cada aplicação.

Através da aplicação do conceito de tempo em um SGBD, torna-se possível armazenar o histórico de alterações de suas bases de dados. Conseqüentemente, não existe a possibilidade de perda de dados pois, mesmo que haja uma alteração na base, os antigos valores não serão sobrepostos pelos novos. Isso é possível pois os dados temporais, como o próprio nome indica, têm uma ligação com algum período, elemento ou instante de tempo.

Bancos de dados temporais são comumente classificados como: instantâneos (*snapshot*), de tempo de transação (*rollback*), de tempo de validade (*histórico*), bitemporais (*temporal*), que contêm tempo de validade e de transação, e multitemporais, que contém os três tipos possíveis de tempo (EDELWEISS, 1998). É importante notar que a associação do tempo de validade e do tempo de transação permite a reconstrução do estado da base de dados em qualquer instante do passado sem a necessidade de operações do tipo *backup* e *recovery*.

2.3 O Modelo Temporal de Versões

Como exemplo de aplicação dos dois conceitos discutidos nas seções anteriores, o Modelo Temporal de Versões (TVM – *Temporal Versions Model*) proposto por Moro em (MORO, 2001; MORO; SAGGIORATO; EDELWEISS; SANTOS, 2001) será apresentado nesta seção. O TVM é um modelo de dados orientado a objetos que suporta de forma uniforme os conceitos de tempo e de versão. Através do conceito de versão o usuário tem a possibilidade de armazenamento de alternativas de projeto, além de diferentes níveis de detalhamento com a utilização de objetos compostos. Com a

utilização da dimensão temporal, é possibilidado o armazenamento de todo o histórico e a evolução dos dados das instâncias dos objetos.

2.3.1 Características Gerais

O TVM (MORO, 2001) é uma extensão ao modelo de versões proposto por Golendziner em (GOLENDZINER, 1995), que possibilita a representação de toda a história dos dados de uma aplicação. A Figura 2.3 ilustra a hierarquia de classes do modelo.

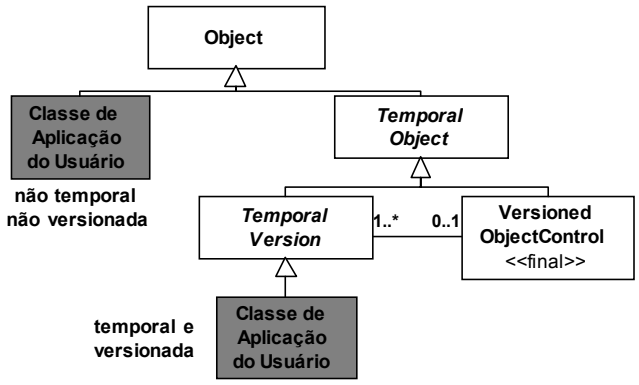


Figura 2.3: Hierarquia de classes do TVM

O modelo permite dois tipos distintos de objetos, pois existem apenas duas classes de aplicação do usuário. O primeiro tipo de objetos herda suas características diretamente da classe *Objetct* e, portanto, não possui implementados os conceitos de versão e de tempo. O segundo tipo de objetos herda tanto as características temporais como as de versões.

Duas formas de exclusão de versões podem ser utilizadas: (i) exclusão lógica, onde a versão excluída é somente desativada e seus dados não são perdidos e, (ii) exclusão física, conhecida como *vacuuming* (JENSEN, 1999), onde os dados da versão são realmente perdidos. A exclusão física vai contra conceitos básicos de bancos de dados temporais, mas é necessária em casos especiais, por exemplo, quando existe a necessidade de diminuição de dados armazenados por falta de espaço físico de armazenamento ou por questões de segurança. Nota-se que, para efetuar este tipo de exclusão, o usuário deve estar ciente que os dados serão realmente perdidos.

2.3.2 Características Específicas da Gerência de Tempo

No TVM, o tempo é associado a objetos, versões, atributos e relacionamentos, permitindo assim uma modelagem mais flexível (MORO et al., 2001). É permitida a utilização de duas formas de ordenação temporal, o *tempo ramificado* para um objeto e o *tempo lineramente ordenado* para suas versões. Assim, um objeto tem uma linha de tempo para cada uma de suas versões, possibilitanto a coexistência de um mesmo objeto em um ponto qualquer no tempo. A representação de tempo no TVM é discreta com a utilização do conceito de *chronons* e o tipo de dados temporais utilizado é o *elemento temporal.*

O modelo TVM é definido como *bitemporal*, tratando de tempo de transação e de tempo de validade. Tanto atributos como relacionamentos de um objeto temporal versionado podem ser definidos como *estáticos* ou *temporalizados* e, nada impede que uma mesma classe tenha atributos e relacionamentos de ambos os tipos.

Algumas restrições de integridade foram definidas para manter a integridade dos tempos de vida dos objetos, versões e objetos versionados no TVM. De acordo com essas regras, o tempo associado a suas versões deve estar contido no tempo de vida do objeto versionado, e o tempo associado às variações dos atributos temporalizados de uma versão deve estar contido no tempo de vida da versão.

Também foram definidas algumas regras em relação à inserção e atualização de valores de atributos temporalizados, e à exclusão de objetos. Toda informação que for inserida (primeira definição de um valor para um atributo) em uma base de dados bitemporal deve receber seus tempos de transação e de validade. Nunca há perda de informação na atualização de dados. No caso da exclusão lógica de objetos, a versão é desativada e seu tempo de vida é finalizado.

2.3.3 Características Específicas da Gerência de Versões

As versões em uma aplicação representam alternativas de projeto e a evolução de seus dados. Cada versão possui um estado que reflete seu estágio de desenvolvimento e/ou consistência durante seu tempo de vida (MORO et al., 2001). De acordo com o estado de cada versão é definido um conjunto de operações possíveis sobre a mesma. A Figura 2.4 ilustra os estados e transições de estado possíveis para uma versão qualquer.

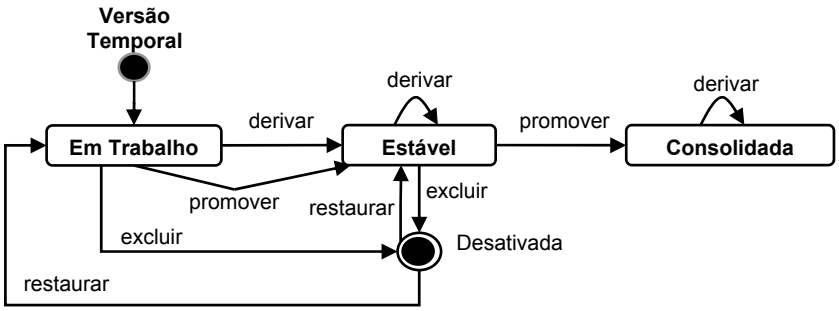


Figura 2.4: Diagrama de estados de uma versão

Quando criada, toda versão está no estado *em trabalho*. Este estado pode se alterar de acordo com a ocorrência de operações sobre a versão. A Tabela 2.1 define as operações possíveis e impossíveis de serem executadas quando uma versão está em cada um dos estados existentes.

Tabela 2.1: Estados e operações sobre versões

Estado / Operação	Em trabalho	Estável	Consolidada	Desativada
Derivar	Sim	Sim	Sim	Não Definida
Promover	Sim	Sim	Não Definida	Não Definida
Alterar	Sim	Não	Não	Não
Excluir	Sim	Sim*	Não	Não
Consultar	Sim	Sim	Sim	Sim
Compartilhar	Não	Sim	Sim	Não
Restaurar	Não Definida	Não Definida	Não Definida	Sim

* Caso não possua Sucessora

2.3.4 Linguagem de Consulta (TVQL)

Nesta seção são apresentadas as principais características da linguagem de consulta a bancos de dados temporais e de versões TVQL (*Temporal Versioned Query Language*), definida para contemplar os aspectos temporais e de versionamento do modelo TVM. Essa linguagem, proposta por Moro e outros em (MORO et al., 2001; MORO;

EDELWEISS; ZAUPA; SANTOS, 2002), é uma extensão da linguagem SQL que visa contemplar, além dos aspectos tradicionais, aspectos temporais e de versões.

Vale ressaltar a importância da linguagem TVQL em relação ao trabalho como um todo, pois a mesma é utilizada como base para a linguagem de especificação de restrições de integridade proposta para o modelo TVM. A seguir são apresentadas as características da TVQL que a diferem da linguagem SQL, permitindo consultas sobre dados temporais e de versões.

2.3.4.1 Características Relacionadas ao Conceito de Tempo

A palavra reservada EVER é uma das características temporais mais importantes da TVQL. Ela foi definida para considerar todos os valores do histórico de vida de objetos e versões e pode ser utilizada nas cláusulas SELECT ou WHERE. Caso uma consulta TVQL não apresente nenhuma condição temporal específica em suas cláusulas SELECT ou WHERE, a mesma será efetuada considerando somente os dados atuais dos objetos ativos, ou seja, os objetos que não foram excluídos.

Quando a palavra reservada EVER é utilizada na cláusula SELECT, a consulta retorna todo o histórico das propriedades temporais selecionadas e considera o histórico das propriedades temporais que forem mencionadas na cláusula WHERE. Já quando EVER é utilizada na cláusula WHERE, são selecionados os valores atuais, mas na cláusula condicional é considerado todo o histórico das propriedades temporais.

Considere as duas consultas seguintes sobre a classe *Pais* que é temporal versionada. Os atributos *regime_politico*, *nome* e *renda_per_capita* fazem parte de seu conjunto de atributos e somente o atributo *nome* é do tipo não temporal:

```
select nome, regime_politico
from Pais
where renda_per_capita >= 2000;

select ever nome, regime_politico
from Pais
where renda_per_capita >= 2000;
```

A primeira consulta retorna os nomes e regimes políticos *atuais* de todos os países que têm, *na data da consulta*, uma renda per capita maior ou igual a dois mil. A segunda consulta retorna os nomes e o *histórico* de regimes políticos de todos os países que *têm ou já tiveram* uma renda per capita maior ou igual a dois mil. Considere agora a seguinte consulta sobre a mesma classe *Pais*:

```
select nome, regime_politico
from Pais
where ever renda_per_capita >= 2000;
```

O resultado desta consulta é o conjunto de nomes e regimes políticos *atuais* de todos os países que *têm ou já tiveram* uma renda per capita maior ou igual a dois mil.

Outra possibilidade é a seleção de acordo com valores atuais e históricos ao mesmo tempo. Segundo a regra geral, após a palavra reservada EVER, são considerados os históricos de todas as propriedades temporais mencionadas. Existe uma forma de anular essa regra através da função PRESENT, que considera os valores atuais das propriedades que estiverem dentro de sua expressão parâmetro na cláusula WHERE. Um exemplo de utilização desta função ocorre quando se deseja consultar os nomes e o *histórico* de regimes políticos de todos os paises que possuem, *atualmente*, renda per capita maior ou igual a dois mil. Essa consulta pode ser vista a seguir:

```
select ever nome, regime_politico
from Pais
where present(renda_per_capita >= 2000);
```

2.3.4.2 Características Relacionadas ao Conceito de Versão

Em uma consulta TVQL, o usuário define na cláusula FROM os objetos e versões a serem consultados. Os objetos e versões correntes são especificados pelo nome da classe à qual pertencem, enquanto todas as versões de todos os objetos de uma classe são consideradas quando é acrescentada a palavra reservada VERSIONS após o nome da classe. São possíveis consultas sobre os estados das versões, as versões correntes, uma ou várias versões de um objeto, a navegação na hierarquia e as configurações.

2.3.4.3 Características Gerais

A linguagem TVQL permite consultas que não envolvam os conceitos de tempo e de versão, consultas que envolvam apenas um desses conceitos e consultas que envolvam os dois conceitos em conjunto. A seguir, é apresentada de uma forma sucinta a sintaxe da linguagem TVQL. A sintaxe completa da mesma pode ser encontrada em (MORO et al., 2001).

```
<query> ::= select [ ever ] [ distinct ]  <targetC> { ",” <targetC> }*
   from <identificC> { ",” <identificC> }* [ where [ ever ] <searchC> ]
   [ group by <groupC> { ",” <groupC> }*  [ having <logicalExpr> ] ]
   [ order by <orderC> { ",” <orderC> }*  [ <setOp> <query> ] ] ";”
<targetC> ::= ( "*” | <propertyName> | <aggregationFunctions> |
   <preDefInterval> | <preDefInstant> )  [ as <identifier> ]
<identificC> ::= <className> [ .versions ] [ <aliasName> ]
<searchC> ::= ( <logicalExpr> | <tempExpr> )
<groupC> ::= ( <propertyName> | <preDefInterval> | <preDefInstant> )
```

2.4 Considerações Finais

Nesse capítulo foi apresentada a base conceitual necessária à classificação e especificação de restrições de integridade que utilizam o conceito de tempo e de versão. Como exemplo, um modelo de dados que aplica esses conceitos (TVM) e sua linguagem de consulta (TVQL) foram apresentados.

Depois de apresentada essa base conceitual, é possível a descrição das principais diferenças estruturais entre um SGBD que suporta os conceitos de tempo e de versão e os tradicionais. Percebe-se facilmente a necessidade de uma estrutura muito mais complexa para a implementação, implantação e manutenção de um SGBD que aplica os conceitos de tempo e versão do que a existente um SGBD tradicional. Essas diferenças vão desde a forma com que os dados devem ser fisicamente armazenados até a forma com que eles devem ser representados e interpretados pelo SGBD. Apesar desta maior complexidade de um SGBD que suporta os conceitos de tempo e de versão, acredita-se que sua utilização seja viável e necessária, visto que este tipo de SGBD facilita grandemente o trabalho do desenvolvedor no nível de aplicação em realidades com necessidades que vão além das tradicionais.

3 RESTRIÇÕES DE INTEGRIDADE COM TEMPO E VERSÕES

Este capítulo apresenta uma introdução a restrições de integridade sobre bases de dados temporais e de versões, bem como o estado da arte deste tema através da análise dos principais trabalhos encontrados na literatura.

3.1 Restrições de Integridade: conceitos e terminologias

Integridade é uma propriedade de um banco de dados que se refere à validade do seu conteúdo e à forma pela qual o mesmo foi alcançado. Segundo (SANTOS, 1980), uma restrição de integridade é uma regra que restringe o conjunto de estados íntegros e/ou o conjunto de transições válidas em um banco de dados. Tradicionalmente estas restrições são utilizadas para garantir que o conteúdo de uma base de dados descreva de forma precisa a realidade modelada (DOUCET et al., 1997).

Restrições de integridade sobre bancos de dados temporais definem condições que devem ser respeitadas em qualquer tempo. Essas restrições podem referenciar tipos distintos de tempo: tempo de transação, tempo de validade ou ambos. Além disso, as alterações sobre um banco de dados bitemporal envolvem obrigatoriamente o tempo de transação corrente, mas podem afetar dados validos no passado, no presente ou no futuro. Dessa forma, a manutenção da integridade dos dados deve considerar, além do estado de tempo de transação atual, todos os estados de tempo de validade no passado, no presente e no futuro da base de dados.

O conceito de tempo pode ainda estar relacionado às restrições em si, não aos dados por elas referenciados. Considerando seu tipo de tempo e abrangência temporal, existe a possibilidade de definição de períodos onde uma determinada restrição deva ser verificada e de armazenagem de seu histórico de alterações. Dessa forma, por exemplo, os dados válidos em um determinado período no passado ou no futuro podem ser verificados pelas restrições válidas no mesmo período.

Em bases de dados de versões, o estado atual pode englobar versões distintas de um mesmo conjunto de dados de acordo com alternativas de projeto. Considerando sua abrangência de versionamento, restrições de integridade podem estar relacionadas a objetos e todas as suas versões (objetos versionados) ou apenas a versões específicas dos mesmos. O conceito de versão também pode ser aplicado às próprias restrições de integridade, possibilitando a existência de versões distintas para uma mesma restrição. Por exemplo, uma restrição pode assumir uma forma específica para versões de objetos em fase de desenvolvimento e outra para as demais versões de objetos.

São raros e insuficientes os trabalhos relacionados tanto à modelagem quanto à verificação de restrições neste tipo de base de dados. Em sua maioria, os trabalhos existentes não abrangem todos os tipos possíveis de restrições de integridade e, muitas

vezes, oferecem suporte somente ao conceito de tempo ou somente ao conceito de versão.

Boa parte desses trabalhos segue uma ordem lógica que divide em partes o processo de desenvolvimento de métodos para a manutenção da integridade de dados: (i) definição de uma classificação agrupando os tipos possíveis de restrições com características comuns; (ii) criação de métodos de especificação para as classes de restrições definidas; (iii) escolha de estratégias e técnicas que permitam a validação dos tipos de restrições classificadas, de acordo com características específicas das restrições de cada classe, e (iv) otimização do processo de validação dessas restrições.

As seções seguintes analisam diversos trabalhos, apresentando a classificação proposta por eles e uma visão geral sobre os métodos que utilizam para a manutenção da integridade de dados. A última seção apresenta conclusões e uma análise que visa comparar os trabalhos descritos em relação ao trabalho aqui proposto.

3.2 Trabalho de DOUCET, Anne et al.

Doucet e outros apresentam diversos trabalhos relacionados à manutenção de integridade de bases de dados sobre um modelo de versões específico (DOUCET et al., 1996a; DOUCET et al., 1996b; DOUCET; MONTIES, 1997). Basicamente, este conjunto de trabalhos apresenta um *framework* para a manutenção de restrições de integridade especificadas através de lógica de primeira ordem. A manutenção dessas restrições é discutida sobre a execução de transações ACID.

Com base no *framework* criado, Doucet passa a considerar a manutenção de restrições temporais. O trabalho (DOUCET et al., 1997) é o único encontrado na literatura que considera ambos conceitos de tempo e de versão sobre restrições de integridade. Doucet e outros propõem a utilização de um modelo de versões como ferramenta para a implementação de dimensões temporais em bases de dados, além da manutenção de restrições de integridade sobre essas bases.

Restrições sobre versões de objetos (multiversionadas) são classificadas de acordo com sua abrangência (*scope*) e ligações (*binding*) sobre objetos, classes e versões de objetos. São ainda consideradas a abrangência de estados e algumas características semânticas dessas restrições. Entretanto, características importantes não foram tratadas, como a temporalidade e o versionamento da própria restrição (apenas citados), sua abrangência temporal, tipos temporais e de versionamento, granularidade temporal, entre diversas outras.

A especificação das restrições consideradas por Doucet é feita através de uma lógica temporal, baseada em lógica de primeira ordem. As restrições especificadas em lógica temporal são então mapeadas para restrições em lógica de primeira ordem (*mv-constraints*) sobre o modelo de versões utilizado. Detalhes do processo de verificação dessas restrições não são apresentados. Apenas alguns comentários enfatizam que elas devem ser verificadas somente em momentos específicos e que os algoritmos de verificação devem levar em conta os aspectos e classes definidos em sua classificação de restrições.

3.3 Trabalho de RAM, D. Janaki et al.

Ram e outros (RAM et al., 1997) propõem um modelo de dados orientado a objetos específico para a gerência de processos distribuídos de *design* colaborativo. Esse modelo é baseado em objetos comuns e objetos restritivos (*constraint meta-objects*). Cada objeto comum é associado a um espaço restritivo (*constraint space*) composto por

um ou mais objetos restritivos. A colaboração entre os membros de uma equipe de *design* é modelada através de dependências sobre diversos espaços restritivos, enquanto a atividade colaborativa é proporcionada através de transações que satisfazem as restrições locais e as propagam entre todos os espaços restritivos inter-relacionados. A especificação dessas restrições é feita através de uma adaptação da linguagem de programação C++, utilizada também para a definição das classes presentes na base de dados.

Esse trabalho apresenta uma classificação interessante de restrições de integridade, considerando inicialmente sua abrangência, por eles chamada de espaço de validação. As possíveis dependências entre restrições incompletas também são consideradas. Além disso, uma certa ordenação entre o processo de verificação de restrições é levada em conta através da classificação de restrições que devem notificar sua verificação a outras e restrições que devem aguardar notificações para iniciar sua validação.

Apesar de ser extremamente importante para aplicações de *design*, o versionamento de objetos é tratado de forma superficial neste trabalho, enquanto a temporalidade não é nem mesmo citada. Assim, com base nesse modelo, Srinath, Ramakrishnan e Ram (2000) propõem um modelo para a gerência de versões em projetos de *design*. Esse modelo propõe técnicas para a propagação de mudanças sobre versões de objetos armazenando os diferentes significados semânticos que cada versão possui sobre um projeto. A gerência de configurações também é levada em conta nesse contexto. Entretanto, a manutenção de restrições de integridade não é tratada no modelo.

3.4 Trabalho de DAYAL, Umeshwar et al.

Em (DAYAL et al., 1988), Dayal apresenta o projeto HiPAC (High *Performance ACtive database system*) que propõe a manutenção de restrições de integridade temporais (*timing constraints*) através de regras de condição-ação. Um modelo de conhecimento é proposto a fim de prover as primitivas necessárias para a definição de restrições temporais, regras de condição-ação e eventos previsíveis.

Inicialmente é apresentada uma classificação abrangente de restrições, considerando características como sua natureza, ordem de precedência de validação, noções de tempo, previsibilidade de eventos, escopo de avaliação, abrangência de estados, interdependência e pontos de verificação. Entretanto, nenhuma característica de versionamento é considerada.

Não é apresentado nenhum método particular para a especificação das restrições consideradas por Dayal. Entretanto, sua validação é discutida a fundo através da utilização de gatilhos. Com base na classificação previamente apresentada, técnicas de validação são propostas de acordo com diversas características de restrições temporais. Por fim, é proposta a arquitetura do projeto e são feitas diversas considerações relacionadas à performance do sistema.

3.5 Trabalho de CASTILHO, CASANOVA e FURTADO

Castilho, Casanova e Furtado (1982) propõem um *framework* temporal para a especificação de restrições de integridade em de bases de dados, considerando restrições com abrangência de estados estática e de transição.

Dois níveis de especificação são definidos. No primeiro nível, a descrição da base de dados não indica as formas como ela deverá ser atualizada. Restrições de transição são então especificadas, com a utilização de uma variante de lógica temporal proposta por eles. No segundo nível de especificação, a descrição da base de dados inclui um

conjunto pré-definido de operações de atualização que, por convenção, deve ser utilizado por qualquer transação que precise atualizar os dados. Com a utilização desse conjunto pré-definido de operações de atualização, nenhuma violação de restrição pode ocorrer. Logicamente, essa afirmação é verdadeira considerando apenas a existência de restrições estáticas e de transição.

O conceito de especificação em dois níveis é considerado importante por permitir a utilização de diferentes níveis de abstração. O primeiro nível serve principalmente para a documentação relacionada ao comportamento desejado da base de dados. Diferentemente, o segundo nível está intimamente ligado à implementação da base de dados desejada e contém um conjunto pré-definido de operações, baseadas na utilização de gatilhos, que permitem a criação e a atualização dos dados.

Com base no texto é possível extrair classificações de restrições de integridade quanto à sua abrangência de estados e temporalidade. Vale ainda ressaltar que nenhuma característica relacionada ao versionamento de dados foi considerada.

3.6 Trabalho de BÖHLEN, Michael H.

Böhlen (BÖHLEN, 1994) analisa a manutenção da integridade de bases de dados temporais em relação a restrições de tempo de validade. A manutenção relacionada a essas restrições é mais difícil do que a manutenção de restrições de tempo de transação, pois é permitido ao usuário alterar qualquer um dos estados de uma base de dados de tempo de validade.

Primeiramente, Böhlen apresenta uma classificação de restrições de integridade sobre bases de dados temporais, por ele chamada de taxonomia. De acordo com essa classificação restrições podem ser temporais ou não temporais. Restrições temporais podem estar relacionadas ao tempo de validade, tempo de transação, ou ambos tipos de tempo. Finalmente, de acordo com suas abrangências de estados, essas restrições são sub-classificadas dando origem a sete novas classes de restrições. Vale ressaltar que não foi considerada nenhuma característica relacionada ao conceito de versão.

Após classificadas as restrições, Böhlen utiliza a linguagem de consulta temporal ChronoLog (BÖHLEN; MARTI, 1994), baseada em lógica de primeira ordem, a fim de permitir a especificação de restrições de tempo de validade. Através dessa técnica, o usuário pode até especificar o período de validade de uma restrição de integridade, ou seja, o período em que ela deverá ser respeitada pelos dados da base.

Por fim, diversas características inerentes a restrições de tempo de validade são discutidas, a fim de adaptar métodos incrementais de manutenção de consistência para manter esse tipo de restrição.

3.7 Trabalho de ESCOFET e SISTAC

Em seus trabalhos (ESCOFET; SISTAC, 1994; ESCOFET; SISTAC, 1996a; ESCOFET; SISTAC, 1996b; ESCOFET, 2001), Escofet e Sistac analisam restrições de integridade temporais com o objetivo de obter vantagens para sua manutenção em bases de dados bitemporais e dedutivas. Características relacionadas ao conceito de versão não são consideradas.

Escofet (2001) apresenta uma classificação de restrições temporais, por ela chamada de taxonomia, a fim de definir as melhores formas para especificá-las e mantê-las. De acordo com sua abrangência de estados e o tipo de tempo que utilizam, restrições são classificadas em dez classes distintas que são discutidas em detalhe posteriormente. Além disso, um segundo critério de classificação considera diversas características

semânticas de restrições sobre distintos tipos de tempo, permitindo a criação de sete novas classes de restrições. Essa classificação se baseia fortemente nos trabalhos de Böhlen (1994) e Doucet (DOUCET et al., 1997).

Detalhes referentes à especificação de restrições não são apresentados, mas é considerada a utilização de lógica de primeira ordem. Entretanto, a manutenção dessas restrições é bem analisada pelos autores.

Antes da análise para a manutenção, é apresentado um estudo de transações sobre bases de dados temporais dedutivas. Como resultado, é introduzido o conceito de transação coerente que permite a rejeição de transações (não coerentes), sem a necessidade de demais análises.

Para a manutenção de restrições sobre a execução de transações coerentes, é derivado um conjunto de regras de transição de acordo com o tipo de tempo utilizado. Devido ao demasiado número de regras criadas, com base na classificação proposta, um estudo detalhado dessas restrições apresenta uma redução considerável no número de regras. Finalmente, são apresentados exemplos de aplicação do método proposto.

3.8 Trabalho de CHOMICKI e TOMAN

Chomicki (1992) propõe a utilização de uma linguagem de lógica temporal de passado (*Past Metric Temporal Logic*) para a especificação de restrições de integridade de tempo real. Também são propostos métodos de avaliação dessas restrições sobre bases de dados não temporais (*History-less Methods*). Além disso, é discutida brevemente a possibilidade de implementação de restrições de tempo real através de regras de condição-ação com condições temporais.

Em trabalhos subseqüentes (CHOMICKI, 1995; CHOMOCKI; TOMAN, 1995), Chomicki e Toman propõem uma arquitetura de manutenção de restrições temporais sobre bancos de dados ativos. Essa arquitetura considera a utilização de um compilador que permite a transformação automática de restrições de integridade em um conjunto de regras de condição-ação (gatilhos) aplicáveis a um banco de dados relacional. Essa arquitetura pode ser utilizada sobre diversos sistemas de bancos de dados ativos e linguagens de especificação de restrições temporais. A implementação apresentada considera a especificação de restrições em lógica temporal de passado (*Past Temporal Logic*) e os sistemas Starburst ou INGRES.

São considerados a abrangência de estados, o tipo e a condição temporal de restrições de integridade. Além disso, também é considerada a possibilidade de restrições que possuam ou não conjuntos de ações de violação (*Side-Effect*). Vale notar que características de versionamento e a temporalidade da própria restrição não são consideradas em momento algum.

3.9 Trabalho de MEDEIROS, JOMIER e CELLARY

Medeiros, Jomier e Cellary (1992) analisam o problema da manutenção de restrições de integridade sobre bases de dados de versões. O principal intuito é permitir a manutenção automática de certos tipos de restrições não suportadas por um SGBD tradicional que, conseqüentemente, acabam sendo mantidas por programas de aplicação. Vale notar que nenhuma característica temporal é considerada.

Primeiramente são apresentadas as principais características desejáveis a métodos tradicionais de manutenção de restrições de integridade e mecanismos de controle de versionamento. Logo após, são sistematicamente discutidos os possíveis problemas decorrentes da integração de restrições de integridade e do controle de versionamento

sobre bases de dados. Nessa discussão, além da manutenção de consistência entre configurações, também são considerados problemas relacionados às alterações sobre visões de usuários e à evolução e propagação de restrições.

Após essa discussão, a fim de permitir a manutenção de restrições de integridade sobre bases de dados de versões, um *framework* é apresentado. Uma de suas características mais interessantes é que, além do versionamento de partes da base de dados, também é considerada a existência e a manutenção de diferentes versões de uma mesma restrição, de acordo com a realidade. Durante a apresentação do *framework*, uma nova classe de restrições de integridade é definida. Assim, restrições sobre o controle de versões são restrições que estabelecem regras para a criação e manutenção de relacionamentos entre versões de objetos.

Quanto ao alcance de restrições, são consideradas as suas abrangências de estados e as suas abrangências de versionamento. Além disso, considera-se que restrições podem ser dependentes da aplicação, do modelo de dados ou do controle de versionamento. Elas podem ainda ser definidas sobre o conteúdo dos dados, sobre o comportamento ou sobre a estrutura do esquema. Também é considerada a possibilidade de versionamento da própria restrição.

3.10 Trabalho de PLEXOUSAKIS e COWLEY

Nos trabalhos (PLEXOUSAKIS, 1993; PLEXOUSAKIS, 1995), Plexousakis trata a manutenção de restrições semânticas sobre bases de conhecimento temporais e dedutivas. Seu principal objetivo é a definição de técnicas para a simplificação deste tipo de restrições a fim de facilitar o processo de manutenção de integridade.

Regras dedutivas e restrições de integridade são representadas através da linguagem Telos (MYLOPOULOS et al., 1990), uma linguagem híbrida para a representação de conhecimento. Telos provê uma sub-linguagem de asserção capaz de especificar restrições de estado e dinâmicas. Restrições dinâmicas também podem ser de transação, ou epistêmicas, se referindo a dois ou mais estados epistêmicos da base (tempo de validade e tempo de transação).

Um método de compilação possibilita diversas transformações sintáticas, semânticas e temporais sobre restrições de integridade e regras dedutivas. O resultado da simplificação é então utilizado na criação de um grafo de dependência que permite apenas a execução de alterações válidas sobre os dados. A utilização de regras dedutivas para facilitar a validação de restrições de integridade também é discutida. Além disso, são propostos algoritmos para facilitar a atualização dos grafos de dependência de acordo com modificações nos conjuntos de restrições de integridade e regras dedutivas.

Em um novo trabalho, Cowley e Plexousakis (2000) consideram a manutenção de restrições de integridade sobre bases de dados com indeterminação temporal. Indeterminação temporal envolve incertezas nos tempos relacionados aos dados e a utilização de diferentes granularidades temporais em uma mesma base de dados. Nesse trabalho, é proposto um *framework* para a manutenção de integridade deste tipo de base de dados, baseado em uma álgebra para a especificação de intervalos de tempo com indeterminação e semânticas para a validação de restrições temporais em potencial.

Durante a simplificação e manutenção de restrições de integridade, são consideradas diversas características importantes, como a abrangência de estados, a dependência de restrições sobre regras dedutivas, a possibilidade de restrições sobre a estrutura ou sobre a semântica de bases de dados e a validação de restrições em potencial para bases com indeterminação temporal.

3.11 Considerações Finais

Este capítulo apresentou o estado da arte referente à manutenção da integridade de dados em bases temporais de versões. Diversos trabalhos relacionados ao tema foram analisados e suas principais características discutidas. Considera-se que as características mais importantes a serem consideradas são:

- público alvo, que pode ser aplicações em geral, aplicações específicas de *design*, bases de conhecimento (*knowledge bases*), entre outros;
- forma de apresentação da classificação, que pode estar explícita ou implicita;
- tipo de modelo de dados e banco de dados utilizados;
- métodos propostos para a especificação e manutenção de restrições;
- linguagem de atualização de dados utilizada para a definição de restrições com ações de violação;
- formas de otimização do processo;
- técnicas de propagação de mudanças sobre o conjunto de restrições de integridade durante o tempo de vida da base de dados;
- técnicas e ferramentas utilizadas na implementação, além do tipo de *hardware* utilizado;

De acordo com essas características, a Tabela 3.1 ilustra uma comparação entre todos os trabalhos apresentados no capítulo.

A realidade apresentada serve como maior motivação para o desenvolvimento do presente trabalho. Acredita-se que é imprescindível a criação de trabalhos que apresentem formas completas de modelagem e verificação de restrições de integridade em bancos de dados temporais de versões. Acredita-se também que trabalhos como este facilitarão consideravelmente a incorporação de um SGBD que suporta os conceitos de tempo e de versão às aplicações comerciais comumente baseadas em sistemas tradicionais.

Tabela 3.1: Comparação entre os trabalhos analisados

Características / Trabalhos Relacionados	DOUCET, Anne et al.	RAM, D. Janaki et al.	DAYAL, Umeshwar et al.	CASTILHO, CASANOVA e FURTADO	BÖHLEN, Michael H.
Público Alvo	Geral	Aplicações de *Design* Colaborativo	Bases de Conhecimento (*knowledge bases*)	Geral	Geral
Forma de Apresentação da Classificação	Explícita	Implícita	Implícita	Explícita	Explícita
Modelo de Dados	Orientado a Objetos	Orientado a Objetos	Orientado a Objetos e Semântico	ND	Relacional
Tipo de Banco de Dados	Temporal de Versões[1]	Tradicional ou Versionado[2]	Temporal e Ativo	Temporal	Temporal e Dedutivo
Método de Especificação de Restrições	Lógica Temporal	Linguagem baseada em C++	ND	Lógica Temporal	Linguagem de Consulta ChronoLog (BÖHLEN; MARTI, 1994)
Método de Manutenção de Restrições	Verifica apenas as restrições referentes aos dados afetados, após a execução de transações ACID.	Valida localmente restrições e posteriormente as propaga entre membros de *design*.	Transforma restrições em regras de condição-ação de um banco de dados ativo.	Com base nas restrições, cria um conjunto de operações possíveis de atualização de dados.	Baseia-se na adaptação de métodos incrementais de manutenção da integridade de bases tradicionais.
Linguagem de Atualização de Dados	ND	ND	Álgebra PROBE (MANOLA; DAYAL, 1986)	ND	ND
Otimização do Processo de Manutenção	Considera as características comuns de restrições (classificação proposta).	NC	Baseada em diferentes níveis de rigor na validação.	NC	Baseada em uma álgebra de intervalos temporais.
Técnica de Propagação de Mudanças em Restrições	NC	NC	NC	NC	NC
Implementação	Baseada no SGBD $O_{2..}$	Baseada em C++ e estações SUN.	Baseada em gatilhos (*triggers*).	Baseada em gatilhos (*triggers*).	Utiliza o SGBD Oracle, a linguagem SICStus-Prolog e plataformas UNIX.

ND – Não Definido NC – Não Considerado

[1] Não considera a utilização conjunta de ambos conceitos.

[2] O modelo de versões proposto não considera restrições de integridade.

Tabela 3.1: Comparação entre os trabalhos analisados (cont.)

Características / Trabalhos Relacionados	ESCOFET e SISTAC	CHOMICKI e TOMAN	MEDEIROS, JOMIER e CELLARY	PLEXOUSAKIS e COWLEY
Público Alvo	Geral	Geral	Aplicações de *Design* Colaborativo	Bases de Conhecimento (*knowledge bases*)
Forma de Apresentação da Classificação	Explícita	Implícita	Implícita	Implícita
Modelo de Dados	ND	Relacional	Orientado a Objetos	ND
Tipo de Banco de Dados	Temporal e Dedutivo	Ativo e de Tempo Real	Versionado	Temporal e Dedutivo[1]
Método de Especificação de Restrições	Lógica de Primeira Ordem	Lógica Temporal de Passado	ND	Linguagem de representação de conhecimento Telos (MYLOPOULOS et al., 1990)
Método de Manutenção de Restrições	Baseia-se no conceito de transações coerentes e na derivação de restrições em um conjunto de regras dedutivas de transição.	Baseia-se em métodos que não necessitam de dados históricos (*History-Less Methods*) e na transformação de restrições em gatilhos (*triggers*) de uma base relacional.	Baseada na manutenção de consistência de configurações. Considera também a evolução e propagação de restrições, a atualização de visões e o versionamento de dados, restrições e esquemas.	Efetua transformações sintáticas, semânticas e temporais sobre restrições e regras dedutivas a fim de criar grafos de dependência que permitam apenas alterações válidas sobre os dados.
Linguagem de Atualização de Dados	ND	SQL	ND	ND
Otimização do Processo de Manutenção	De acordo com características das restrições (classificação proposta), efetua a redução do conjunto de regras dedutivas de transição.	Considera métodos de otimização algébrica e contextual a fim de minimizar os dados auxiliares necessários à aplicação dos métodos propostos.	NC	Baseada em métodos para a simplificação das restrições e regras dedutivas.
Técnica de Propagação de Mudanças em Restrições	NC	NC	Baseada na criação de versões de esquemas que possuem suas próprias versões de restrições e na adaptação dos dados a essas versões.	Baseada em métodos de atualização dos grafos de dependência.
Implementação	Baseada em um sistema Prolog.	Utiliza o SGBD Starburst ou INGRES.	Baseada no SGBD O_2.	ND

ND – Não Definido NC – Não Considerado

[1] Com a possibilidade de utilização de tempo com indeterminação.

4 CLASSIFICAÇÃO DE RESTRIÇÕES DE INTEGRIDADE EM BANCOS DE DADOS TEMPORAIS DE VERSÕES

Este capítulo apresenta uma classificação de restrições de integridade em bases de dados temporais de versões. O principal objetivo desta classificação é disponibilizar uma análise profunda sobre diversas características das restrições aplicáveis a bancos de dados temporais de versões. Esta proposta poderá auxiliar trabalhos futuros de definição, especificação e verificação de restrições sobre bases de dados temporais de versões, além de servir de base para a definição de procedimentos de otimização na manutenção da integridade de seus dados.

É importante enfatizar que, apesar do fato que a classificação de restrições possa ser utilizada de diversas maneiras, seu principal alvo é o projetista do SGBD, pois com base na classificação, torna-se possível a criação de técnicas distintas para a manutenção de restrições com características comuns, de forma a se obter vantagens sobre elas.

4.1 Conceitos Introdutórios

Segundo (SANTOS, 1980), a especificação completa de uma restrição de integridade é composta pelos seguintes componentes: *conjunto restringente*, *conjunto restringido*, *condição restritiva* (*asserção*), *pontos de verificação* e *ações de violação*.

O *conjunto restringente*, ou simplesmente *restringente*, é composto por objetos, armazenados ou não na base de dados, que são utilizados no estabelecimento de condições a serem respeitadas pelos dados do banco de dados, os quais correspondem aos integrantes do *restringido*. Assim, o conjunto de objetos, armazenados na base de dados, cujos conteúdos são restringidos por uma restrição de integridade é conhecido como seu *conjunto restringido*, ou simplesmente, *restringido*.

Supõe-se neste trabalho que todos os objetos, restringidos diretamente por uma restrição de integridade, devem pertencer a um único domínio de esquemas, entidades, relacionamentos ou valor. Essa consideração também se faz válida quanto aos domínios relacionados à temporalidade e ao versionamento dos integrantes do restringido. Esta ressalva é considerada, pois restrições de integridade com restringidos de domínios múltiplos sempre podem ser decompostas em duas ou mais restrições de domínios únicos. Assim, critérios e sub-critérios relacionados diretamente ao restringido não classificam a homogeneidade, diferentemente dos relacionados ao restringente, que classificam a homogeneidade e sub-classificam restrições *homogêneas* de acordo com seus domínios. Restrições *heterogêneas* não são explicitamente sub-classificadas, mas poderiam ser, de acordo com as possíveis combinações das classes homogêneas.

Considera-se como *ponto de verificação* de uma restrição de integridade o momento em que se inicia o processo de verificação de validade da restrição em questão, em

relação à base de dados. A definição do conjunto de *pontos de verificação* relacionado a uma restrição de integridade deve obrigatoriamente ser efetuada antes de sua ativação, exceto em restrições de *acionamento manual*.

A *condição restritiva* de uma restrição de integridade consiste em uma expressão lógica que relaciona os integrantes do *restringente* e *restringido*. A manutenção da integridade dos dados da base de dados, em relação à restrição em questão, é efetuada com base na verificação da validade desta condição em todos os *pontos de verificação* da restrição.

As *ações de violação* correspondem às ações a serem executadas sobre a base logo após a violação da restrição em questão, a fim de restabelecer a integridade do banco de dados.

O termo restrição de integridade, considerado no presente trabalho, diz respeito não somente ao conceito pelo qual serão restringidos os dados de uma base de dados, pois além do conceito, esse termo diz respeito à forma com que ele será aplicado e à forma com que a integridade dos dados será assegurada. Para tanto, a classificação de restrições de integridade, em relação a alguns de seus aspectos e critérios, necessita também de características relacionadas à definição e à especificação da restrição, ao esquema utilizado para representar a realidade modelada e aos detalhes do Sistema Gerenciador de Bancos de Dados (SGBD) e do modelo de dados adotado.

Com base em seus componentes e nas considerações apresentadas, restrições de integridade devem ser classificadas pelos seguintes aspectos: (i) *origem*, que diz respeito ao agente que as originou; (ii) *substância*, que evidencia o tipo de condicionamento imposto por elas à evolução do banco de dados; (iii) *especificação*, que considera a forma com que as mesmas estão especificadas; (iv) *aplicação*, que analisa as formas de manutenção da integridade do banco de dados em relação às mesmas; (v) *temporalidade*, que diz respeito às características temporais referentes aos seus conjuntos restringidos e restringentes, e (vi) *versionamento*, que considera as características de versionamento relacionadas aos conjuntos restringidos e restringentes das mesmas.

É importante deixar claro que esta classificação poderá ser útil para qualquer trabalho relacionado à manutenção eficiente e eficaz da integridade dos dados de bases temporais de versões, independentemente de outras características do modelo de dados ou do SGBD adotado. Além disso, devido à existência de aspectos específicos para a temporalidade e o versionamento, a classificação pode ser facilmente adaptada para bancos de dados tradicionais ou que apliquem somente um desses conceitos.

A classificação também se faz importante no contexto de engenharia de software, pois ela pode ser vista como um passo importante em busca da manutenção automática de restrições que ainda precisam ser mantidas por aplicações ou programas de auditoria. Considerando possível essa manutenção automática, em conjunto com o suporte oferecido por um SGBD que considera os conceitos de tempo e de versão, projetos de software podem ser simplificados significativamente, visto que a manutenção da integridade de seus dados com características temporais de versões também será de responsabilidade do SGBD.

As seções seguintes se destinam a descrever detalhadamente cada um dos aspectos citados, além de todos os critérios, sub-critérios e classes a eles relacionados.

4.2 Origem

Conforme dito anteriormente, bases de dados representam parte do mundo real. Para que seja possível uma representação fiel da realidade modelada, seus dados devem

respeitar todas as restrições impostas pela realidade. Essas restrições podem ser provenientes de diversos agentes do mundo real, como por exemplo: a forma de trabalho da empresa em questão, leis que devam ser respeitadas, aspectos naturais, entre outros. De acordo com essa realidade, o aspecto *origem* da classificação diz respeito aos diferentes tipos de agentes que podem originar restrições de integridade em bases de dados temporais de versões.

Considerando o aspecto *origem*, são estabelecidas as seguintes classes de restrições: (i) *ambiente,* (ii) *empresa,* (iii) *modelo de dados*, e (iv) *implementação*, conforme ilustrado na Figura 4.1.

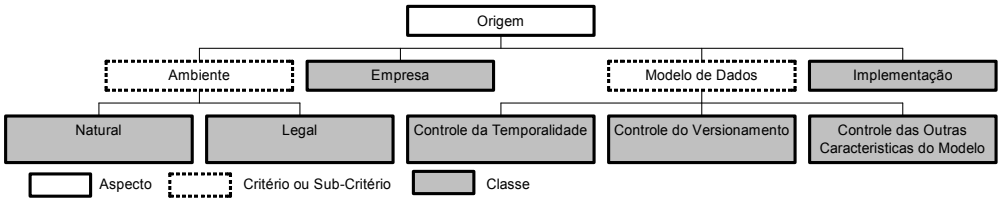


Figura 4.1: Aspecto *Origem*

Restrições de integridade com origem no *ambiente* dizem respeito às restrições impostas por agentes externos à empresa que devem ser representadas e respeitadas em seu banco de dados. Restrições desta classe podem ainda ser divididas em restrições provenientes de fontes *naturais* e de fontes *legais*.

Restrições com origem na *empresa* são impostas pela forma de funcionamento da organização à qual a base de dados estará representando. Mesmo em modelos de dados que representem realidades parecidas, é grande a possibilidade de que as restrições desta classe não sejam idênticas. Vale ainda ressaltar que costuma ser proporcional a quantidade e o detalhamento das restrições originadas pela empresa em relação ao nível de automação e complexidade de sua implementação e verificação na base de dados resultante.

Segundo Santos (1980), restrições de integridade originadas pelo *modelo de dados* surgem como conseqüência de sua inadequação à informação a ser representada, ou de um mau projeto do esquema. A fim de exemplificar essa classe de restrições, considere uma modelagem de dados baseada no modelo relacional, onde é necessária a representação da realidade através de um conjunto de relações e relacionamentos entre as mesmas. Neste tipo de modelagem é muito comum a utilização do conceito de chave estrangeira para a representação dos relacionamentos entre relações. Este conceito toma como base uma restrição de integridade, originada no *modelo de dados*, comumente conhecida como integridade referencial.

Considerando que o foco principal deste trabalho é a classificação de restrições de integridade de acordo com suas características temporais e de versionamento, considera-se que restrições de integridade originadas no *modelo de dados* podem ser provenientes do controle de características da *temporalidade*, do *versionamento* ou de *outras características* do mesmo.

As restrições originadas na *implementação* têm como principal objetivo a adequação dos dados às estruturas de dados fisicamente utilizadas para o armazenamento e recuperação de dados na base de dados. O tipo mais comum de restrições desta classe é o que estabelece um determinado domínio de valores para atributos da base de dados.

4.3 Substância

O aspecto *substância* possui o objetivo de permitir a diferenciação de restrições de integridade de acordo com o tipo de condicionamento, imposto pela restrição, à evolução do banco de dados. Os critérios relevantes a este aspecto, assim como seus sub-critérios serão expostos e definidos de forma detalhada nas seções seguintes.

4.3.1 Abrangência de Estados

Conceitualmente, costuma-se chamar de estado de um banco de dados o conjunto de valores referentes ao mesmo em um determinado momento, ou seja, um estado do banco de dados refere-se a um *"snapshot"* da base de dados. De acordo com essa definição, o critério *abrangência de estados* diz respeito a quais estados da base de dados devem ser conhecidos e utilizados durante o processo de verificação de uma restrição de integridade. De acordo com este critério, com base em (SANTOS, 1980; CASTILHO; CASANOVA; FURTADO, 1982; CASTILHO, 1987; BÖHLEN, 1994; GETZ; LIPECK, 1995), classificou-se as restrições em *estáticas* e *dinâmicas*.

Restrições de integridade *estáticas* restringem quais dados podem fazer parte da base de dados, enquanto restrições *dinâmicas* restringem a evolução desses dados (CASTILHO; CASANOVA; FURTADO, 1982).

As restrições de integridade *estáticas*, também conhecidas com intra-estado, são restrições que abrangem somente um estado da base de dados, ou seja, necessitam do conhecimento e da utilização de somente um estado da base de dados para que possam ser verificadas. Esta classe de restrições de integridade é bastante conhecida e, diversas definições para a mesma podem ser encontradas na literatura (CASTILHO; CASANOVA; FURTADO, 1982; CASTILHO, 1987; MEDEIROS; JOMIER; CELLARY, 1992; BÖHLEN, 1994; CHOMICK; TOMAN, 1995; DOUCET et al., 1997; ESCOFET, 2001;). A seguir, tem-se a definição de Castilho:

> "Considerando o estado da base em um certo instante, restrições de integridade estáticas são aquelas cuja validade pode ser verificada examinando somente um estado da base" (CASTILHO, 1987)

Divergindo de outros trabalhos (CHOMICKI; TOMAN, 1995; GERTZ; LIPECK, 1995; ESCOFET, 2001), esta classificação considera que restrições de integridade *estáticas* também podem ser temporais. Por exemplo, a seguinte restrição em linguagem coloquial: "*Nenhum projeto deve estar inativo assim que se iniciar o dia 1º de janeiro de 1982*." diz respeito a uma base de dados que controla o andamento de uma série de projetos de software e, referencia temporalmente somente um estado da base de dados, ou seja, referencia somente o estado da base de dados no primeiro momento do dia em questão. Portanto, a restrição é classificada como *estática e temporal*.

As restrições *dinâmicas*, ao contrário das *estáticas*, necessitam do conhecimento e da utilização de dois ou mais estados da base de dados para que possam ser verificadas. Este tipo de restrição *é obrigatoriamente temporal*, pois, mesmo que o tempo não seja explicitamente referenciado, o conceito temporal é implicitamente utilizado quando são analisados estados distintos de uma base de dados. É possível ainda dividir o conjunto de restrições *dinâmicas* em dois sub-conjuntos disjuntos, o das restrições de *transição* e o das restrições que *não são de transição*.

Segundo (CASTILHO, 1987), uma restrição de integridade pode ser classificada como restrição de *transição* quando sua validade pode ser verificada examinando dois ou mais estados da base de dados que se seguem no fluxo do tempo. Restrições dessa classe restringem a forma como os dados de uma base de dados podem ser atualizados (CASTILHO; CASANOVA; FURTADO, 1982). De acordo com as definições

anteriores, conclui-se que restrições de integridade de *transição* são restrições que necessitam do conhecimento e verificação de dois e somente dois estados da base de dados consecutivos em relação ao fluxo do tempo. Um exemplo clássico desse tipo de restrição é definido em linguagem coloquial como: *"O valor do salário de um funcionário não pode decrescer."*.

Além das restrições de *transição* existem ainda as restrições que *não são de transição*. Restrições *dinâmicas não de transição* são restrições de integridade que referenciam e utilizam dois ou mais estados não consecutivos da base de dados para que possam ser verificadas ou restrições que referenciam e utilizam mais do que dois estados em sua verificação. A restrição em linguagem coloquial *"Nenhum projeto pode estar inativo após o dia 1º de janeiro de 2002."* é *dinâmica não de transição*, pois todos os estados da base de dados posteriores ao dia 1º de janeiro de 2002 devem ser conhecidos e analisados, e poderão ser necessários mais do que dois estados da base de dados em sua verificação.

A Figura 4.2 ilustra a classificação em relação ao critério *abrangência de estados*.

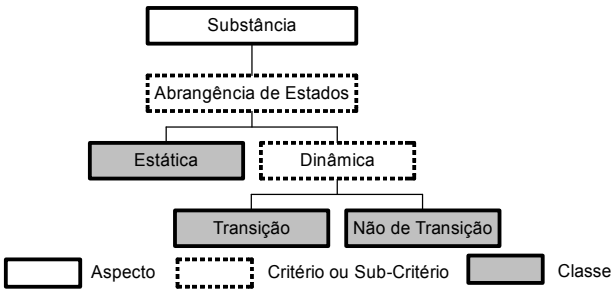


Figura 4.2: Critério *Abrangência de Estados*

4.3.2 Tipo do Restringente

A classificação em relação ao critério de *tipo do restringente* diferencia as restrições de integridade com base nos tipos dos integrantes de seus conjuntos restringentes. Caso todos os participantes do restringente de uma restrição forem de mesmo tipo, a mesma será classificada como possuidora de um restringente de tipo *homogêneo*. Caso o restringente possua integrantes de tipos distintos, a restrição possuirá um restringente de tipo *heterogêneo*.

Os tipos possíveis de integrantes do conjunto restringente são *esquema*, *entidade*, *relacionamento* e *domínio*. Um integrante do tipo *domínio* pode ser um *valor*, quando o *domínio* definido corresponde a um valor único, uma *faixa de valores*, quando o domínio é definido através de características as quais os valores devam possuir, ou uma *enumeração de valores*, quando o *domínio* é definido através da enumeração explícita de valores. Como exemplo dessa classificação, a restrição em linguagem coloquial *"O campo sexo deverá conter o valor Feminino ou Masculino."*, definida sobre uma base com dados sobre pessoas, pode ser classificada como possuidora de um restringente de tipo *enumeração de valores*, enquanto a restrição em linguagem coloquial *"O campo salário deve possuir valores positivos de pertencentes aos números reais."*, definida sobre uma base com dados de funcionários, pode ser classificada como possuidora de um restringente de tipo *faixa de valores*.

Quando um integrante do conjunto restringente de uma restrição é do tipo *domínio*, existem algumas restrições em relação à sua classificação sob outros critérios. Dentre elas está o fato de que a *formação*, a *abrangência* e a *abrangência de ligação* de seu restringente devem obrigatoriamente ser *indefinidas*. Além disso, caso a restrição seja

do tipo *temporal*, a *abrangência temporal* do restringente será *indefinida* e, caso a restrição seja do tipo *versionada*, a *abrangência de versionamento* de seu restringente também deverá ser *indefinida*.

A Figura 4.3 ilustra graficamente a classificação de acordo com o critério *tipo do restringente*.

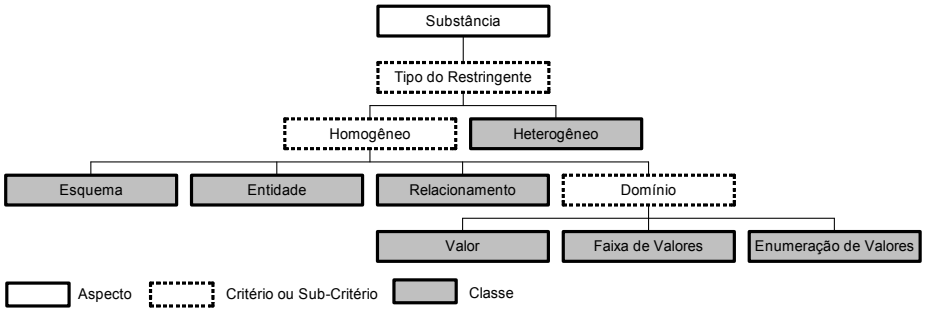


Figura 4.3: Critério *Tipo do Restringente*

4.3.3 Tipo do Restringido

De forma praticamente análoga ao critério anterior, este critério diferencia os tipos dos integrantes do conjunto restringido de uma restrição de integridade. Entretanto, existem algumas diferenças na classificação em relação ao *tipo do restringido* (Figura 4.4).

A primeira delas é que conforme foi dito anteriormente, neste trabalho, considera-se somente restrições de integridade cujo conjunto restringido possua integrantes de mesmo tipo e, portanto, não existe a possibilidade de restringidos de tipo *heterogêneo*. É possível que uma única restrição afete de forma *indireta* objetos de tipos diferentes, mas, neste trabalho são considerados apenas os efeitos diretos das restrições impostas aos dados.

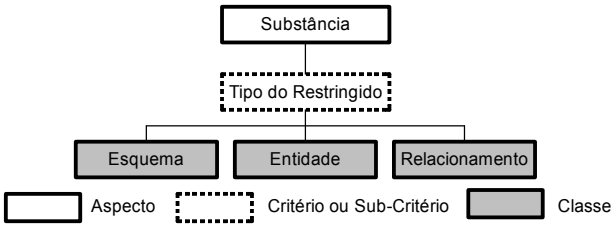


Figura 4.4: Critério *Tipo do Restringido*

A segunda característica que diferencia este critério é a não existência de *domínio* (valor) como tipo do restringido de uma restrição de integridade. Assim, restrições como *"O maior salário deve ser menor ou igual a dez vezes o menor salário."* ou *"Nenhuma quantidade deve ser maior do que mil."* o restringido será um atributo integrante de uma entidade e, portanto, restrições como estas possuem um conjunto restringido com integrantes de tipo *entidade*.

Nesta classificação, é importante notar que, em relação ao critério *tipo do restringido*, não estão sendo consideradas restrições que atingem atributos de forma isolada. Essas características são expostas através dos critérios de abrangência presentes neste mesmo aspecto.

4.3.4 Formação do Restringente

É importante considerar que em uma base de dados é comum a existência de objetos primitivos do modelo de dados adotado e de objetos construídos com base objetos

primitivos e em construtores do modelo. Como exemplo, o construtor de especialização possui a capacidade de criar novos objetos, tomando como base objetos menos especializados.

Integrantes de tipo domínio do conjunto restringente não são considerados parte da base de dados e, com isso, não podem ser ditos primitivos ou construídos. Assim, considera-se que suas formações sejam indefinida.

Com base nestes fatos, criou-se, para esta classificação, o critério *formação do restringente* que diz respeito à formação dos integrantes do conjunto restringente de uma restrição de integridade. Esta formação pode ser considerada *homogênea*, quando todos os seus integrantes forem de mesmo tipo, ou *heterogênea*, quando o conjunto restringente possuir integrantes de dois ou mais tipos distintos.

Restrições *homogêneas* são sub-classificadas em: (i) formação *indefinida*, quando os integrantes do restringente são de tipo *domínio*, (ii) formação *primitiva*, quando todos os integrantes do restringente são objetos primitivos do modelo de dados adotado e, (iii) formação *construída*, quando todos os integrantes do restringente são objetos construídos com a utilização de construtores do modelo de dados adotado. A Figura 4.5 ilustra graficamente este critério da classificação.

Considera-se ainda que este critério da classificação depende fortemente do esquema conceitual adotado, pois dependendo da forma de como o mesmo foi projetado, objetos similares podem ser representados como objetos primitivos ou construídos.

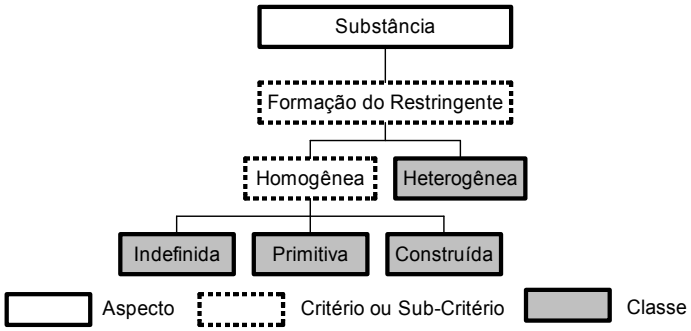


Figura 4.5: Critério *Formação do Restringente*

4.3.5 Formação do Restringido

De forma análoga ao critério *formação do restringente* (Seção 4.3.4), este critério classifica restrições de acordo com o tipo dos integrantes de seus conjuntos restringidos. A principal diferença é que não se classifica a homogeneidade do restringido. Além disso, considerando que todos os integrantes do conjunto restringido devem fazer parte da base de dados em questão, não se considera existência de restringido com formação *indefinida*.

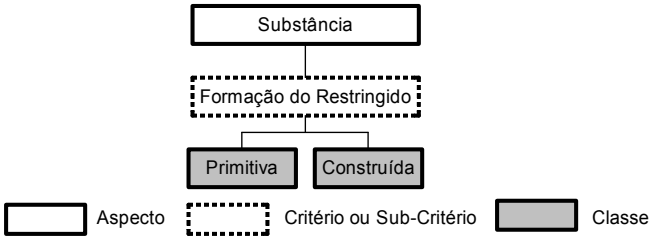


Figura 4.6: Critério *Formação do Restringido*

Considerando as observações anteriores, em relação à formação de seus restringidos, classificam-se restrições de integridade em bases de dados temporais de versões como *primitivas* ou *construídas*, assim como pode ser visto na Figura 4.6.

4.3.6 Formação da Condição Restritiva

Um dos principais componentes de uma restrição de integridade consiste em sua condição restritiva, também conhecida como asserção. É através dela que é estabelecida a relação entre os componentes do restringido e do restringente de forma que se constitua uma condição a ser respeitada pelos dados da base de dados em questão.

Uma condição restritiva é composta por uma ou mais comparações simples, constituídas de dois elementos relacionados através de um operador relacional. Comparações simples de uma mesma restrição devem, obrigatoriamente, estar relacionadas através de operadores lógicos. Com base nisso, restrições de integridade com condições restritivas formadas por uma única comparação simples são classificadas como *simples*, enquanto restrições de integridade possuidoras de condições restritivas com duas ou mais comparações simples são classificadas como *compostas*.

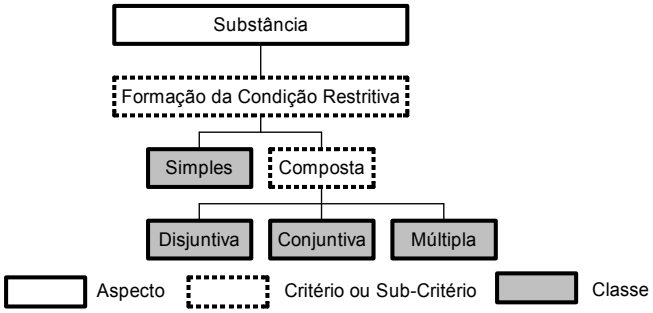


Figura 4.7: Critério *Formação da Condição Restritiva*

Dentre a classe de restrições *compostas*, existe ainda uma sub-classificação. Quando uma restrição de integridade possuir condição restritiva formada por duas ou mais comparações simples, relacionadas através do operador lógico de disjunção *ou*, a mesma será dita *disjuntiva*. Quando todas as condições simples estiverem relacionadas através do operador lógico de conjunção *e*, a restrição será dita *conjuntiva*. Finalmente, quando ocorrerem ambos operadores lógicos *e* e *ou* relacionando as condições simples de uma mesma condição restritiva, a restrição será dita *múltipla*.

A Figura 4.7, ilustra a classificação quanto ao critério *formação da condição restritiva*.

4.3.7 Aspecto Restringente

A condição a ser respeitada pelos dados de uma base de dados, devido à existência de uma restrição de integridade, pode ter sua validade dependente de diferentes aspectos relacionados aos integrantes do conjunto restringente da restrição. Dentre esses aspectos está a *cardinalidade*, o *comportamento*, a *estrutura*, a *composição interna*, características referentes à *temporalidade* e, características de *versionamento* de um conjunto de objetos com determinadas características comuns.

O aspecto *cardinalidade* pode ser utilizado para restringir os dados em relação ao número de objetos, que possuam determinadas características comuns. Dessa forma, este aspecto pode ser especializado em *presença*, quando referido à existência de um único objeto, *ausência*, quando refere-se a não existência de determinado objeto e, *presença múltipla*, quando refere-se à existência de um determinado número (maior do que um) de objetos da base de dados. É possível ainda que a *cardinalidade* esteja

relacionada à participação de um objeto como componente de um outro e não somente à existência pura do objeto na base de dados (SANTOS, 1980).

Restrições de integridade que possuam integrantes de seus restringentes de tipo *esquema* podem ainda restringir os dados da base de dados em relação ao *comportamento* de objetos ou à *estrutura* do próprio esquema. Segundo (MEDEIROS, 1992), em um mundo orientado a objetos, este primeiro aspecto corresponde a estabelecer diretivas de consistência para métodos. Como exemplo, é possível citar restrições, com abrangência de estados *estática*, de domínio sobre parâmetros de métodos e, restrições, com abrangência de estados *dinâmica*, de especificação de seqüências de execução de métodos.

Outra possibilidade é que restrições restrinjam os dados com base na *composição interna* de objetos da própria base, sendo que o termo *composição interna* refere-se, por exemplo, a valores de atributos, propriedades de componentes de objetos construídos, entre outros.

É possível ainda que restrições de integridade restrinjam a base de dados com base em aspectos da *temporalidade* ou do *versionamento* de integrantes do conjunto restringente. Nesses casos, é de fácil percepção o fato que essas restrições deverão ser classificadas como *temporais* ou *versionadas*, de acordo com o aspecto utilizado. Como exemplo de restrições cujos restringentes se baseiam em características temporais, está a restrição em linguagem coloquial *"Não é permitido parar e reiniciar o andamento de um projeto."* que se baseia em características temporais dos componentes da base de dados que representam projetos. Em relação às restrições que se baseiam em aspectos de versionamento, é possível citar, as seguintes restrições *"Uma versão consolidada de um objeto não pode sofrer alterações."* e *"Para ocorrer uma derivação de uma versão de objeto, a mesma deve ser estável ou consolidada."*.

Em restrições classificadas como *temporais*, os aspectos citados anteriormente podem estar condicionados a um período. Dessa forma, a seguinte restrição em linguagem coloquial *"O pagamento do boleto deve ocorrer antes do envio da mercadoria."* refere-se à *presença* dos componentes da base de dados que representam o pagamento do boleto em um período anterior ao atual. Em relação ao versionamento de restrições, esses aspectos também podem estar relacionados a determinadas versões de objetos.

Com base em todos os aspectos descritos, classificam-se restrições de integridade em relação ao critério *aspecto restringente* de acordo com a Figura 4.8.

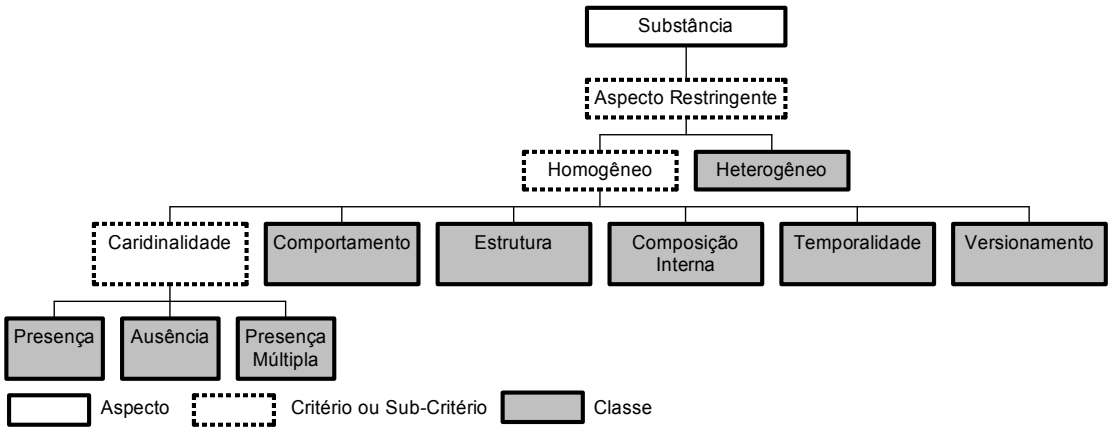


Figura 4.8: Critério *Aspecto Restringente*

4.3.8 Aspecto Restringido

A classificação de restrições de integridade em relação ao critério *aspecto restringido* é praticamente análoga ao critério anterior. A única diferença consiste na não classificação da homogeneidade do *aspecto restringido*. A classificação de restrições, quanto ao critério *aspecto restringido*, é apresentada através da Figura 4.9.

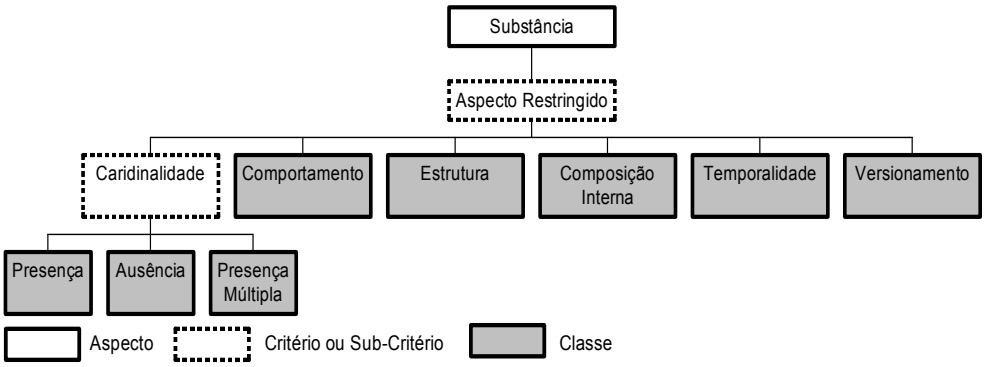


Figura 4.9: Critério *Aspecto Restringido*

4.3.9 Propósito

O principal intuito deste critério é possibilitar a diferenciação das restrições de acordo com a razão a que se destinam. Classificam-se ortogonalmente restrições de integridade de propósito *condicional*, *causativo* e *verificativo*, conforme ilustrado na Figura 4.10. Nota-se ainda uma relação íntima entre esse critério e o de *tratamento* (Seção 4.5.6), do aspecto *aplicação*, pois com base em seu *propósito*, podem ser definidas ações de violação e ações de prevenção de violação da restrição

Restrições de integridade de propósito *condicional*, também conhecidas como *hard constraints* (DAYAL et al., 1988; GOONETILLAKE; CARNDUFF; GRAY, 2002), *não violáveis* (BUCHMANN; CARRERA; VAZQUEZ-GALINDO, 1986) ou *requeridas* (FREEMAN-BENSON; BORNING, 1992), não permitem que alterações na base de dados violem sua condição restritiva, de forma que, quando violadas, ocorre o impedimento da efetivação da transação que causou a violação. A transação em questão é desfeita e não influencia de maneira nenhuma o estado válido da base de dados. Obviamente, este tipo de restrição de integridade deve possuir *pontos de verificação baseados em operação* e *tratamento* de *reconstituição*.

Restrições de propósito *causativo* estabelecem ações a serem executadas sobre a base de dados após sua violação, conseqüentemente, este tipo de restrição deve possuir *tratamento* de *complementação*. Este tipo de restrição é comumente representada através de regras de condição e ação e, quando restrições possuem *pontos de verificação baseados em operação*, costumam ser especificadas em um SGBD comercial através de gatilhos (*triggers*) ligados a eventos de alteração dos dados da base em questão. Em particular, restrições desta classe devem ser modeladas em duas partes, a condição, que pode ser especificada da mesma forma que outras restrições e, a ação, que costuma ser especificada através de uma seqüência de comandos de atualização de dados presentes na linguagem de manipulação de dados (DML) do modelo utilizado.

A última classe de restrições é a classe de propósito *verificativo*. Este tipo de restrição, também conhecido como *soft constraint* (DAYAL et al., 1988; GOONETILLAKE; CARNDUFF; GRAY, 2002), *violável* (BUCHMANN; CARRERA; VAZQUEZ-GALINDO, 1986) ou *preferencial* (FREEMAN-BENSON; BORNING, 1992) especifica condições que deveriam ser respeitadas, mas que não influenciam

diretamente a execução de operações sobre a base de dados e que, caso violadas, não acarretam na execução automática de nenhuma ação de reparo. Normalmente, este tipo de restrição é utilizado para facilitar a administração da base de dados, por parte do DBA, de forma que o mesmo esteja ciente das restrições dessa classe que não estão sendo respeitadas pela base de dados atual e, caso necessário, tome providências para alterar essa situação. Conseqüentemente, este tipo de restrição de integridade deve possuir *tratamento* de *detecção*.

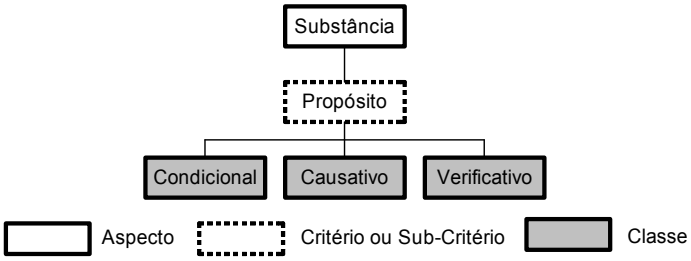


Figura 4.10: Critério *Propósito*

4.3.10 Abrangência do Restringente

Um dos principais critérios a serem analisados durante a otimização do processo de verificação de restrições de integridade diz respeito ao escopo das restrições, ou seja, à área da base de dados a ser analisada para que a verificação da restrição possa ser realizada. Dessa forma, é possível dizer que a abrangência de uma restrição de integridade está intimamente ligada à sua complexidade de verificação e à quantidade de objetos a serem examinados durante a verificação de sua validade.

Baseado nos fatos descritos criou-se o critério *abrangência do restringente*, como parte integrante do aspecto *substância*, afim de classificar restrições de integridade em bases de dados temporais de versões em relação à abrangência de seus conjuntos restringentes.

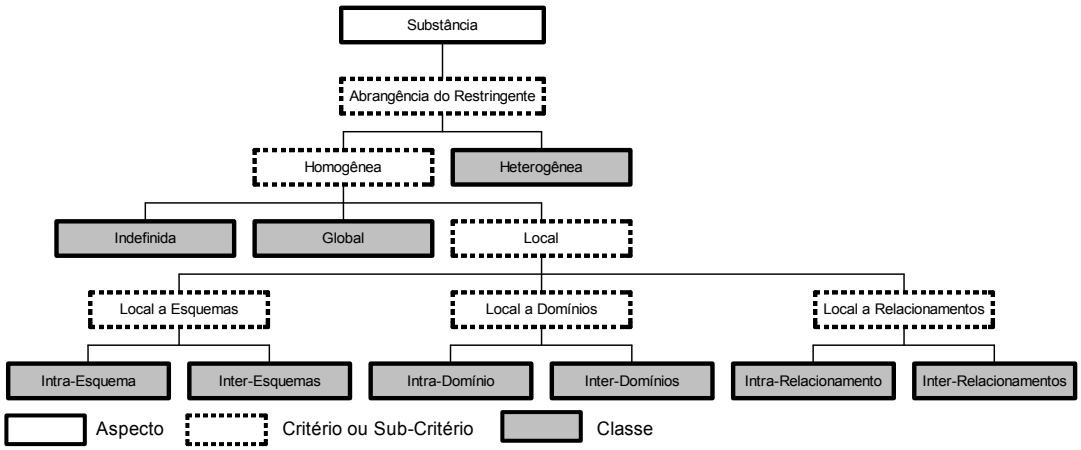


Figura 4.11: Critério *Abrangência do Restringente*

A *abrangência do restringente* (Figura 4.11) pode ser: (i) *indefinida*, quando o mesmo for de tipo *domínio*, (ii) *global*, quando englobar a base de dados como um todo ou, (iii) *local*, quando englobar apenas partes específicas da base possuidoras de características comuns. Dentre as restrições com restringentes de abrangência *local*, existe ainda uma sub-classificação que as divide em restrições com abrangência *local a esquemas*, que só se faz válida quando é aplicado à base de dados o suporte adequado à

evolução de esquemas, *local a domínios*, e *local a relacionamentos*. Nota-se ainda que cada uma dessas classes é sub-classificada em *intra-classe*, quando o restringente envolver apenas um esquema, domínio ou relacionamento, e *inter-classes*, caso contrário.

Este critério é complementado pelo critério *abrangência de ligação do restringente* (Seção 4.3.12) e, em restrições de integridade com características *temporais*, o mesmo também é complementado pelo critério *abrangência temporal do restringente* (Seção 4.6.4) pertencente ao aspecto *temporalidade*, de forma que, estes critérios devem estar condizentes em relação à classificação de uma mesma restrição de integridade. Este fato ocorre de forma análoga em restrições com características de *versionamento*, em relação ao critério *abrangência de versionamento do restringente* (Seção 4.7.3) presente no aspecto *versionamento*.

4.3.11 Abrangência do Restringido

Este critério é praticamente análogo ao anterior. Considera-se que todas as observações feitas, em relação ao conjunto restringente de uma restrição, também se apliquem a seu conjunto restringido, inclusive a possibilidade de complementação através do critério de *abrangência de ligação do restringido* (Seção 4.3.13) e de critérios pertencentes aos aspectos *temporalidade* e *versionamento*. Somente duas alterações, em relação à *abrangência do restringente*, se aplicam a este critério. A primeira delas diz respeito à não classificação quanto à homogeneidade da *abrangência de restringidos* e, a segunda alteração deve-se à não existência de restringidos de *tipo domínio*, resultando na extinção da classe de restrições com restringidos de abrangência *indefinida*.

Como resultado, em relação ao critério abrangência do restringido, chegou-se à classificação de restrições de integridade, em bases de dados temporais de versões, exposta através da Figura 4.12.

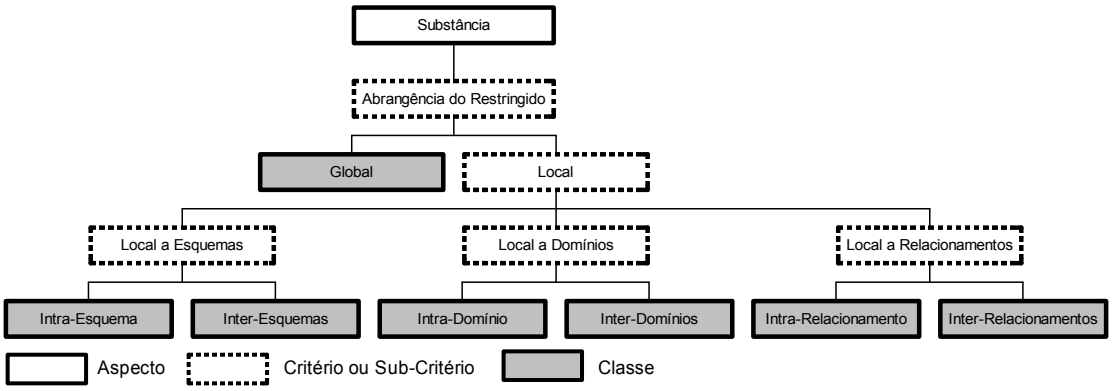


Figura 4.12: Critério *Abrangência do Restringido*

4.3.12 Abrangência de Ligação do Restringente

A abrangência entre as ligações de componentes de restringidos e restringentes, necessárias para a verificação de restrições, também deve ser analisada. Com essas ligações em mente, o critério *abrangência de ligação do restringente* classifica restrições de forma análoga à *abrangência do restringente*, exceto que a abrangência de ligação *local* pode ser local a *objetos*, *atributos*, *relacionamentos*, *métodos*, *temporalidades* ou *versionamentos*.

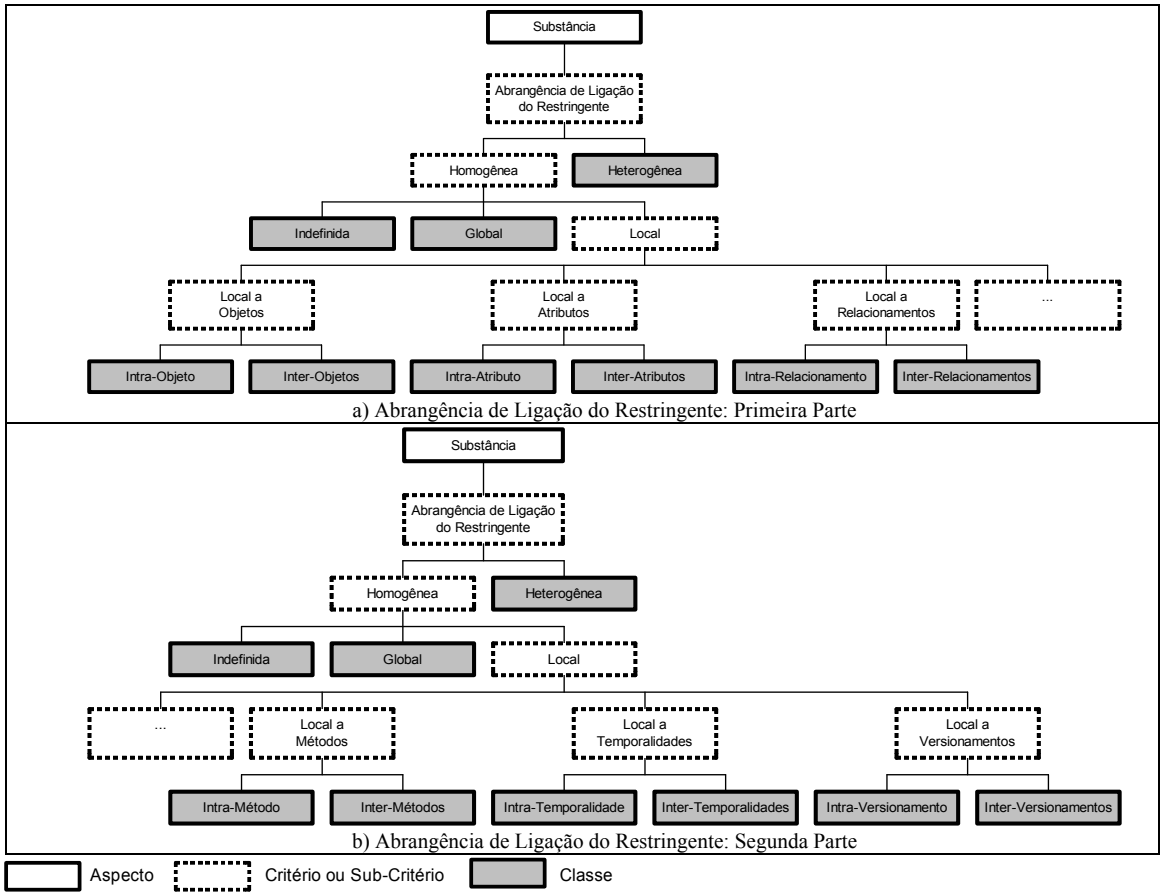


Figura 4.13: Critério *Abrangência de Ligação do Restringente*

A Figura 4.13, dividida em duas partes para permitir sua visualização, ilustra a classificação por este critério.

4.3.13 Abrangência de Ligação do Restringido

A *abrangência de ligação do restringido* é classificada de acordo com o critério anterior, exceto pela impossibilidade de abrangência *indefinida* ou *heterogênea*. A Figura 4.14, dividida em duas partes para permitir sua visualização, ilustra esta classificação.

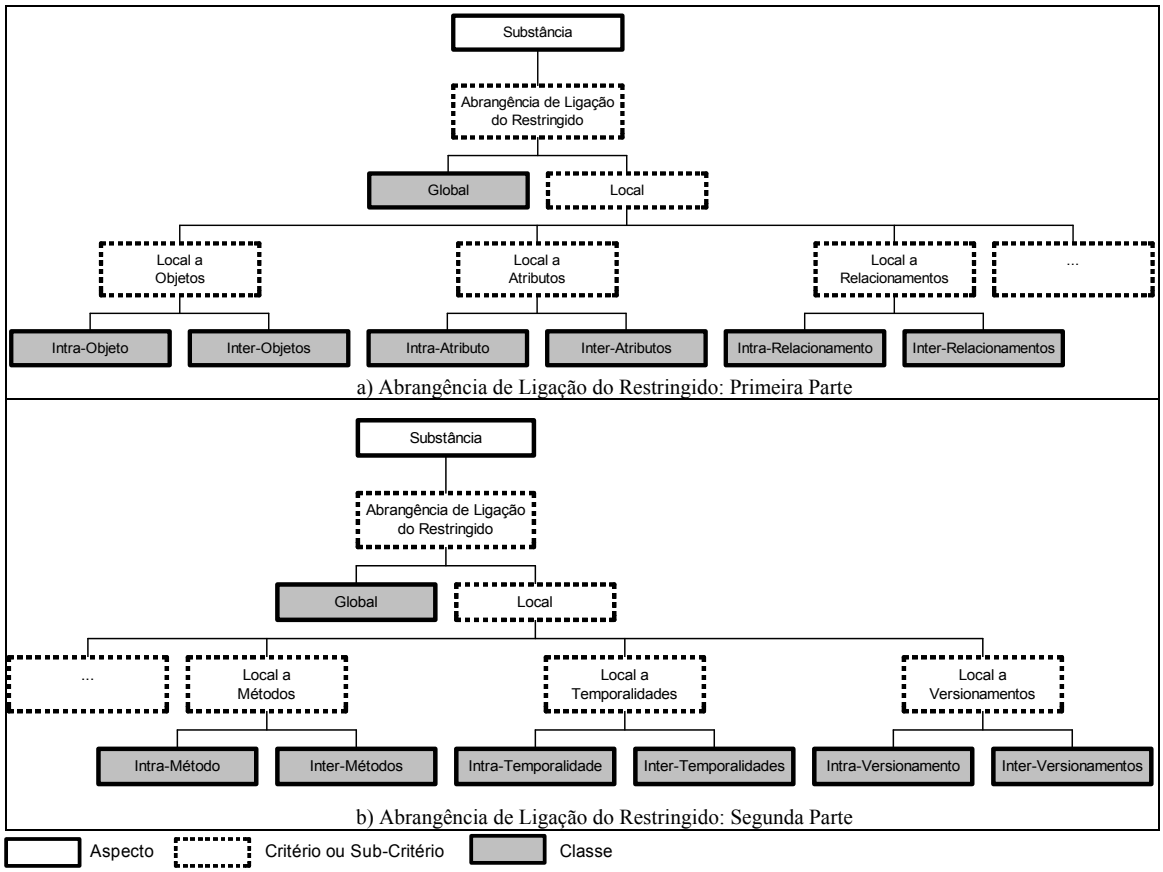


Figura 4.14: Critério *Abrangência de Ligação do Restringido*

4.3.14 Completude

Conforme dito anteriormente, o principal intuito de uma base de dados é representar a parte do mundo real que necessita ser automatizada. Para que a representação da realidade modelada seja fiel, diversas restrições de integridade inerentes à realidade devem ser respeitadas pela base. Através do aspecto *origem*, percebe-se que restrições de integridade podem ser provenientes de diversos agentes distintos, mas todas devem ser especificadas e respeitadas de alguma forma, para que se represente fielmente a realidade.

Em muitos casos, a realidade determina restrições complexas e, que pensando nelas como um todo, se torna difícil a especificação e verificação na base de dados. Nesses casos, é comum a representação de uma única restrição complexa do mundo real através de diversas restrições mais simples, que devem interagir a fim de representar na base de dados a restrição da realidade como um todo. A representação de uma restrição real através de duas ou mais restrições no esquema facilita muito sua especificação e verificação. No entanto, elas devem ser ativadas e desativadas em conjunto e, possivelmente, possuir uma ordem pré-definida de verificação. Nesta classificação, estas relações são analisadas através do critério *completude*, inerente ao aspecto *substância*, e dos critérios *ordem de precedência* (Seção 4.5.2) e *tipo de ativação e desativação* (Seção 4.5.4), inerentes ao aspecto *aplicação*. De acordo com isso, define-se que restrições simples, relacionadas entre si, devem obrigatoriamente ser ativadas e desativadas em conjunto e possivelmente, possuir uma ordem pré-definida de verificação.

Além disso, como restrições complexas podem ser especificadas através de um conjunto de restrições simples relacionadas entre si, é possível que restrições simples

necessitem da criação de novas restrições simples que, sucessivamente, podem necessitar da criação de outras restrições simples.

De acordo com esses aspectos, classifica-se uma restrição de integridade em: (i) *completa*, quando a restrição especificada na base de dados representa inteiramente a restrição em questão da realidade e, (ii) *incompleta*, quando a restrição da base de dados representa somente parte da restrição do mundo real e, com isso, necessita da existência de outras para representá-la completamente.

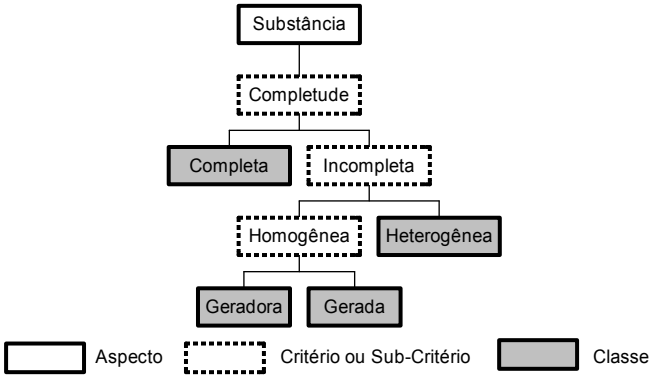


Figura 4.15: Critério *Completude*

Com base em (RAM et al., 1997), restrições de integridade *incompletas* podem ser *homogêneas geradoras*, quando somente necessitarem da geração de novas restrições *incompletas*, ou *homogêneas geradas*, quando forem geradas a partir de outras restrições, mas não gerarem nenhuma nova restrição. Percebe-se ainda, a existência de uma certa hierarquia entre restrições *incompletas* possibilitando que uma mesma restrição possa ter sido gerada a partir da exigência de outra e ao mesmo tempo, exigir a geração de novas restrições. Este tipo de restrição é classificada como *heterogênea*.

Restrições *completas* são comuns e diversos exemplos já foram citados ao longo do capítulo. Entretanto, exemplos de restrições *incompletas* são mais raros. Considere a base de dados de uma aplicação que controla pousos e decolagens em um aeroporto. Nessa base, a seguinte restrição poderia existir: *"É desejável que todos os procedimentos de segurança sejam efetuados em qualquer pouso ou decolagem."*. Nota-se claramente que a manutenção desta restrição na base de dados envolve vários passos distintos. Além disso, a forma com que foi executado um procedimento ou a não execução do mesmo pode influenciar os procedimentos posteriores. Nesse caso, é quase imprescindível que a restrição seja dividida em restrições mais simples relacionadas a cada um desses procedimentos. Sendo assim, cada uma dessas novas restrições seria classificada como *incompleta* e *gerada*.

A classificação de restrições de integridade, quanto ao critério *completude*, é definida de acordo com a Figura 4.15.

4.4 Especificação

O termo especificação de uma restrição de integridade diz respeito à sua incorporação à especificação de uma base de dados (SANTOS, 1980). Após a incorporação, os dados da base em questão, referentes à restrição, deverão respeitá-la de forma incondicional.

Esta seção apresenta mais um aspecto relevante à classificação de restrições de integridade em bancos de dados temporais de versões, o aspecto *especificação*. Através deste aspecto, classificam-se restrições de integridade de acordo com as formas com que

as mesmas podem ser representadas (especificadas) sobre uma base de dados. Além da declaração da restrição propriamente dita, consideram-se parte deste aspecto características temporais e de versionamento inerentes à restrição em si e não aos seus conjuntos restringidos e restringentes. A seguir, serão apresentados em detalhes todos os critérios da classificação de restrições de integridade inerentes ao aspecto *especificação*.

4.4.1 Declaração

O critério *declaração* diz respeito à forma com que restrições são definidas sobre uma base de dados temporal de versões. Este critério está relacionado não somente ao conceito das restrições de integridade, mas principalmente, está relacionado a restrições de integridade já modeladas e especificadas de alguma maneira em uma base de dados. Essa afirmação é importante, pois, em muitos casos, uma mesma restrição de integridade conceitual pode ser declarada de formas diferentes.

Classificam-se restrições de integridade com *declaração implícita*, as restrições cuja declaração seja inerente ao modelo de dados definido para base de dados, ou seja, restrições desta classe possuem *declaração implícita* em relação ao modelo de dados definido para a aplicação em questão. Como exemplo estão as restrições de não duplicidade de tuplas e a manutenção da relação entre atributos de chave primária e de chave estrangeira, no modelo relacional, a cardinalidade de relacionamentos e agregações, definidas diretamente no modelo Entidade Relacionamento Estendido (EER), entre outras (PLEXOUSAKIS, 1993; SILVA, 1997).

Em relação à sua *declaração*, restrições de integridade *explícitas* são restrições cuja representação corresponde a declarações explícitas na especificação do banco de dados em questão, que não estejam representadas através do modelo de dados da aplicação. De forma ideal, restrições dessa classe deveriam corresponder somente a peculiaridades da realidade modelada, mas é comum a utilização de restrições com *declaração explícita* para suprir deficiências do modelo de dados ou da modelagem utilizada (SANTOS, 1980; PLEXOUSAKIS, 1993).

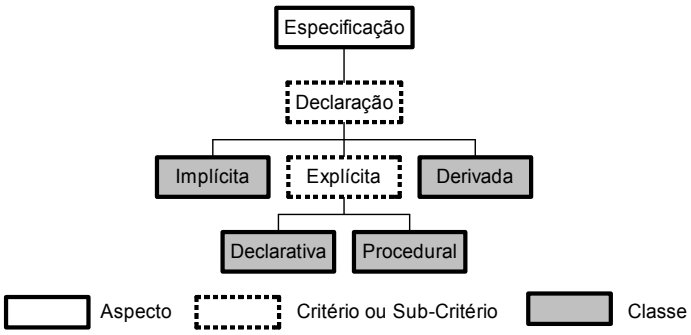


Figura 4.16: Critério *Declaração*

Esta classe de restrições de integridade pode ainda ser sub-divida de forma a separar restrições de integridade definidas de forma *declarativa* e restrições definidas de forma *procedural*. Restrições *declarativas* especificam somente o conceito da restrição a ser mantida (o que fazer) e não mencionam a forma como mantê-la. Como exemplo de restrições dessa classe estão as que utilizam linguagens de consulta baseadas em SQL de forma a especificar estados inconsistentes da base de dados. Em contrapartida, restrições definidas de forma *procedural* não especificam claramente seu conceito, mas especificam todos os passos a serem seguidos afim de mantê-la (como fazer).

Restrições com *declaração derivada* são restrições não explicitamente declaradas que surgem como conseqüência da existência conjunta de outras restrições de integridade, *explicitas* ou *implícitas*.

É importante frisar que restrições de integridade com *origem* no *modelo de dados* ou na *implementação* são sempre *declaradas* de forma *implícita*, enquanto restrições de integridade com *origem* no *ambiente* ou na *empresa*, de acordo com seu conteúdo, podem ser declaradas de forma *implícita*, *explícita* ou *derivada*.

A Figura 4.16 ilustra graficamente a classificação de restrições de integridade de acordo com o critério *declaração*.

4.4.2 Temporalidade da Restrição

Através do critério *temporalidade da restrição* (Figura 4.17), classificam-se restrições de integridade em bancos de dados temporais de versões quanto à temporalidade da própria restrição e não em relação à temporalidade de seus conjuntos restringente e restringido. Caso a restrição de integridade seja *temporal*, estão relacionados a ela os conceitos de tempo de validade, tempo de transação ou ambos tipos de tempo. Assim, é possível a manutenção de um histórico de modificações efetuadas sobre a restrição e o controle do período em que a mesma deverá ser verificada (validade).

Uma restrição de integridade é dita *não temporal* caso não tenha, envolvido a ela, o conceito de tempo e, conseqüentemente, não sejam armazenadas as modificações sobre a mesma, nem seja considerada sua validade em relação ao tempo. Restrições de integridade *temporais* devem englobar o histórico de todas as suas alterações e podem possuir *tempo de validade*, *tempo de transação*, ambos tipos de tempo (*bitemporal*), ou *tempo definido pelo usuário*. Considera-se ainda que, restrições de integridade *temporais* podem estar relacionadas ao *tempo com determinação*, se referindo a momentos com exatidão, ou ao *tempo com indeterminação*, se referindo a momentos com pouca precisão (JENSEN et al., 1998).

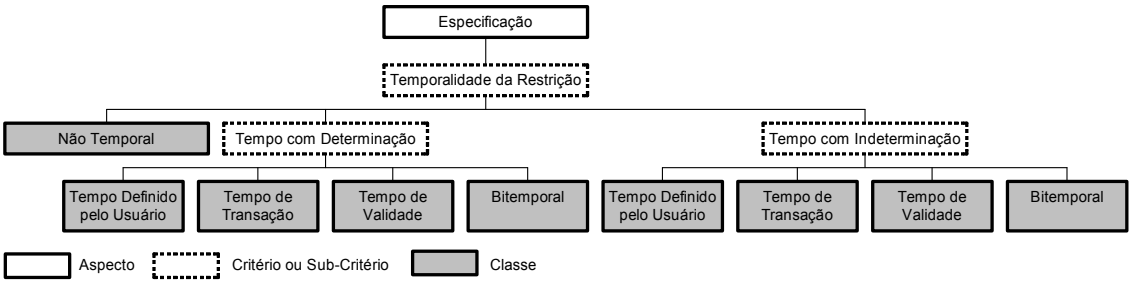


Figura 4.17: Critério *Temporalidade da Restrição*

Como exemplo de restrições normalmente *não temporais*, têm-se as restrições de domínio e integridade referencial, utilizadas no modelo relacional, pois as mesmas não sofrem alterações com o tempo. Considere agora a realidade de uma aplicação que controle as matrículas e atribuições de aulas de uma universidade. Nessa realidade, a seguinte restrição em linguagem coloquial é um exemplo de restrição que deve ser baseada no *tempo com determinação* e *bitemporal*: *"Não devem existir turmas com menos do que vinte alunos na universidade."*. É interessante que essa restrição esteja relacionada à temporalidade, pois ela pode ser revogada a qualquer momento ou o número mínimo de alunos pode variar de acordo com a realidade da universidade.

A fim de evitar dúvidas em relação a essa classificação, é importante citar que o critério *temporalidade da restrição* é completamente ortogonal ao aspecto *temporalidade* desta mesma classificação, ou seja, uma mesma restrição pode ser classificada por qualquer uma das combinações possíveis de ambos componentes. Isto ocorre, pois o primeiro componente é inerente à restrição de integridade em si e o segundo diz respeito à temporalidade de seus conjuntos restringente e restringido.

4.4.3 Abrangência de Temporalidade da Restrição

Este critério visa classificar restrições de integridade de acordo com a abrangência da temporalidade envolvida na restrição. Assim como foi frisado no critério anterior, neste critério, é considerada apenas a temporalidade da restrição em si, e não a temporalidade dos conjuntos restringido e restringente, a ela relacionados.

A *abrangência de temporalidade da restrição* pode ser *global*, quando a mesma estiver relacionada a todo o tempo de vida da base de dados, ou *local*, quando estiver relacionada somente a uma parte do tempo de vida da base de dados. A abrangência *local* pode ainda envolver somente momentos no: (i) *passado*, através de um *ponto no tempo,* de um *intervalo temporal*, ou de um *elemento temporal*, (ii) *presente*, através de um *ponto no tempo*, (iii) *futuro*, através de um *ponto no tempo,* de um *intervalo temporal*, ou de um *elemento temporal* ou, (iv) momentos em dois ou mais desses períodos (*localmente heterogênea*), através de um *intervalo* ou de um *elemento temporal*. Através da Figura 4.18 é exposta graficamente a classificação de restrições quanto a este critério.

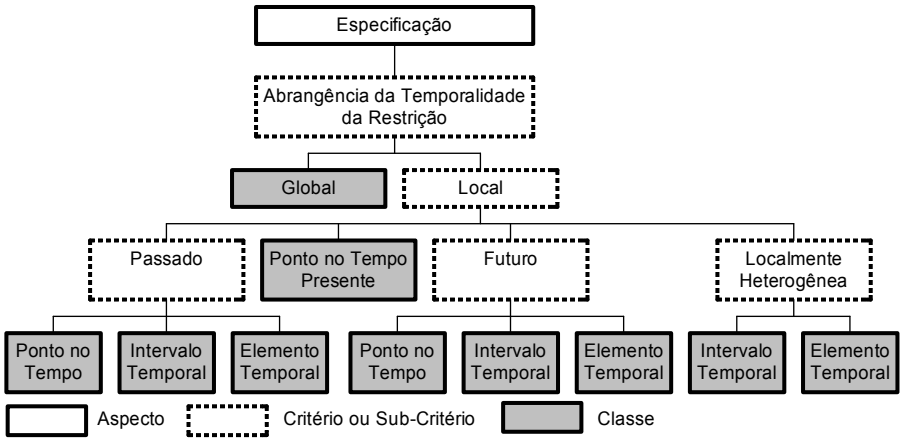


Figura 4.18: Critério *Abrangência da Temporalidade da Restrição*

De acordo com essa classificação, fica claro que restrições de integridade classificadas como *não temporais* no critério anterior deverão ser classificadas como *globais* neste critério, pois, as mesmas serão válidas durante todo o tempo de vida da base de dados.

Quanto à abrangência de temporalidade, os exemplos de restrições de domínio e integridade referencial citados no critério anterior são classificados como *globais*, pois essas restrições devem sempre ser válidas durante todo o tempo de vida da base de dados. Além disso, a abrangência temporal da restrição sobre o número mínimo de alunos será *local* e, sua sub-classificação depende do período em que esta restrição deva ser respeitada.

4.4.4 Versionamento da Restrição

De forma quase análoga aos dois critérios anteriores, este critério diz respeito ao versionamento relacionado à restrição em si, e não ao versionamento dos conjuntos restringente e restringido, por ela referidos. Dessa forma, em relação a este critério, restrições de integridade são classificadas *não versionadas* quando não existe nenhum controle de versionamento sobre a mesma ou *versionadas* caso contrário. A classificação de restrições pode ser vista na Figura 4.19.

A fim de exemplificar o versionamento de restrições, considere a realidade de um banco de dados de uma ferramenta do tipo CAD (*Computer-Aided Design*), onde são controladas diversas versões de objetos durante o processo de *design*. Nesse contexto, uma restrição sobre a dimensão dos objetos desenvolvidos pode ser versionada, a fim de restringir de formas distintas diferentes versões de objetos.

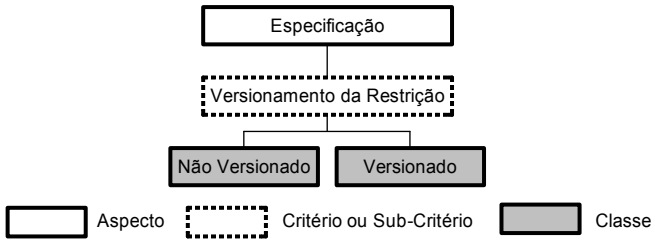


Figura 4.19: Critério *Versionamento da Restrição*

4.5 Aplicação

O aspecto *aplicação* diz respeito à forma com que a integridade dos dados é mantida, com base nas restrições existentes. Novamente, neste aspecto uma restrição de integridade não é classificada com base somente em seu conceito, pois, além do conceito, é necessário o conhecimento da forma com que a restrição foi modelada e especificada, além do ambiente em que a mesma está envolvida para ser possível sua classificação. De um modo geral, pode-se dizer que cada um dos critérios deste aspecto relaciona classes que representam alternativas para a manutenção da integridade de uma base de dados temporal de versões de acordo com as características de cada uma das restrições de integridade relacionadas a ela.

As seções seguintes descrevem os critérios ligados a este aspecto da classificação.

4.5.1 Dependência sobre Regras Dedutivas

É extremamente desejável que bancos de dados temporais de versões sejam gerenciados por um SGBD que suporte os conceitos de tempo e de versão. Além dessa necessidade e das necessidades básicas de qualquer SGBD, não existe outra característica que possa ser considerada obrigatória em relação ao processo de gerência dessas bases de dados. Bases de dados temporais de versões podem ser gerenciadas por um SGBD relacional, orientado a objetos, objeto-relacional, entre outros. Além disso, nada impede que o mesmo também possua características de SGBD ativo ou dedutivo.

Em um SGBD dedutivo, cada base de dados possui um conjunto de regras de dedução utilizadas para deduzir fatos com base nos dados existentes no próprio banco de dados. Nesses casos, é comum que o SGBD verifique, dentre as regras de dedução da base, quais podem ser utilizadas para auxiliar a verificação de restrições de integridade. Assim, de certa forma, é possível dizer que restrições que utilizam regras de dedução em sua verificação são dependentes das mesmas (PLEXOUSAKIS, 1993).

De acordo com essa realidade, classificam-se restrições de integridade de acordo com sua dependência sobre regras dedutivas. É obvio que essa classificação depende

não somente do conceito da restrição, pois depende também de sua especificação, das características do SGBD utilizado e das regras dedutivas. Em relação à *dependência sobre regras dedutivas*, uma restrição de integridade é classificada como: (i) *indeterminada*, quando o SGBD utilizado não possuir características dedutivas, (ii) *dependente*, quando o SGBD utilizado for dedutivo e a verificação da restrição depender de alguma forma de regra(s) dedutiva(s) relacionada(s) à base em questão e, (iii) *independente*, quando o SGBD utilizado for dedutivo e a verificação da restrição independer de qualquer regra dedutiva.

Restrições *dependentes de regras dedutivas* são ainda sub-classificadas em duas classes distintas que são: (i) a classe das restrições *diretamente dependentes*, que compõe restrições cuja verificação utiliza diretamente regras dedutivas e, (ii) a classe das restrições *transitivamente dependentes*, que engloba as restrições que, de acordo com o critério *completude* (Seção 4.3.14) do aspecto *substância*, tenham sido geradas por alguma restrição que dependa diretamente ou transitivamente de regras dedutivas para sua verificação.

A fim de exemplificar a utilização deste critério, considere a existência de um SGBD dedutivo que controle uma base de dados com os ramais telefônicos de funcionários e departamentos em uma empresa qualquer. Nesta base, existe a seguinte regra dedutiva *"O prefixo do ramal telefônico de um funcionário é idêntico ao do departamento onde ele trabalha."*. Nesse contexto, a seguinte restrição seria *diretamente dependente* desta regra dedutiva: *"O volume mensal de telefonemas provenientes de um departamento não pode ultrapassar vinte horas."*. Isso ocorre pois, a restrição utilizaria a regra dedutiva a fim de identificar os departamentos referentes a cada uma das ligações efetuadas pelos funcionários da empresa.

De acordo com essa classificação, a Figura 4.20 ilustra graficamente o critério *dependência sobre regras dedutivas*.

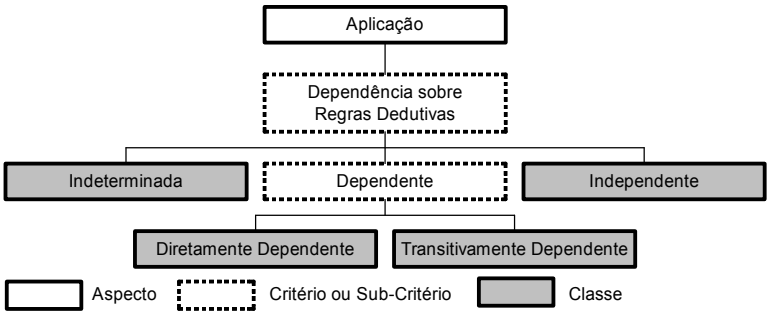


Figura 4.20: Critério *Dependência sobre Regras Dedutivas*

4.5.2 Ordem de Precedência

O principal objetivo deste critério na classificação de restrições é classificá-las de acordo com a relação de ordem de ocorrência de suas verificações. Vale lembrar, que considera-se como verificação de uma restrição de integridade não somente a análise de sua condição restritiva, mas também a execução de possíveis ações corretivas.

Em relação à *ordem de precedência*, uma restrição pode ser considerada *livre*, quando sua verificação não estiver relacionada à verificação de nenhuma outra restrição, ou *ordenada*, quando existir alguma relação de ordem entre sua verificação e a verificação de outra(s) restrição(ões).

Restrições *ordenadas* podem possuir *ordem de precedência* de verificação de dois tipos: (i) *intra-transação*, quando a ordenação estiver relacionada somente a uma transação de forma que a ordem previamente estabelecida seja obedecida somente entre

as verificações de restrições relacionadas a mesma transação, ou (ii) *inter-transação*, quando a ordem previamente estabelecida tiver que ser obedecida por todas as verificações de restrições de integridade.

Além disso, com base em (DAYAL et al., 1988; RAM et al., 1997), restrições *ordenadas* em qualquer um dos dois tipos de ordenação acima podem ser consideradas *dependentes*, quando sua verificação depender da verificação prévia de outra(s) restrição(ões), ou *dependidas*, quando sua verificação for necessária para que outra(s) restrição(ões) possa(m) ser verificada(s).

Dessa forma, em relação à *ordem de precedência*, restrições de integridade ordenadas podem ser de quatro grupos: *intra-transação dependentes*, *intra-transação dependidas*, *inter-transação dependentes* ou *inter-transação dependidas*. Caso uma restrição de integridade *ordenada* se enquadre em somente um desses grupos (dependente ou dependida), a mesma será dita *homogênea* e, caso contrário (dependente e dependida), ela será dita *heterogênea*. Dessa forma, a classificação de restrições de integridade em relação à *ordem de precedência* foi definida de acordo com a Figura 4.21.

É importante citar que este critério está intimamente ligado aos critérios *completude* (Seção 4.3.14) e *propósito* (Seção 4.3.9) do aspecto *substância*, pois na maioria das vezes, restrições *ordenadas* também são integrantes da classe de restrições *incompletas*, por representarem, em conjunto com outras, alguma restrição complexa da realidade modelada. Além disso, elas também costumam ser da classe de restrições de propósito *causativo*, pois a verificação de restrições posteriores pode depender das ações corretivas de restrição(ões) verificadas anteriormente. Além disso, a ordem de precedência também pode ser útil em sistemas onde diversas transações ocorrem concorrentemente de forma que a verificação de restrições possa necessitar de uma certa ordenação.

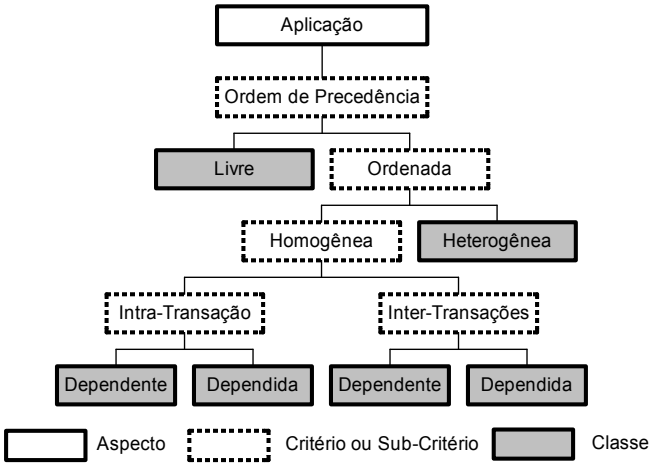


Figura 4.21: Critério *Ordem de Precedência*

Restrições *livres* são comuns e diversas delas já foram apresentadas em exemplos anteriores. Entretanto, exemplos de restrições *ordenadas* são mais raros. Assim, considere novamente a base de dados de uma aplicação que controla pousos e decolagens em um aeroporto. Nessa base, a seguinte restrição poderia existir: *"É desejável que todos os procedimentos de segurança sejam efetuados em qualquer pouso ou decolagem."*. Nota-se claramente que a manutenção desta restrição na base de dados envolve vários passos distintos. Além disso, a forma com que foi executado um procedimento ou a não execução do mesmo pode influenciar os procedimentos

posteriores. Nesse caso, é quase imprescindível que a restrição seja dividida em restrições mais simples relacionadas a cada um desses procedimentos. Sendo assim, cada uma dessas novas restrições seria classificada como *ordenada* em relação a este critério e classificada como *incompleta* e *gerada*, em relação ao critério *completude* (Seção 4.3.14) do aspecto *substância*.

4.5.3 Ativação e Desativação

Restrições de integridade relacionadas a bases de dados podem estar ativas e conseqüentemente exigirem que os dados respeitem suas condições restritivas ou, inativas, quando, temporariamente, não estiverem influenciando a manutenção da integridade da base de dados. Considerando esse fato, os processos de *ativação e desativação* de uma restrição de integridade são muito importantes, pois a ocorrência de qualquer um dos dois reflete grandemente na manutenção da integridade dos dados do banco de dados. Sendo assim, faz-se necessária uma classificação de restrições de integridade em relação à sua *ativação e desativação*.

A *ativação e desativação* de uma restrição de integridade pode ser: (i) *livre*, quando sua ocorrência for independente dos dados da própria base de dados ou, (ii) *restrita* quando sua ocorrência depender de alguma forma dos dados presentes na base. Dentre as restrições desta segunda classe existem restrições *restritas à existência*, que possuem ativação e desativação intimamente ligadas à existência de *objeto(s)*, *relacionamento(s)* e/ou *valor(es) de atributo(s)* com determinadas características comuns na base de dados ou, *restritas a não existência*, que possuem ativação e desativação ligadas à não existência dos mesmos. Existe ainda a possibilidade de que uma mesma restrição da classe *restrita* se enquadre em mais do que uma das classes descritas acima, sendo assim chamada de *heterogênea*. A Figura 4.22 exibe de forma gráfica a classificação segundo o critério *ativação e desativação*.

É importante notar que qualquer restrição de integridade com *ativação e desativação restrita* pode também ser representada através de uma restrição *livre* que possua uma condição explícita em relação ao aspecto restringente. Dessa forma, por exemplo, uma restrição *restrita à existência* de um *objeto* poderia ser representada através de uma restrição *livre* cuja condição restritiva possua uma condição referente à existência do objeto em questão. Nesse caso, em linguagem coloquial, a condição restritiva seria algo como: *"Se o objeto existir, deverão ser verdadeiras as condições..."*. Nota-se que a verificação das restrições no segundo caso é menos otimizada, pois mesmo que o objeto não exista, sua condição restritiva continuará sendo analisada em seus pontos de verificação.

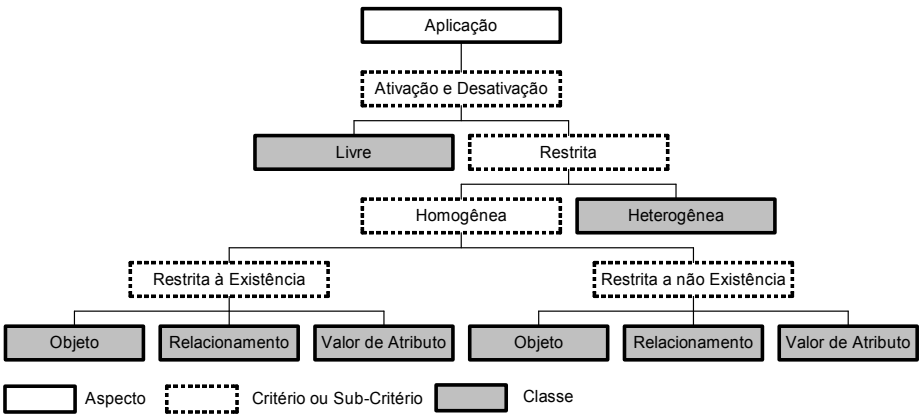


Figura 4.22: Critério *Ativação e Desativação*

4.5.4 Tipo de Ativação e Desativação

Assim como foi citado no critério anterior, tanto a ativação como a desativação de uma restrição de integridade reflete de forma importante no processo de manutenção da integridade dos dados de qualquer tipo de base.

Dessa forma, com base em (DAYAL et al., 1988), utilizou-se mais um critério relacionado à ativação e desativação de restrições, a fim de classificá-las de acordo com seu *tipo de ativação e desativação*. Como resultado, restrições de integridade podem possuir *tipo de ativação e desativação independente*, quando seus procedimentos de ativação e desativação não estiverem relacionados aos de nenhuma outra restrição, ou *tipo de ativação e desativação dependente* quando seus procedimentos de ativação e desativação estiverem relacionados respectivamente aos de outra(s) restrição(ões) de integridade, de forma que os mesmos só possam ocorrer em conjunto. A Figura 4.23 ilustra a classificação sobre este critério.

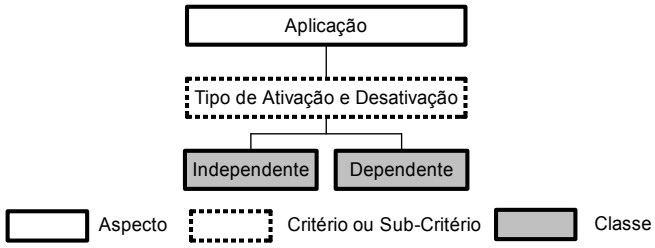


Figura 4.23: Critério *Tipo de Ativação e Desativação*

Normalmente, este critério está ligado ao critério *completude* (1.3.14) do aspecto, pois, na maioria das vezes, restrições de integridade com *tipo de ativação e desativação dependente* são classificadas como *incompletas* e, restrições de integridade com *tipo de ativação e desativação independente* são classificadas como *completas*. Isso ocorre, pois uma restrição *incompleta* representa, em conjunto com outra(s), uma única restrição de integridade do mundo real e, deve trabalhar em conjunto com ela(s).

4.5.5 Acionamento

O termo *acionamento* de uma restrição de integridade é utilizado para representar o inicio de seu processo de verificação de validade. Uma restrição de integridade possui *acionamento*: (i) *manual*, quando o início de sua verificação for determinado de forma explícita pelo administrador da base de dados (DBA) ou por qualquer usuário comum, com a utilização de um comando isolado ou em um programa, como por exemplo, uma aplicação de auditoria (SANTOS, 1980), ou (ii) *automático*, quando sua verificação sempre se iniciar de forma automática, devido ao controle exercido pelo SGBD, sempre que for alcançado algum *ponto de verificação* relacionado à restrição em questão.

Sempre que o *acionamento* de uma restrição de integridade se enquadrar somente em um dos dois tipos acima, o mesmo será dito *homogêneo*. Além disso, também existe a possibilidade de ocorrência de restrições de integridade com *acionamento heterogêneo*, por englobarem características de ambos tipos de acionamento. Assim, o *acionamento* destas restrições ocorre nos pontos de verificação pré-definidos e também em pontos onde ocorrer uma requisição explícita de verificação de validade, por parte do DBA ou de outro usuário. Através da Figura 4.24, é possível a visualização gráfica da classificação em relação ao critério *acionamento*.

Vale ainda notar que *restrições homogêneas manuais*, não possuem pontos de verificação pré-definidos e que as demais restrições possuem pontos de verificação que devem ser definidos juntamente com sua especificação.

Exemplos de restrições com *acionamento manual* são bastante comuns. Dentre eles, têm-se restrições de domínio de atributos, integridade referencial, cardinalidade, entre outros. Restrições com *acionamento manual* são mais raras, mas é possível citar restrições não obrigatórias sobre metas de qualidade a serem alcançadas. Muitas vezes essas restrições possuem verificação complexa e dispendiosa, de forma a serem verificadas manualmente apenas quando necessário aos usuários do banco de dados.

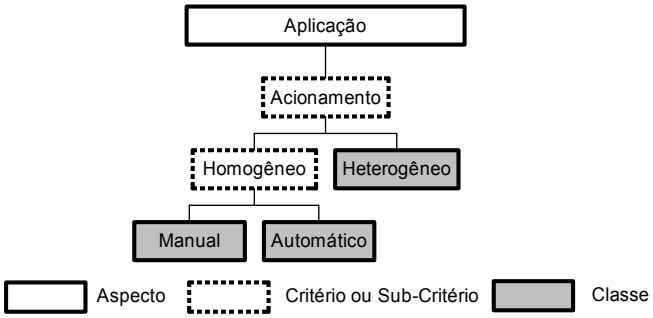


Figura 4.24: Critério *Acionamento*

4.5.6 Tratamento

O critério *tratamento* da classificação de restrições de integridade as classifica de acordo com o tipo de *tratamento* a ser recebido a fim de manter a integridade dos dados. O *tratamento* mais adequado para uma restrição de integridade depende de características conceituais e de implementação da própria restrição, além de característica intrínsecas ao SGBD utilizado.

Conceitualmente falando, o *tratamento* de *prevenção* é considerado ideal para restrições de integridade, pois, nele são estabelecidas e executadas medidas que impedem sua violação. Teoricamente, este tratamento é ideal, mas na prática, o mesmo costuma ser utilizado somente em restrições com condições restritivas não muito complexas, como por exemplo, restrições sobre o domínio de atributos. Em restrições mais complexas, este *tratamento* pode se tornar demasiadamente dispendioso, pois, muitas vezes, este tipo de restrição é verificado somente no final de grandes transações e, para esses casos, fica difícil prever se alguma ação ou conjunto de ações da transação viola a restrição.

O *tratamento* de *detecção* se limita a acusar a violação de uma restrição. Desta forma, após violada a restrição, ações corretivas devem ser efetuadas por parte do DBA ou de algum usuário da base de dados a fim de retorná-la a um estado íntegro. Este tratamento não é considerado completo, pois, restrições baseadas no mesmo não garantem a integridade dos dados da base de dados. Por outro lado, ele pode ser necessário quando as ações após a violação das restrições não forem padronizadas, necessitando assim, de uma análise profunda para cada caso. Considerando a base de dados de uma empresa qualquer, exemplos dessas restrições seriam metas de qualidade que são apenas desejáveis, não obrigatórias.

Restrições de integridade podem ainda receber o *tratamento* de *complementação* que consiste na execução de um conjunto de ações corretivas após a verificação de sua violação. Logicamente, estas ações devem ser sempre definidas antes do momento de ativação da restrição, sendo que, na grande maioria dos casos, as mesmas são definidas e especificadas em conjunto com a própria restrição. A aplicação deste *tratamento*,

sobre restrições com *ponto de verificação baseado em operação*, costuma ser facilmente efetuada em sistemas gerenciadores de bancos de dados comerciais através da utilização de *triggers* que são disparados nos pontos de verificação das respectivas restrições. Como exemplo dessas restrições tem-se a seguinte restrição: *"O salário de um funcionário deve ser maior ou igual ao salário base de sua categoria após três meses de contratação."*. Considerando que a violação dessa restrição é irreversível, ações para redefinir o salário dos funcionários devem ser tomadas após cada violação.

Além dos *tratamentos* citados até aqui, existe ainda o *tratamento* de *reconstituição*, que consiste na tomada de medidas, após uma violação de restrição, que reconstituam a base de dados ao estado sobre o qual foi aplicada a operação ou transação que violou sua integridade. Isto significa que, após a reconstituição, o estado atual da base de dados não deverá ter sofrido nenhuma influência por parte da operação ou transação que violou a restrição. Além disso, em casos de execução concorrente de transações, deve ser garantido que todas as transações executadas sobre a base de dados, exceto a que causou a violação, não sofram nenhuma influência em relação a este fato. Exemplos dessa classe de restrição são comuns. Dentre eles estão as restrições de integridade referencial do modelo relacional.

É de fácil percepção que somente restrições com *pontos de verificação baseados em operação* poderão ter *tratamento de reconstituição*. Nota-se também que a aplicação deste tipo de tratamento pode se tornar complexa e dispendiosa, pois, a mesma necessita de um controle sobre cópias de segurança (*backups*) da base de dados e também de registros sobre as operações efetuadas em transações (*transaction logs*), de forma a possibilitar a reconstituição da base de dados. Infelizmente, apesar de dispendioso, na maioria das vezes este tipo de *tratamento* se faz necessário, por não existirem ações de reparo genéricas o suficiente para solucionar os problemas causados por qualquer violação de uma mesma restrição ou pela necessidade de que a(s) ação(ões) que causou(aram) a violação não influencie(m) a base de dados de forma alguma.

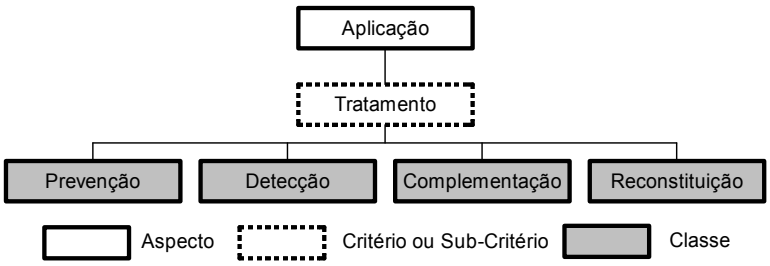


Figura 4.25: Critério *Tratamento*

Este critério da classificação de restrições está intimamente ligado ao critério *propósito* (Seção 4.3.9) do aspecto *substância*, pois, em muitos casos, uma restrição deverá receber seu *tratamento* de acordo com seu *propósito*. O *tratamento* de *prevenção* é aplicável a restrições de qualquer propósito, mas, é importante notar que, quando aplicado este tratamento, não ocorrem violações de restrições e, com isso, o propósito das mesmas não influirá de forma alguma no processo de manutenção da integridade dos dados e, ações reparatórias nunca serão executadas. Em relação aos três *tratamentos* restantes, a definição de qual deles será aplicado em uma restrição deve ocorrer de acordo com um dos casos a seguir: (i) caso a restrição de integridade possua *propósito condicional*, a mesma deverá receber *tratamento* de *reconstituição*, pois a base de dados não poderá ser influenciada pelo causador de sua violação, (ii) caso a restrição possua propósito *causativo*, o *tratamento* aplicável à mesma será o de *complementação*, pois

ações reparatórias deverão ser executadas sobre a base após sua violação e, (iii) caso a restrição possua *propósito verificativo*, a mesma deverá receber *tratamento* de *detecção*, pois sua violação será informada, mas a execução da operação ou transação que causou a violação da restrição será permitida e não serão executadas ações reparatórias.

A Figura 4.25 exibe graficamente a classificação de restrições de integridade em relação ao critério *tratamento*.

4.5.7 Veículo

Este critério da classificação de restrições de integridade se baseia no *veículo* utilizado para a verificação e manutenção da integridade de uma base de dados. Entende-se como *veículo* qualquer componente de software a quem foi designada a função manter uma base de dados integra, baseando-se nas suas restrições de integridade. É importante citar que em restrições com *propósito verificativo*, *acionamento manual* ou *acionamento heterogêneo*, o *veículo* utilizado para manter a base de dados íntegra deverá trabalhar em conjunto com o DBA ou outro usuário do SGBD.

É cabível a existência de um SGBD que monopolize totalmente o processo de verificação e manutenção da integridade de uma base de dados temporal de versões, mas na maioria das vezes, isso não ocorre devido à grande complexidade envolvida no processo. Assim, é comum que a tarefa de manutenção da integridade dos dados seja dividida entre distintos componentes de software. Dentre os *veículos* classificados para a manutenção da integridade de uma base de dados estão o *sistema operacional*, o *sistema gerenciador de bancos de dados*, os *programas de aplicação* e os *utilitários de auditoria*.

Refere-se ao termo *sistema operacional*, o software básico sobre o qual se apóia o SGBD, incluindo compiladores e o sistema de gerência de arquivos. Normalmente, a ele é designada a manutenção da integridade quanto a restrições cuja *origem* seja a *implementação*. Como exemplo deste tipo de restrição, é possível citar as restrições que definem os formatos dos campos de dados da base.

Quanto ao *sistema gerenciador de bancos de dados* (SGBD), é comum a manutenção da integridade dos dados em relação às restrições presentes na especificação do esquema, que não forem próprias do *sistema operacional*. Normalmente, a manutenção das restrições com *declaração implícita* é contemplada pelos procedimentos referentes às operações básicas deste *veículo*, enquanto a manutenção de restrições *explícitas* costuma ser contemplada por esses mesmos procedimentos ou por monitores de integridade, que consistem em componentes do SGBD, desenvolvidos especificamente para esta finalidade.

Quando os recursos oferecidos pelo SGBD são insuficientes, é comum a utilização de *programas de aplicação* para possibilitar a manutenção da integridade de bases de dados. É possível dizer que isto ocorre com grande freqüência, principalmente em sistemas, desenvolvidos sobre um SGBD tradicional, cuja realidade modelada possua características temporais ou de versões. A utilização deste *veículo* para a manutenção de integridade implica em diversos inconvenientes, como a duplicação de esforços, pois todas as aplicações que utilizarem a base deverão considerar a manutenção das restrições em questão, e a dificuldade de coordenação da evolução do conjunto de restrições referentes à base de dados.

Utilitários de auditoria também podem ser utilizados a fim de manter a integridade de dados em uma base temporal de versões. Eles costumam ser executados de forma periódica, baseada em períodos constantes, na ocorrência de determinados eventos ou

até mesmo em momentos determinados pelo DBA ou outro usuário. Muitas vezes, cabe a estes reportar falhas de integridade dos dados em relação a restrições de *propósito verificativo* de forma a permitir a detecção das ações causadoras e das conseqüências relacionadas à quebra da integridade dos dados. Além disso, é possível a execução de ações corretivas com o intermédio do DBA ou outro usuário.

Considera-se que a utilização de qualquer um dos dois últimos *veículos* descritos, a fim de manter a integridade de uma base de dados, é devida a incapacidade ou inadequação do SGBD em relação às necessidades existentes (SANTOS, 1980).

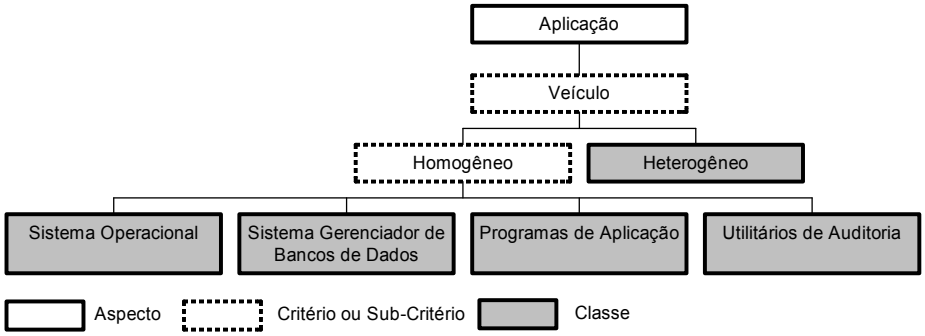


Figura 4.26: Critério *Veículo*

De acordo com o que foi descrito, classifica-se uma restrição de integridade, quanto ao critério *veículo* (Figura 4.26), como *homogênea*, quando todo o processo de verificação e manutenção da integridade dos dados, em relação a ela, estiver relacionado somente a um dos veículos descritos anteriormente, ou *heterogênea*, quando forem utilizados dois ou mais *veículos* distintos durante a verificação e manutenção de integridade relacionada à restrição. A fim de exemplificar situações onde mais do quer um *veículo* é utilizado por uma restrição de integridade, tem-se restrições de *propósito verificativo*, as quais podem ser verificadas através de um determinado *veículo* e, após intervenção do DBA ou outro usuário da base de dados, utiliza-se outro veículo para efetuar ações corretivas sobre os dados da base.

4.5.8 Inspeção

A fim de verificar a validade de uma restrição de integridade, são necessárias *inspeções* sobre os componentes de seus conjuntos restringente e restringido de forma a relacioná-los, através de sua condição restritiva. Considerando somente a presença dos dados existentes na base de dados, *inspeções diretas* aos componentes do restringido e restringente são necessárias na validação de uma restrição. Entretanto, com o propósito de melhoria de performance das operações realizadas por um SGBD, é comum o armazenamento de *metadados* sobre os dados reais da base. Dentre os *metadados* mais freqüentes estão médias, totais, cardinalidade, além de diversos dados estatísticos. Um dos propósitos relacionados ao armazenamento destes *metadados* é a sua utilização durante o processo de verificação de validade de restrições sobre os dados reais da base de dados, de forma a evitar inspeções *diretas* a eles. A essa forma de inspeção, dá-se o nome de inspeção *indireta*.

Existe ainda a possibilidade de que os *metadados* de uma base de dados possam ser utilizados para auxiliar somente em parte da verificação de uma restrição, necessitando assim, o acesso aos dados da base para complementar a verificação. Nestes casos, a inspeção é dita *heterogênea*. Dessa forma, o critério *inspeção* foi definido de acordo com a Figura 4.27.

É interessante notar que, considerando somente essa utilização para os *metadados*, a inspeção *indireta* será vantajosa todas as vezes que a manutenção dos *metadados* envolvidos for menos dispendiosa do que a inspeção direta dos dados necessários para a verificação da validade da restrição.

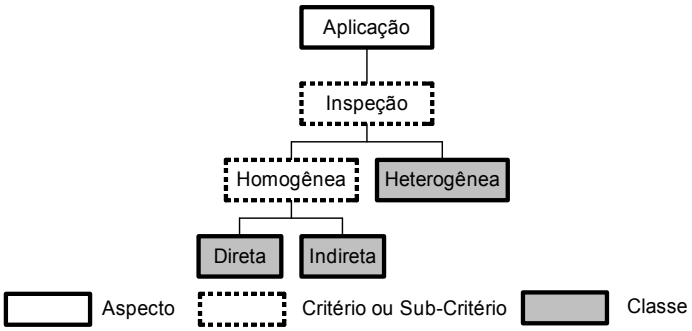


Figura 4.27: Critério *Inspeção*

4.5.9 Ponto de Verificação

Considera-se como *ponto de verificação* de uma restrição de integridade o momento em que se inicia o processo de verificação de validade da restrição em questão em relação ao estado atual da base de dados. A definição do conjunto de *pontos de verificação* relacionados a uma restrição de integridade deve obrigatoriamente ser definida antes de sua ativação, exceto em restrições de *acionamento manual*. Normalmente, essa definição ocorre em conjunto com a especificação da própria restrição. Além disso, é importante considerar que em restrições com *ordem de precedência* a definição deste conjunto de pontos deverá ser baseada nos pontos de verificação de outras restrições.

Pontos de verificação podem se basear em *operações* ou *eventos*. Pontos do primeiro tipo são baseados em *operações* executadas sobre os dados da base de dados. Eles se baseiam na ocorrência de comandos de manipulação de dados executados sobre a base em questão. Com base em (SANTOS, 1980; DOUCET et al., 1997), estes pontos podem ainda ser divididos em *imediatos* quando determinam momentos de verificação imediatamente posteriores à execução de cada operação ou *protelados* quando determinam momentos de verificação posteriores à execução de transações completas, não havendo verificação durante a execução das mesmas. Pontos de verificação também podem se basear na ocorrência de *eventos*. Esses eventos são muitas vezes representados através de condições a serem satisfeitas e podem ser *internos* ou *externos* em relação ao SGBD. Dentre os eventos considerados *internos* os mais comuns são *eventos* relacionados à verificação de outras restrições de integridade, os quais devem ser considerados em restrições com *ordem de precedência* de execução. Os eventos *externos* podem estar relacionados à ocorrência de fatos *temporais* ou *não temporais*.

Portanto, em relação ao critério *ponto de verificação* (Figura 4.28), classifica-se uma restrição de integridade como *homogênea*, quando todos os integrantes de seu conjunto de *pontos de verificação* forem de mesmo tipo ou, *heterogênea* quando os integrantes desse conjunto forem de dois ou mais tipos distintos. Restrições homogêneas são ainda sub-classificadas de acordo com as características de seus *pontos de verificação*.

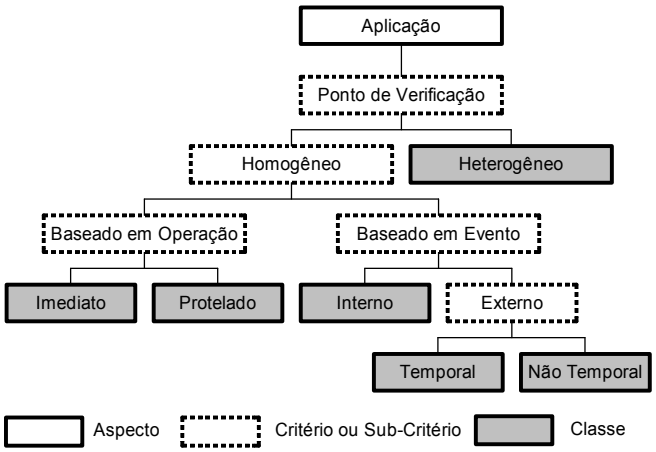


Figura 4.28: Critério *Ponto de verificação*

4.6 Temporalidade

Um dos principais aspectos desta classificação de restrições de integridade em base de dados temporais de versões é o aspecto *temporalidade*. Este aspecto engloba critérios com o intuito de classificar restrições de integridade de acordo com suas características temporais. É importante notar que os aspectos aqui considerados dizem respeito à temporalidade em relação aos conjuntos restringente e restringido envolvidos pela restrição e não à temporalidade da própria restrição, como foi considerado nos critérios *temporalidade da restrição* (Seção 4.4.2) e *abrangência da temporalidade da restrição* (Seção 4.4.3), presentes no aspecto *especificação*.

Também se deve citar que este aspecto não é totalmente disjunto do aspecto *substância*, pois muitos dos critérios aqui considerados poderiam fazer parte do aspecto *substância*. Apesar disso, considerando o fato de que os conceitos de tempo e de versão compõem o foco principal desta classificação, foram criados aspectos que englobam a maior parte tanto das características temporais como das características de versionamento de restrições de integridade em bases temporais de versões.

Dessa forma, considerando a temporalidade da restrição, relacionada aos dados por ela envolvidos, classificam-se restrições de integridade como *não temporais* e *temporais* (Figura 4.29).

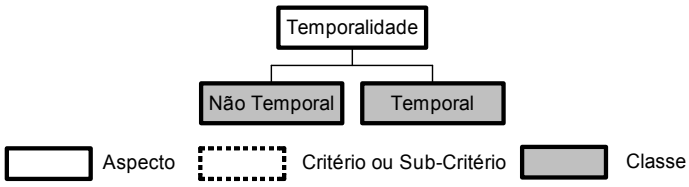


Figura 4.29: Aspecto *Temporalidade*

Restrições de integridade *não temporais* não possuem em sua composição nenhum aspecto relacionando ao tempo, os dados por ela envolvidos. Segundo, (BÖHLEN, 1994), esta classe de restrições engloba restrições de integridade tradicionais, ou seja, restrições não relacionadas ao tempo. Obviamente, este tipo de restrição de integridade deve obrigatoriamente pertencer à classe de restrições com *abrangência de estados estática*, de acordo com a classificação pelo aspecto *substância*. Este fato é claro, pois todas as restrições de integridade com *abrangência de estados dinâmica*, utilizam o conceito de tempo de forma implícita ou explícita.

Restrições de integridade *temporais* são restrições que utilizam o conceito de tempo de forma implícita ou explícita. Não existe restrição quanto à abrangência de estados deste tipo de restrição, pois, elas podem abranger somente um estado (*estática*) ou diversos estados (*dinâmica*) da base de dados. Caso uma restrição de integridade *dinâmica* não referencie o tempo explicitamente, certamente, a mesma referenciará o tempo de forma implícita através de operadores que referenciem valores anteriores (*old*) e posteriores (*new*) à execução de um conjunto de operações sobre os dados da base.

É muito importante notar a noção de *temporalidade*, aqui considerada, depende diretamente do tempo real. Dessa forma, pode-se dizer que restrições de integridade *temporais* dependem da existência de um "*relógio*" cujo fluxo independa de qualquer ação efetuada sobre a base de dados. Assim, essas restrições possuem uma característica interessante, pois sua violação, quando relacionada diretamente ao tempo, é irreversível devido à irreversibilidade do fluxo do tempo (CHOMICKI, 1992). Isto significa que não existe uma forma de efetuar um retorno (*rollback*) ao estado anterior à violação da base de dados, pois a violação não foi devida à alteração de estado da mesma. A solução para este problema costuma consistir na execução de ações compensatórias sobre os dados. Com isso, é comum que essas restrições possuam *propósito causativo*, relacionadas a um conjunto de ações compensatórias, ou *verificativo*, para que sua violação não impeça a utilização da base de dados.

Considerando o principal enfoque deste aspecto da classificação que é a temporalidade dos dados envolvidos pela restrição, restrições de integridade *temporais* foram sub-classificadas de acordo com diversos sub-critérios, os quais serão detalhadamente descritos durante as seções seguintes.

A fim de evitar desentendimentos, deve-se citar que, apesar de haver, de acordo com os critérios estabelecidos para a sub-classificação de restrições desta classe, a possibilidade de existência de restrições cujos restringentes e restringidos possuam tipo de tempo *não temporal*, obviamente, não será considerada esta possibilidade, pois, neste caso, é claro que as restrições deverão ser classificadas como *não temporais*.

4.6.1 Tipo de Tempo do Restringente

Conforme citado anteriormente, o conjunto restringente é composto por diversos componentes, os quais podem fazer parte da própria base de dados ou não. Integrantes do restringente provenientes da própria base de dados podem possuir características temporais implícitas, enquanto integrantes não provenientes da base de dados podem estar explicitamente relacionados ao tempo.

De acordo com essa realidade, os integrantes do restringente podem ser *não temporais* ou possuir os seguintes tipos de tempo: *tempo de transação*, que se refere ao momento em que fatos são armazenados na base de dados, *tempo de validade,* que se refere ao período em que os fatos são válidos na realidade modelada, *bitemporal*, que se refere a ambos tipos de tempo mencionados anteriormente, ou *tempo definido pelo usuário*, que se refere à temporalidade através de atributos, de domínio data e hora, não interpretáveis facilmente (BÖHLEN, 1994; JENSEN et al., 1998; ESCOFET, 2001). Integrantes do restringente, de tipo *domínio*, que representam valores temporais também são classificados com este último tipo de tempo.

Um integrante do restringente é dito *não temporal* quando, em relação ao aspecto *substância*, o mesmo for de tipo *esquema*, *entidade* ou *relacionamento*, definido como *não temporal* na modelagem de dados ou, quando o mesmo for de tipo *domínio* e não estiver relacionando explicitamente parte da base de dados ao tempo. Considera-se ainda que, integrantes do restringente, que possuam algum tipo de temporalidade

implícita ou explícita, podem estar relacionados ao *tempo com determinação*, se referindo a momentos com exatidão, ou ao *tempo com indeterminação*, se referindo a momentos com pouca precisão.

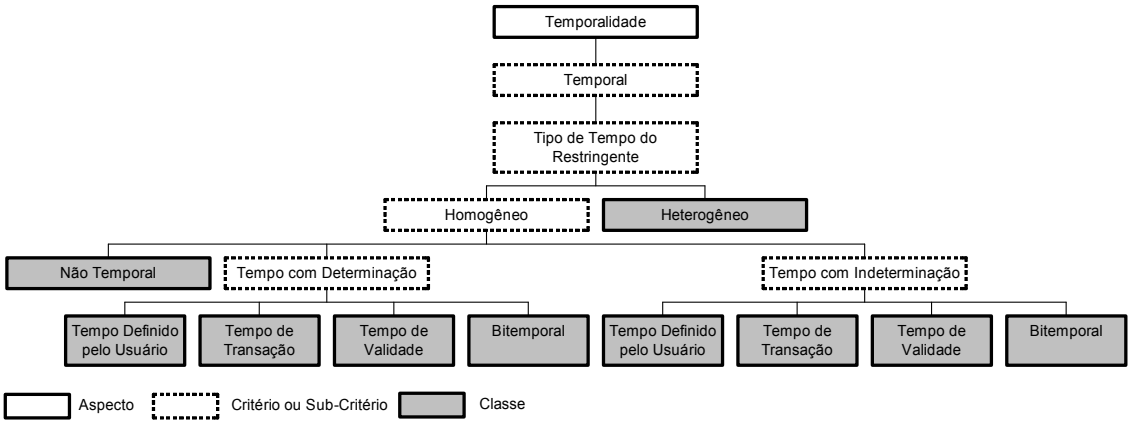


Figura 4.30: Sub-critério *Tipo de Tempo do Restringente*

Segundo (COWLEY; PLEXOUSAKIS, 2000), a utilização do *tempo com indeterminação* é comumente necessária quando existem incertezas na medição do tempo ou diferenças na granularidade temporal dos elementos a serem considerados. Desta forma, este sub-critério da classificação pode estar relacionado ao sub-critério *granularidade temporal* (Seção 4.6.6) que será definido adiante. O tipo mais comum de indeterminação temporal está relacionado ao *tempo de validade* ou ao *tempo definido pelo usuário*. A indeterminação relacionada ao *tempo de transação* é rara, pois esses tempos costumam ser conhecidos com exatidão (JENSEN et al., 1998).

Assim, em relação ao sub-critério *tipo de tempo do restringente*, classificam-se restrições de acordo com as seguintes classes: (i) *homogêneas*, quando todos os integrantes de seu restringente possuírem o mesmo tipo de tempo ou, (ii) *heterogêneas*, quando os integrantes do restringente possuírem dois ou mais tipos distintos de tempo. Além disso, restrições da classe das *homogêneas* são sub-classificadas de acordo com o tipo de tempo de seus restringenes. A fim de prover um melhor entendimento, a Figura 4.30 ilustra graficamente a classificação, segundo este sub-critério.

4.6.2 Tipo de Tempo do Restringido

O sub-critério *tipo de tempo do restringido* (Figura 4.31) é praticamente análogo ao sub-critério anterior, mas diz respeito aos conjuntos restringidos de restrições de integridade. Além de estarem relacionados a conjuntos diferentes, existem poucos fatores que diferem este sub-critério de seu anterior. O primeiro está relacionado ao fato de que não existe a possibilidade de integrantes do restringido de tipo *domínio*. Dessa forma, integrantes desse conjunto, são considerados *não temporais*, quando forem de tipo *esquema*, *entidade* ou *relacionamento* especificados como *não temporais* durante sua modelagem. Já o segundo fator está relacionado ao fato de que não existe a necessidade de classificação da homogeneidade do *tipo de tempo de seus restringidos*.

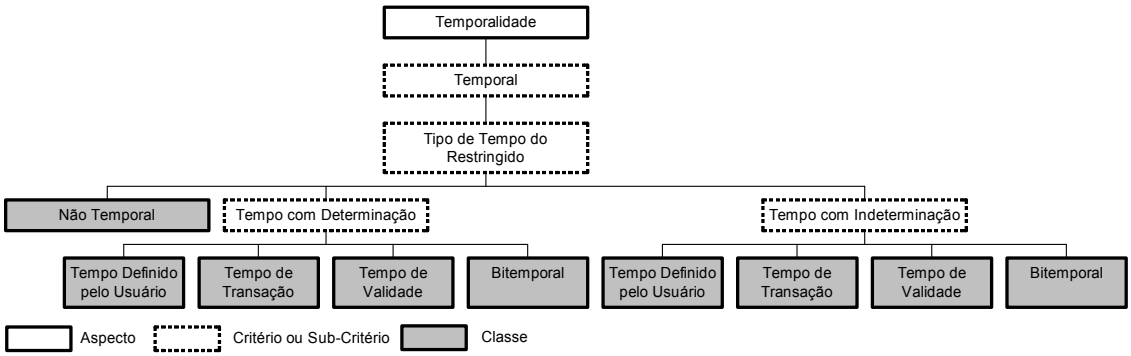


Figura 4.31: Sub-critério *Tipo de Tempo do Restringido*

4.6.3 Referência

Referências presentes em condições restritivas devem relacionar os conjuntos restringido e restringente de forma a restringir a gama de valores possíveis do restringido, com base no conteúdo dos integrantes do restringente. A princípio, baseando-se em (DAYAL et al., 1988), conclui-se que essas referências podem utilizar duas noções distintas de tempo: *tempo absoluto* ou *tempo relativo*.

Segundo (JENSEN et al., 1998), a noção de *tempo absoluto* utiliza tempos específicos independentes de qualquer outro tempo. Quando é utilizada esta noção temporal, diz-se que existe uma relação *quantitativa* entre o restringido e o restringente e, são especificados valores absolutos para a duração de evento(s) ou restringidas as distâncias temporais entre dois ou mais eventos (MEIRI, 1996). Esta noção temporal é comumente representada através da utilização de datas, durações ou *delays*. Como exemplo, tem-se a seguinte restrição em linguagem coloquial: *"O processo de fabricação do produto X não deve levar mais do que duas horas."*.

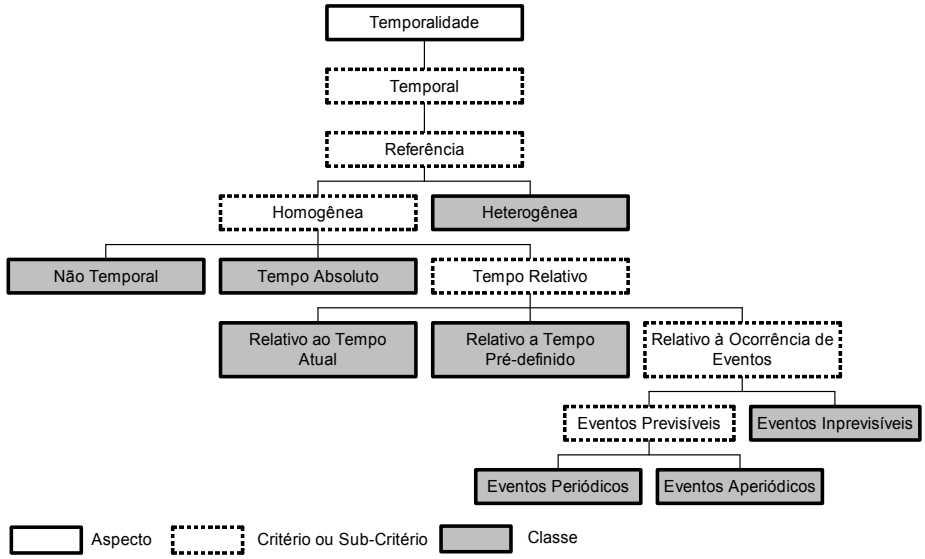


Figura 4.32: Sub-critério *Referência*

É possível afirmar que a noção de *tempo relativo* utiliza tempos relacionados a outros tempos (JENSEN et al., 1998). Dentre os possíveis tempos relacionados têm-se o *tempo atual* (*now*), *tempos pré-definidos* e *tempos de ocorrência de determinados eventos*. Como exemplos de *tempo pré-definido* têm-se datas especificadas explicitamente, tempos de validade de objetos da base de dados, entre outros. Dentre os *tempos relativos à ocorrência de eventos* existe ainda uma subdivisão referente à natureza do evento em questão, que pode ser *previsível*, ou *imprevisível* e, dentre os

previsíveis, têm-se os *periódicos* e os *aperiódicos* (DOUCET et al., 1997; ESCOFET, 2001).

Quando é utilizada a noção de *tempo relativo*, diz-se que a relação entre o restringido e o restringente é *qualitativa* e, diversos operadores especiais podem ser utilizados, como por exemplo, *antes*, *depois* ou *durante* (MEIRI, 1996; BETTINI; WANG; JAJODIA, 1997). A seguinte restrição é um exemplo de utilização desta noção temporal *"O recebimento do aviso de pagamento de boleto deve ocorrer antes do envio da mercadoria."*.

Quanto ao sub-critério *referência* (Figura 4.32), classificam-se restrições de integridade como *homogêneas*, quando suas *referências* utilizam somente uma das diferentes noções de tempo descritas acima, ou *heterogêneas*, quando utilizam duas ou mais noções distintas de tempo. Dentre as *homogêneas*, existe ainda uma sub-classificação, de acordo com as noções de tempo descritas. Além disso, foi classificada a possibilidade de referências *não temporais*, para restrições com restringido ou restringente temporal, que não referenciem a temporalidade dos mesmos em suas condições restritivas.

4.6.4 Abrangência Temporal do Restringente

Assim como foi dito em relação à *abrangência do restringente* (Seção 4.3.10) no aspecto *substância*, um dos principais critérios a serem analisados durante a otimização do processo de verificação de restrições de integridade diz respeito ao escopo das restrições, ou seja, à área da base de dados a ser analisada para que a verificação da restrição possa ser realizada. Assim, é possível dizer que a abrangência de uma restrição de integridade está intimamente ligada à complexidade de seu processo de verificação.

A *abrangência do restringente* (Seção 4.3.10) e a *abrangência de ligação do restringente* (Seção 4.3.12)do aspecto *substância* não levam em consideração aspectos temporais, considerando apenas um estado da base de dados. Com isso, faz-se necessário o sub-critério *abrangência temporal do restringente* para que, em conjunto com os critérios da *substância*, seja possível a análise da abrangência do conjunto restringente em relação a diversos estados da base ao longo de seu tempo de vida.

Dessa forma, a abrangência temporal de um integrante do restringente pode ser classificada como: (i) *indefinida*, quando o mesmo for de tipo *domínio* e não estiver diretamente relacionado ao tempo, *global*, quando o mesmo envolver todo o tempo de vida da base de dados, ou *local*, quando envolver somente parte do tempo de vida da base de dados. A abrangência *local* pode ainda envolver somente momentos no: (i) *passado*, através de um *ponto no tempo,* um *intervalo temporal*, ou um *elemento temporal*, (ii) *presente*, através de um *ponto no tempo*, (iii) *futuro*, através de um *ponto no tempo,* um *intervalo temporal*, ou um *elemento temporal*, (iv) momentos em dois ou mais desses períodos (*localmente heterogênea*), através de um *intervalo* ou *elemento temporal* ou, (v) tempos *independentes do tempo atual*, através de um *ponto no tempo,* um *intervalo temporal*, ou um *elemento temporal*. Este último tipo de abrangência costuma ser representado por integrantes do restringente, de tipo *domínio*, que representam valores temporais.

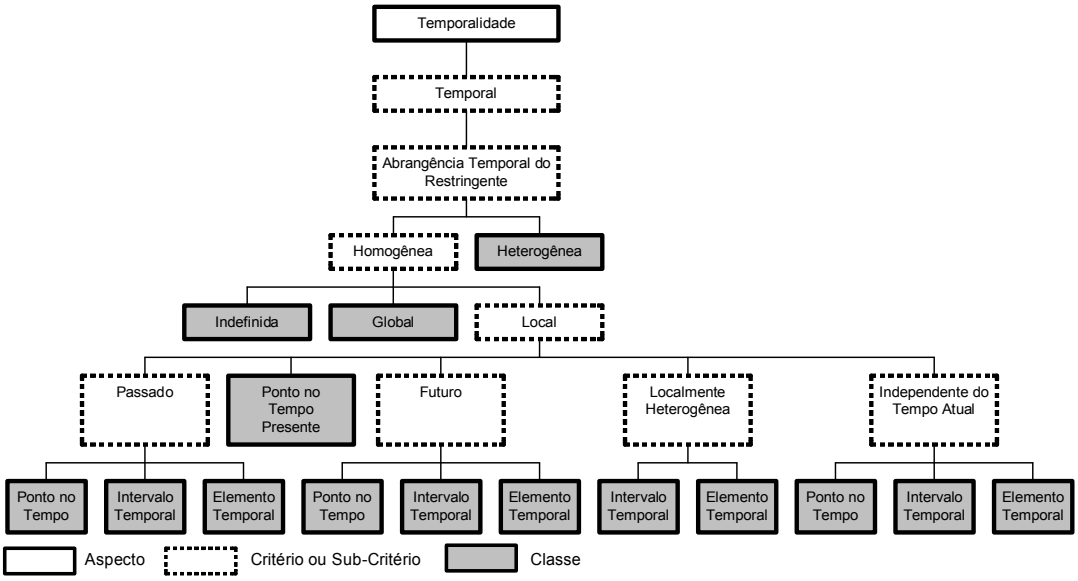


Figura 4.33: Sub-critério *Abrangência Temporal do Restringente*

Finalmente, classificam-se restrições de integridade, de acordo com a *abrangência temporal de seus restringentes*, em *homogêneas*, quando todos os seus integrantes se enquadrarem em uma única classe ou *heterogêneas* quando os integrantes se enquadrarem em duas ou mais classes distintas. Restrições *homogêneas* são ainda sub-classificadas de acordo com as classes descritas. A classificação quanto a este sub-critério é exibida através da Figura 4.33.

Como exemplo de classificação por este critério, considere a seguinte restrição em linguagem coloquial: *"O salário de um empregado não pode ser menor que o piso salarial de sua categoria após três meses de contratação."*. Considere também que esta lei entrou em vigor somente no dia primeiro de janeiro do ano de 1989. Assim, o conjunto restringido dessa restrição é composto pela constante *"três meses"* e por atributos e relacionamentos temporais, de acordo com o modelo da aplicação em questão. Em relação à constante *"três meses"*, a *abrangência temporal* é *indefinida*. Por outro lado, em relação aos outros componentes do restringido, a *abrangência temporal* é *localmente heterogênea* através de um *intervalo temporal* que se inicia no dia primeiro de janeiro de 1989 e não possui final definido. Sendo assim, a abrangência temporal do conjunto restringente como um todo é dita *heterogênea*.

4.6.5 Abrangência Temporal do Restringido

Pode-se dizer que este sub-critério é praticamente análogo ao sub-critério anterior, pois existem poucas diferenças entre os dois. A mais obvia é que considera-se aqui o conjunto restringido ao invés do restringente. Além disso, não existe a possibilidade de restringidos com *abrangência temporal indefinida*, pois não existem integrantes deste conjunto de tipo *domínio*. A abrangência relacionada a *tempos independentes do tempo atual* não se faz necessária para o restringido e, a última diferença diz respeito à impossibilidade de existência de restringidos com integrantes de tipos distintos, de forma extinguir a classificação quanto à homogeneidade da *abrangência temporal do restringido*. Assim, a classificação de restrições de integridade, quanto ao sub-critério *abrangência temporal do restringido*, pode ser visualizada através da Figura 4.34.

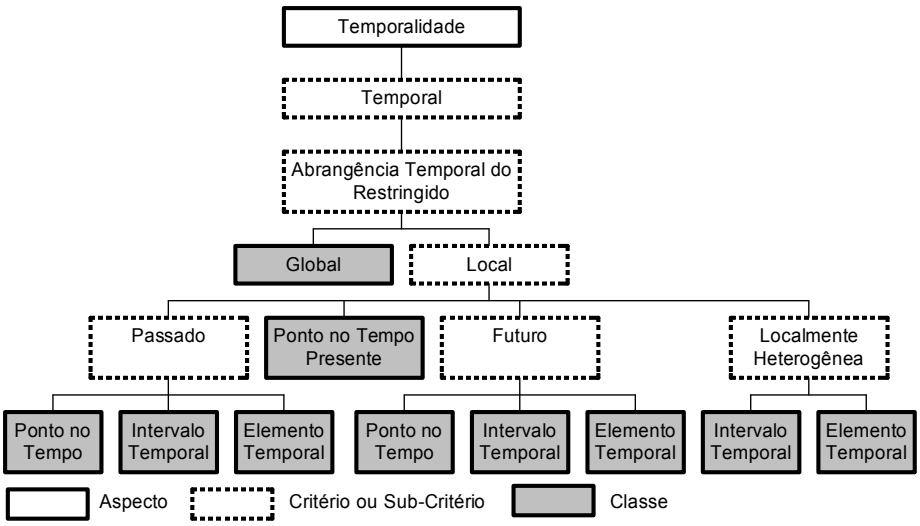


Figura 4.34: Sub-critério *Abrangência Temporal do Restringido*

4.6.6 Granularidade Temporal

Utiliza-se este sub-critério da classificação de restrições de integridade para distingui-las em relação à granularidade temporal dos integrantes de seus conjuntos restringentes e restringidos. O termo granularidade temporal refere-se à unidade de medida utilizada em referências temporais de restrições de integridade. Muitas vezes, esta granularidade está intimamente ligada à unidade mínima temporal (*chronon*) representada pela base de dados em questão. Diversas unidades de medida temporal podem ser utilizadas, dentre elas estão horas, períodos do dia, dias, meses, anos, horários comerciais, dias letivos e dias úteis (BETTINI; WANG; JAJODIA, 1997).

Em relação à *granularidade temporal*, classificam-se restrições de integridade em: restrições com granularidade *homogênea*, quando todos os integrantes de seus restringidos e restringentes possuírem uma única granularidade e, restrições com granularidade *heterogênea*, quando os integrantes de seus restringidos e restringentes possuírem granularidades diferentes. A Figura 4.35 ilustra essa classificação.

Assim como outros critérios desta classificação, a classificação por este sub-critério é considerada muito importante para a otimização do processo de verificação de restrições de integridade, pois a complexidade de verificação de uma restrição costuma ser diretamente proporcional à quantidade de granularidades distintas utilizadas pela mesma.

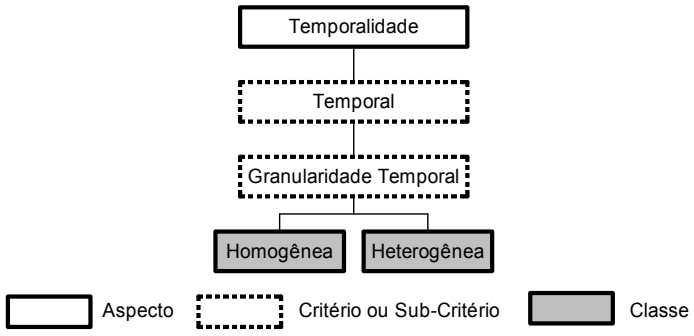


Figura 4.35: Sub-critério *Granularidade Temporal*

Devido à sua simplicidade, não foram consideradas classificações distintas para a granularidade do restringido e do restringente, de forma que o sub-critério *granularidade temporal* classifica a granularidade de ambos conjuntos. Apesar disso, caso necessária, é possível esta distinção considerando a *granularidade temporal homogênea* ou *heterogênea* em relação ao restringente e somente a *granularidade temporal homogênea* em relação ao restringido.

4.7 Versionamento

Além do conceito de versionamento da própria restrição, restrições de integridade em bases de dados temporais de versões podem utilizar o mesmo conceito em relação aos integrantes de seus conjuntos restringente e restringido. Dessa forma, faz-se necessária a criação do aspecto *versionamento* da classificação de restrições, com o intuito de distingui-las baseando-se em características de versionamento de seus restringidos e restringentes.

Assim como no aspecto anterior, é importante citar que este aspecto não está totalmente disjunto do aspecto *substância*. Optou-se pela criação deste aspecto com o intuito de possibilitar a classificação de restrições de integridade, em relação a maior parte de suas características de versionamento, através de um único aspecto.

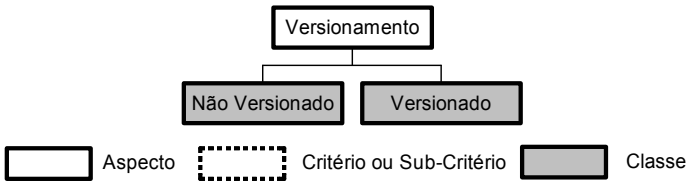


Figura 4.36: Aspecto *Versionamento*

Em um primeiro nível, classificam-se restrições de integridade como *não versionadas*, quando as mesmas não utilizam o conceito de versionamento em nenhum dos integrantes de seus restringidos e restringentes, ou *versionadas*, quando pelo menos um dos integrantes de seus restringentes ou restringidos possuírem características de versionamento. Obviamente, assim como ocorre no aspecto anterior, não é considerada válida a possibilidade de classificação de restrições de integridade *versionadas*, quando as mesmas forem classificadas *não versionadas*, em relação aos sub-critérios de *tipo de versionamento* (Seções 1.7.1 e 1.7.2) de ambos conjuntos restringente e restringido. Através da Figura 4.36 é possível uma visualização deste primeiro nível da classificação.

Considerando que o principal objetivo deste trabalho é classificar restrições de integridade em relação às suas características temporais e de versionamento, restrições de integridade *versionadas* são sub-classificadas de acordo com diversos sub-critérios que serão detalhadamente expostos nas seções seguintes.

4.7.1 Tipo de Versionamento do Restringente

Restrições de integridade *versionadas* podem possuir características de versionamento em integrantes de seu restringente e/ou restringido. Desta forma, definiu-se este sub-critério para possibilitar a classificação de restrições de integridade, em bases de dados temporais de versões, de acordo com as características de versionamento de seus restringentes.

Um integrante do restringente de uma restrição de integridade pode ser classificado como: (i) *versionado*, quando o mesmo for de tipo *esquema*, *entidade* ou

relacionamento, definido com características de versionamento durante a modelagem da base de dados ou, quando for de tipo *domínio*, referenciando explicitamente o versionamento de alguma parte da base de dados, ou, (ii) *não versionado*, quando o mesmo não estiver relacionado diretamente e nem mesmo indiretamente ao conceito de versão.

A partir dessa definição, restrições de integridade são classificadas, quanto ao sub-critério *tipo de versionamento do restringente*, como *homogêneas*, quando todos os integrantes de seus restringentes possuem o mesmo tipo de versionamento, ou *heterogêneas*, quando as mesmas possuem integrantes dos restringentes de dois ou mais tipos distintos. Restrições *homogêneas* são ainda sub-classificadas de acordo com o tipo dos integrantes de seus restringentes. A Figura 4.37 ilustra graficamente a classificação de restrições de integridade segundo este sub-critério.

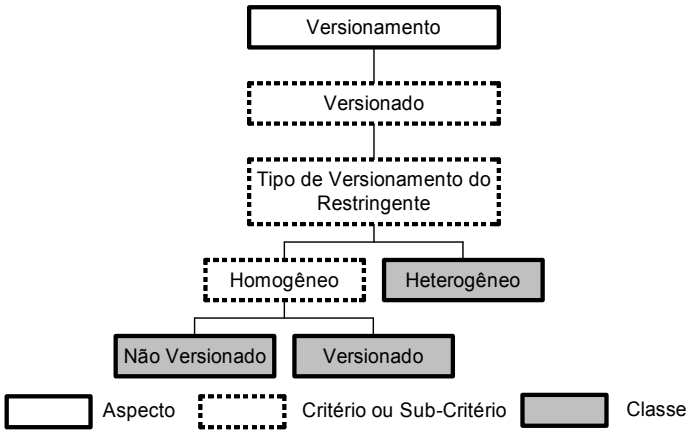


Figura 4.37: Sub-critério *Tipo de Versionamento do Restringente*

4.7.2 Tipo de Versionamento do Restringido

É possível dizer que este sub-critério é análogo ao anterior, exceto pelo fato de que, aqui são considerados os conjuntos restringidos das restrições de integridade e não é considerada a possibilidade de existência de restringidos com integrantes de tipos distintos, eliminando assim a necessidade de classificação da homogeneidade do *tipo de versionamento*. A Figura 4.38 ilustra a classificação de restrições de integridade quanto ao *tipo de versionamento de seus restringidos*.

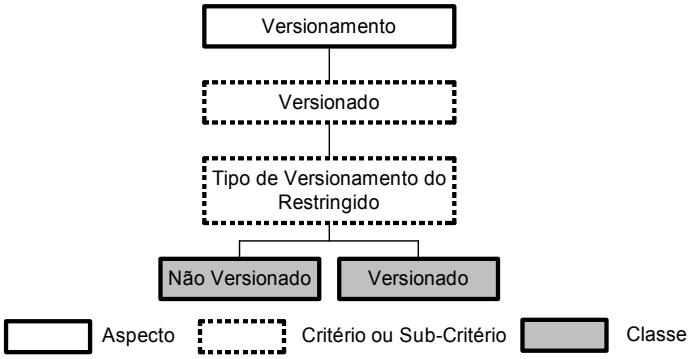


Figura 4.38: Sub-critério *Tipo de Versionamento do Restringido*

4.7.3 Abrangência de Versionamento do Restringente

Assim como foi visto em critérios anteriores, a abrangência do restringente de uma restrição de integridade diz respeito à quantidade de informação da base de dados que se necessita analisar durante o processo de verificação da validade da restrição em questão.

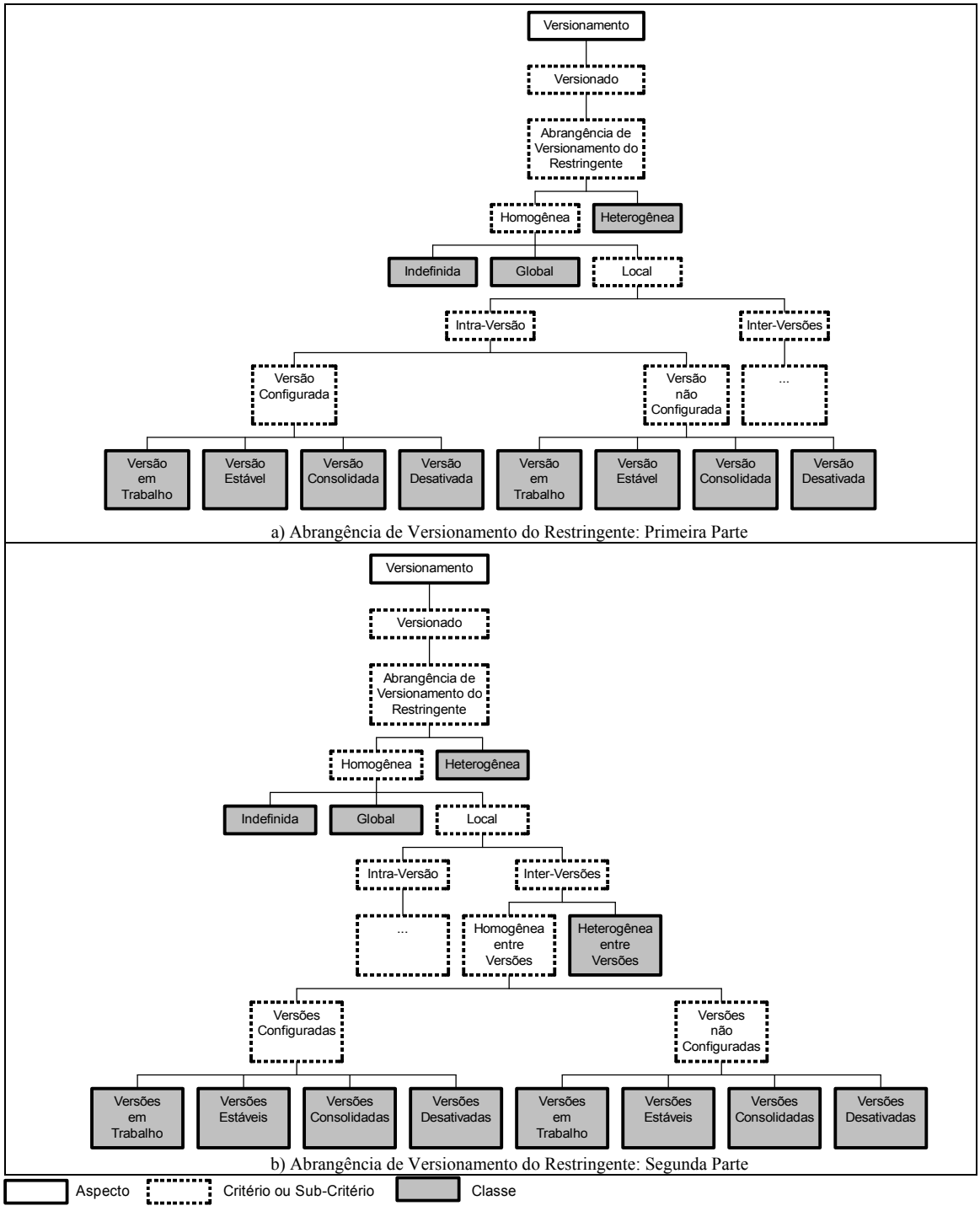


Figura 4.39: Sub-critério *Abrangência de Versionamento do Restringente*

De acordo com suas características de versionamento, um integrante do restringente pode possuir um dos seguintes tipos de abrangência: (i) *indefinida*, quando, em relação ao aspecto *substância*, o mesmo possuir o tipo *domínio* e não estiver diretamente relacionado ao versionamento de nenhum componente da base de dados, (ii) *global*, quando o integrante envolver todas as versões de determinado(s) componente(s) da base ou, (iii) *local*, quando envolver somente parte das versões de determinado(s) componente(s) da base de dados.

Com base em (MEDEIROS, 1992; DOUCET et al., 1997), a abrangência *local* de um integrante do restringente pode ainda estar relacionada a uma única versão (*intra-*

versão), que pode ser *configurada* ou *não configurada* e estará em um dos seguintes estados: em *trabalho*, *estável*, *consolidada* ou *desativada*. Além disso, a abrangência *local* também pode estar relacionada a duas ou mais versões distintas (*inter-versões*), as quais possuirão as mesmas características citadas acima. Com isso, integrantes do restringente, *locais* a duas ou mais versões, possuem abrangência de versionamento *homogênea entre versões*, caso todas as sua versões possuam características de estado e configuração idênticas. Caso contrário, os mesmos possuirão abrangência de versionamento *heterogênea entre versões*.

De acordo com essa discriminação, classificam-se restrições de integridade, quanto à *abrangência de versionamento de seus restringentes*, em *homogêneas*, quando todos os integrantes possuírem abrangência de mesmo tipo (*indefinida*, *global* ou *local*) ou, *heterogêneas*, quando as mesmas possuírem integrantes de dois ou mais tipos distintos. Restrições com abrangência *homogênea* são ainda sub-classificadas de acordo com o tipo de abrangência, a configuração e o estado dos integrantes de seus restringentes. A Figura 4.39, dividida em duas partes para facilidar a visualização, ilustra a classificação por este sub-critério.

É possível dizer que este sub-critério está fortemente relacionado aos critérios *abrangência do restringente* (Seção 4.3.10), *abrangência de ligação do restringente* (Seção 4.3.12) e *abrangência temporal do restringente* (4.6.4), pertencentes aos aspectos *substância* e *temporalidade*, sendo que através dos critérios pertencentes ao aspecto *substância*, classificam-se restrições de integridade segundo a abrangência do restringente em apenas um estado da base de dados. A abrangência do restringente relacionada a diversos estados da base, ao longo do tempo, é considerada através do sub-critério do aspecto *temporalidade*. Assim, é possível verificar que em nenhum desses critérios foi considerada a abrangência do restringente em relação a diferentes versões de componentes da base de dados. Dessa forma, fez-se necessária a criação do sub-critério *abrangência de versionamento do restringente* a fim de possibilitar, em conjunto com os anteriores, a análise e classificação completa de restrições de integridade, em relação à abrangência de seus restringentes.

4.7.4 Abrangência de Versionamento do Restringido

Pode-se dizer que este sub-critério é praticamente análogo ao anterior. A diferença mais óbvia é que se considera aqui o conjunto restringido ao invés do restringente. Além disso, não existe a possibilidade de restringidos com *abrangência de versionamento indefinida*, e a última diferença diz respeito à impossibilidade de existência de restringidos com integrantes de tipos distintos, de forma que não se faz necessária a classificação quanto à homogeneidade da *abrangência de versionamento do restringido*.

Dessa forma, a classificação de restrições de integridade, quanto ao sub-critério *abrangência de versionamento do restringido*, pode ser visualizada através da Figura 4.40.

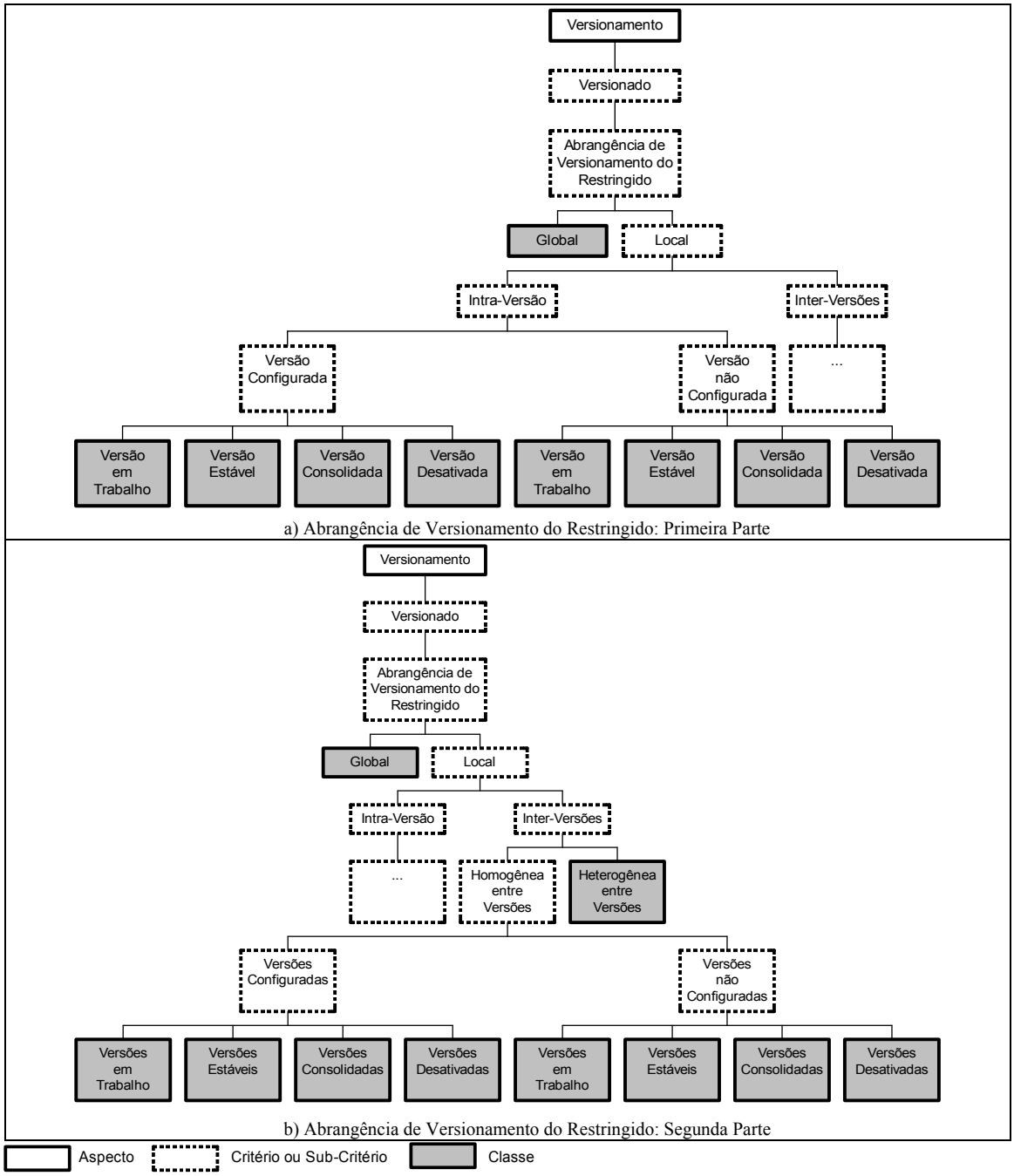


Figura 4.40: Sub-critério *Abrangência de Versionamento do Restringido*

4.8 Considerações Finais

Este capítulo apresentou uma classificação detalhada de restrições de integridade em bases de dados temporais de versões. Restrições foram analisadas sob sua *origem*, *substância*, *especificação*, *aplicação*, *temporalidade* e *versionamento*, possibilitando a criação de classes ortogonais para cada aspecto, desde que não houvesse perda em abrangência. Quanto às classes onde não foi possível total ortogonalidade, suas descrições apresentaram, de forma clara e precisa, todas as dependências existentes.

Esta classificação pode ser considerada como uma das principais contribuições do presente trabalho e, é importante deixar claro que ela poderá ser útil para qualquer trabalho relacionado à manutenção eficiente e eficaz da integridade dos dados de bases

temporais de versões, independentemente de outras características do modelo de dados ou do SGBD adotado.

A classificação proposta se diferencia de outras por priorizar e detalhar características temporais e de versionamento, não deixando de lado as outras características existentes. Além disso, devido à existência de aspectos específicos para a temporalidade e o versionamento, essa classificação pode ser facilmente adaptada para bancos de dados tradicionais ou para os que apliquem somente um desses conceitos.

O capítulo seguinte apresenta um estudo comparativo sobre a abrangência da classificação proposta. Nele, é demonstrado que a presente classificação é mais completa do que as outras encontradas na literatura. Devido a este fato, acredita-se que traga mais benefícios sua utilização como base de trabalhos de especificação e verificação de restrições de integridade do que a utilização de qualquer uma das outras classificações. Após essa análise, acredita-se que o próximo passo, em busca da manutenção da integridade de dados em um banco de dados temporal de versões, consiste na definição de formas para sua especificação.

5 ANÁLISE DE ABRANGÊNCIA DA CLASSIFICAÇÃO

Neste capítulo é apresentado um estudo que visa analisar a abrangência da classificação de restrições definida no Capítulo 4, a fim de mostrar que a classificação proposta é mais completa do que as outras encontradas na literatura.

Um dos principais objetivos deste trabalho como um todo é definir uma completa classificação de restrições de integridade para qualquer banco de dados temporal de versões, facilmente adaptável a outros tipos de bases de dados. De forma ideal, a completude da classificação deveria ser provada com base em métodos formais. Na prática, esses métodos costumam ser empregados para definir a semântica de linguagens e de sistemas. Seu uso é justificado por reduzir ambigüidades em especificações, como também prover meios para provar propriedades desejadas sobre os objetos especificados (NIELSON; NIELSON, 1992). Entretanto, no caso de uma classificação de restrições de integridade, não existe uma construção específica de linguagem que possa ser avaliada por um modelo matemático que descreva seu comportamento. Assim, apesar das vantagens decorrentes de um tratamento formal, seu uso não pode ser feito a fim de provar a completude da classificação proposta.

Com base nos fatos descritos, a alternativa encontrada para analisar a completude da classificação foi compará-la com outras encontradas na literatura. A fim de comparar as classificações, mapeou-se cada uma das classes de todas as classificações apresentadas no Capítulo 3 para classes equivalentes da classificação proposta no Capítulo 4. O resultado desta experiência indica que a classificação proposta engloba todas as classes representadas nesses outros trabalhos, além de classes não consideradas por eles. Assim, não fica provada formalmente a completude da presente classificação, mas é demonstrado que ela é mais completa do que estas encontradas na literatura.

As seções seguintes apresentam em detalhes o mapeamento da classificação proposta em relação a nove outras propostas apresentadas no Capítulo 3.

5.1 Mapeamento em Relação a DOUCET, Anne et al.

O mapeamento para o trabalho de Doucet e outros (DOUCET et al., 1997) pode ser considerado o mais importante dentre os mapeamentos aqui apresentados, pois este é o único trabalho encontrado na literatura que engloba ambos conceitos de tempo e versão. Esse mapeamento é apresentado na Figura 5.1. Nessa figura, a árvore superior (Figura 5.1a) representa a classificação de Doucet, enquanto as demais árvores (Figura 5.1b) representam partes da classificação proposta. Círculos rotulados mostram as ligações entre as classes equivalentes das duas classificações. Vale notar que esta mesma estruturação de figuras também será utilizada nos mapeamentos relacionados aos demais trabalhos.

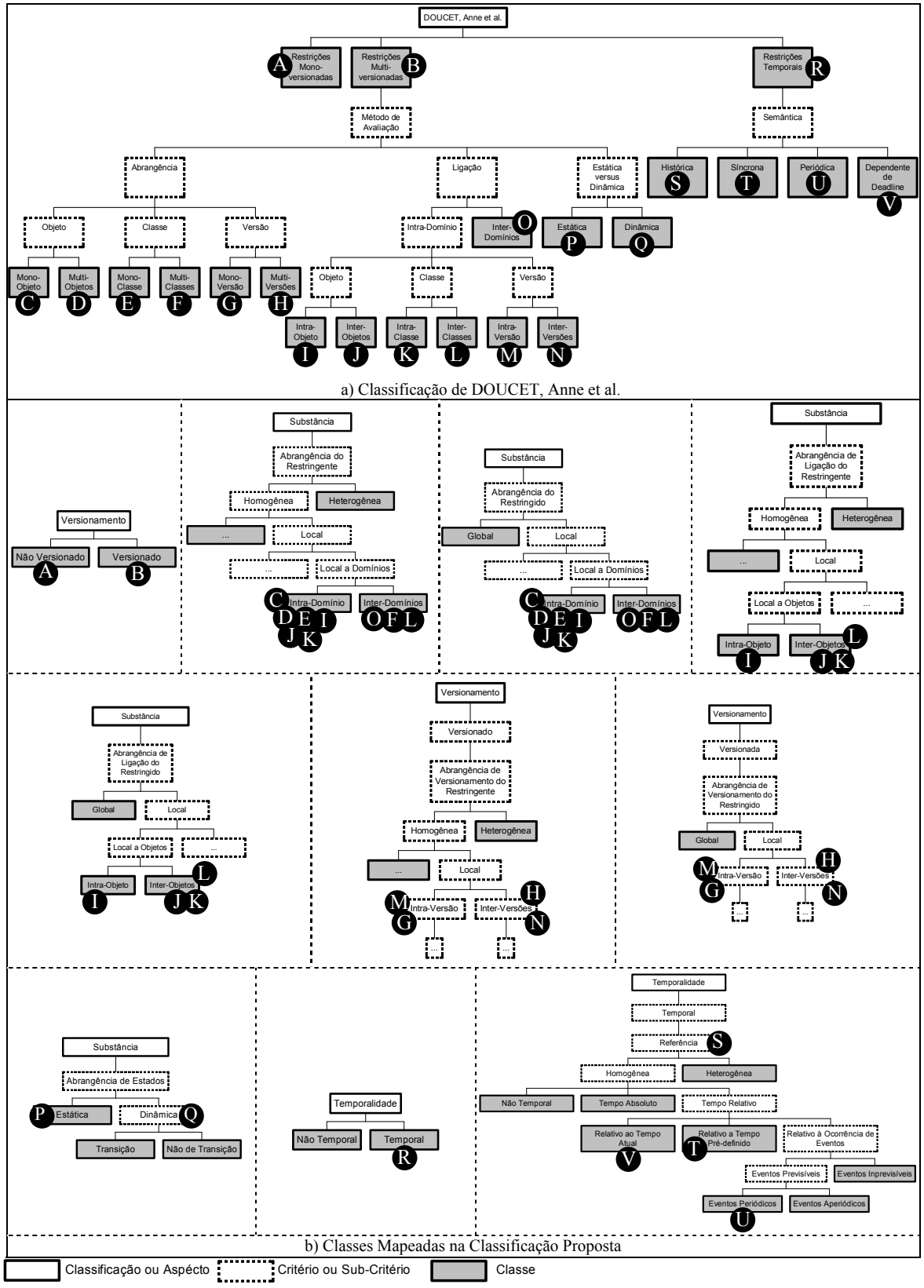


Figura 5.1: Mapeamento em relação a DOUCET, Anne et al.

Como pode ser visto, as restrições *monoversionadas*, *multiversionadas* e *temporais* de Doucet, são representadas respectivamente por restrições *não versionadas* e *versionadas*, do aspecto *versionamento*, e pelas restrições *temporais*, do aspecto

temporalidade. Isso ocorre pois, nessas classes, Doucet considera apenas a temporalidade e o versionamento dos dados envolvidos pelas restrições. A temporalidade e o versionamento das próprias restrições não são considerados.

Todas as classes dos critérios de *abrangência* (*scope*) e *ligação* (*binding*) no trabalho de Doucet são representadas pelos critérios e sub-critérios de *abrangência* e *abrangência de ligação* presentes nos aspectos de *substância* e *versionamento*. Nota-se ainda que a abrangência de temporalidade não é considerada.

Finalmente, as restrições *dinâmicas* e *estáticas* de Doucet são representadas pelas classes do critério *abrangência de estados* do aspecto *sustância*, enquanto seu aspecto de *semântica* é mapeado para classes correspondentes no sub-critério *referência* do aspecto *temporalidade*. Isso ocorre pois este aspecto considera referências de restrições relacionadas ao *tempo atual*, *tempo pré-definido* ou à *ocorrência de eventos*.

5.2 Mapeamento em Relação a RAM, D. Janaki et al.

Na classificação apresentada por Ram e outros (RAM et al., 1997), a principal característica levada em consideração é a abrangência de restrições de integridade, por eles chamada de *espaço de validação*.

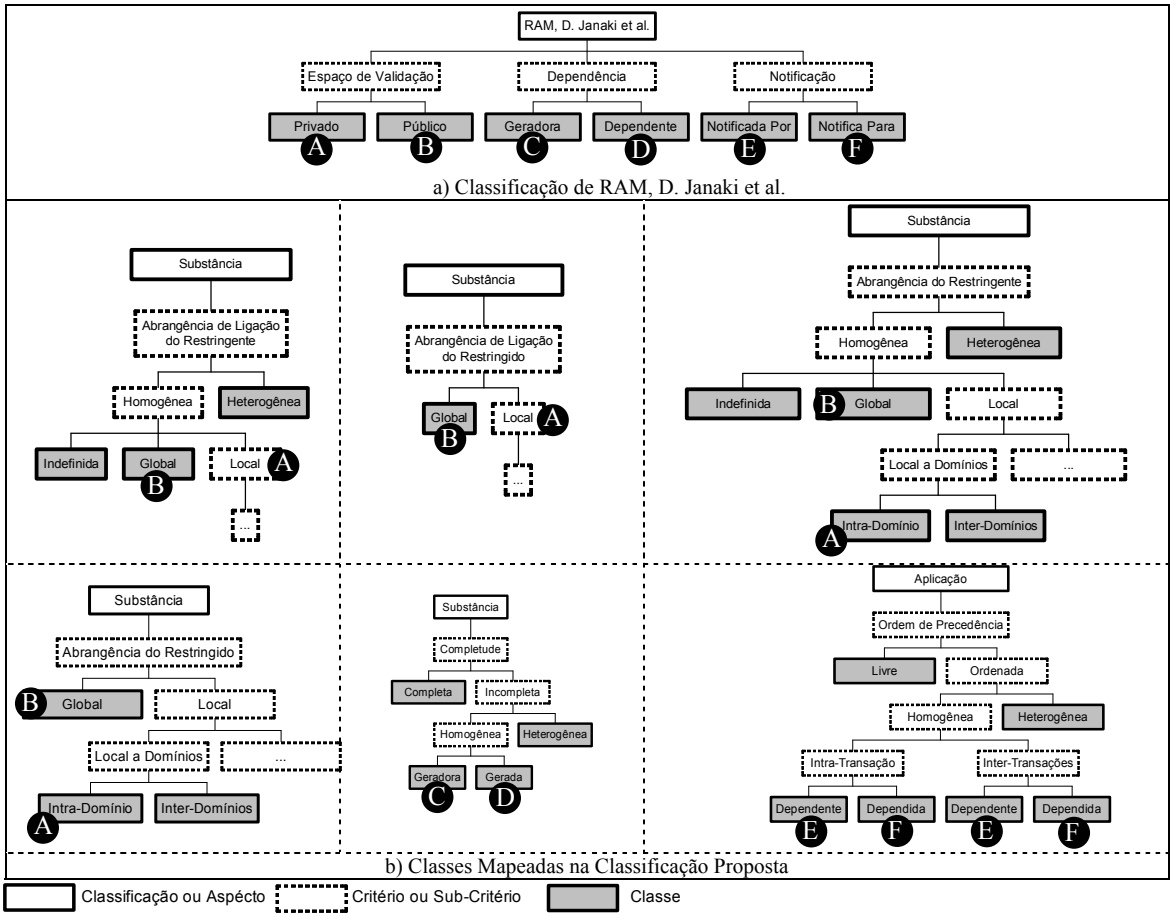


Figura 5.2: Mapeamento em relação a RAM, D. Janaki et al.

Ram classifica restrições *privadas* e *públicas*, de acordo com a abrangência de dados por elas envolvidos. Assim, como pode ser visto na Figura 5.2, essas classes são facilmente representadas na presente classificação através das classes existentes nos

critérios de *abrangência* e *abrangência de ligação* do aspecto *substância*. As abrangências de temporalidade e de versionamento não são consideradas.

Além disso, o critério *dependência* de Ram é representado totalmente pelo critério *completude* da classificação, onde restrições *incompletas* podem ser *geradoras* ou *geradas* de/por outras restrições. Finalmente, as classes do critério *notificação* de Ram são representadas pelas diferentes classes de restrições *ordenadas* no critério *ordem de precedência* da classificação proposta.

5.3 Mapeamento em Relação a DAYAL, Umeshwar et al.

Na classificação apresentada por Dayal e outros (DAYAL et al., 1988) são analisadas diversas características de restrições *temporais*, por eles chamadas de *timing constraints*. Características de versionamento não são consideradas.

É analisada a temporalidade dos dados envolvidos por uma restrição (conjuntos restringente e restringido). Entretanto, a temporalidade da restrição em si não é considerada. Dessa forma, nota-se claramente a equivalência entre as restrições *temporais* de Dayal e as restrições *temporais* do aspecto *temporalidade* da classificação proposta.

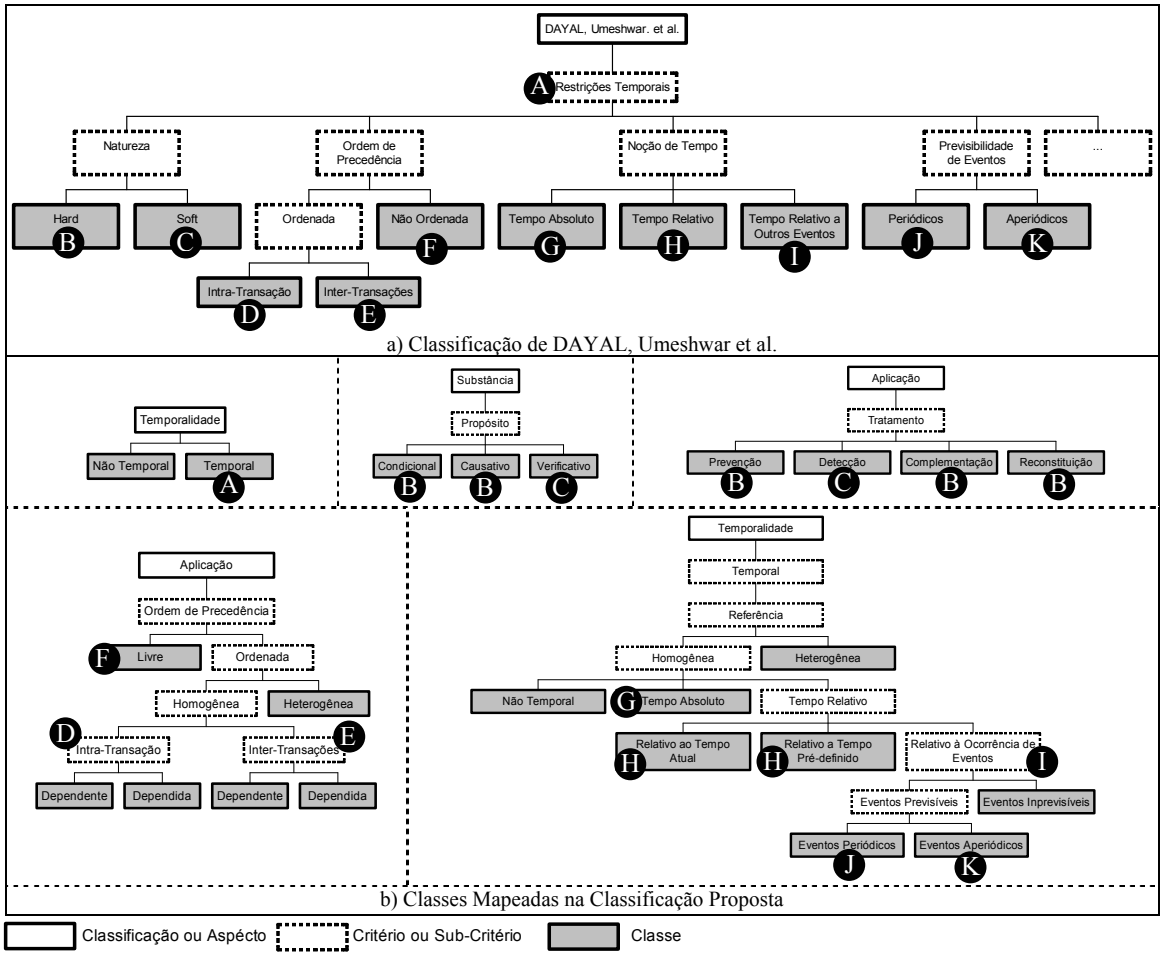


Figura 5.3: Mapeamento em relação a DAYAL, Umeshwar et al.

Assim como no presente trabalho, Dayal analisa restrições *temporais* por diversos sub-critérios. O primeiro deles diz respeito à *natureza* de restrições de integridade, classificando respectivamente restrições obrigatórias e desejáveis como *hard* ou *soft*.

Como pode ser visto na Figura 5.3, o segundo grupo de restrições é representado, na classificação proposta, por restrições com *propósito verificativo* e *tratamento* de *detecção*, de acordo com os aspectos *substância* e *aplicação*. As restrições classificadas como *hard* por Dayal são representadas por restrições com qualquer outro *propósito* ou *tratamento*.

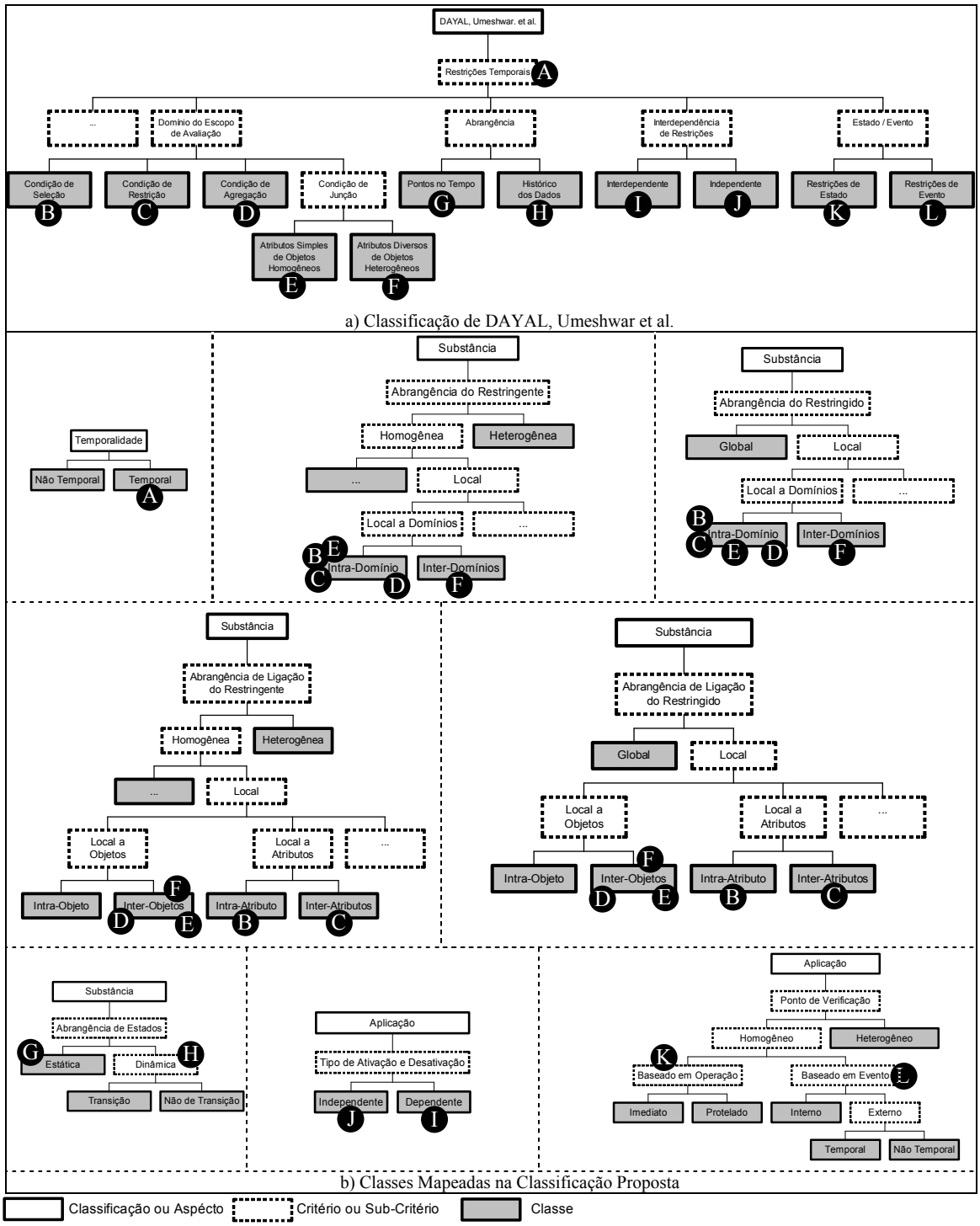


Figura 5.4: Mapeamento em relação a DAYAL, Umeshwar et al. (cont.)

Dayal também analisa a ordem de precedência de restrições *temporais*. Na presente classificação, este sub-critério é visivelmente representado pelo critério *ordem de*

precedência presente no aspecto *aplicação*. Finalmente, as classes definidas pela noção de tempo e pela previsibilidade de eventos (*precipitating events*) de Dayal são mapeadas para classes presentes no sub-critério *referência* do aspecto *temporalidade*, de acordo com a utilização de referências baseadas no tempo absoluto e tempos relativos ao tempo atual, a tempo pré-definido e à ocorrência de eventos.

O *domínio do escopo de avaliação* de restrições *temporais* também foi analisado por Dayal. Através da Figura 5.4, percebe-se que todas as classes definidas por este sub-critério são representadas por classes dos critérios de *abrangência* e *abrangência de ligação* do aspecto *substância*. Dayal ainda define a existência de restrições sobre *pontos no tempo* ou sobre o *histórico dos dados*, respectivamente representadas por restrições *estáticas* e *dinâmicas* no critério *abrangência de estados* da classificação proposta.

Finalmente, a *interdependência de restrições*, prevista por Dayal, é representada por classes do critério *tipo de ativação e desativação* do aspecto *aplicação*, enquanto as restrições de *estados* ou *eventos* são respectivamente mapeadas para restrições *baseadas em operação* e *baseadas em evento*, de acordo com o aspecto *aplicação*.

5.4 Mapeamento em Relação a CASTILHO, CASANOVA e FURTADO

Uma breve classificação de restrições de integridade é apresentada por Castilho, Casanova e Furtado em seu trabalho (CASTILHO; CASANOVA; FURTADO, 1982). São analisadas a *abrangência de estados* e a *temporalidade* de restrições de integridade.

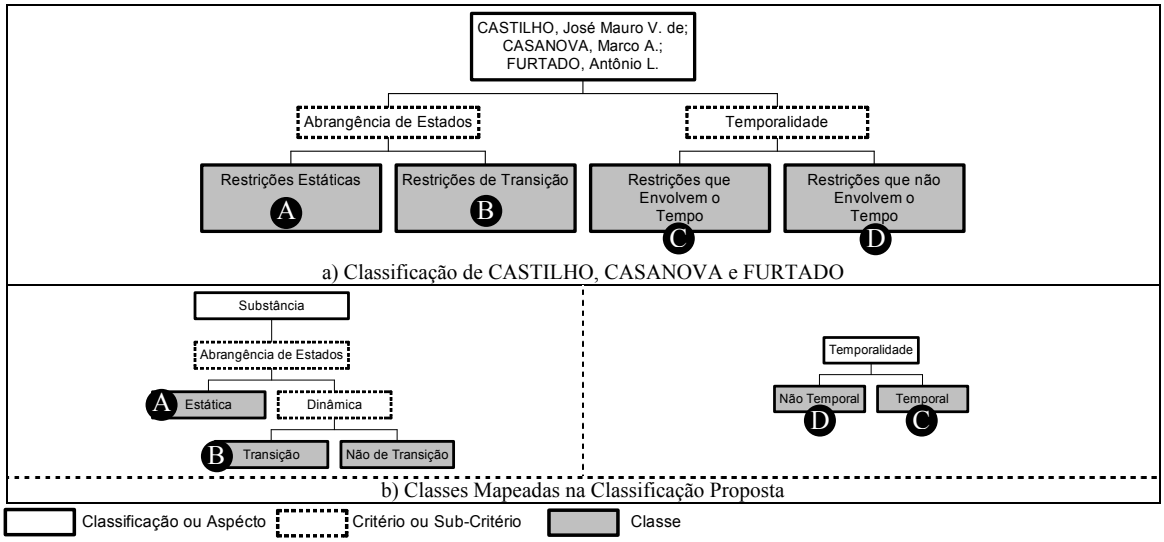


Figura 5.5: Mapeamento em relação a CASTILHO, CASANOVA e FURTADO

As *restrições estáticas* e de *transição* de Castilho são representadas na classificação proposta por restrições de mesmo nome, de acordo com o critério *abrangência de estados* do aspecto *substância*. Além disso, a *temporalidade* considerada por eles é facilmente mapeada para restrições *temporais* e *não temporais* do aspecto *temporalidade*, visto que estas consideram apenas a temporalidade dos dados envolvidos na restrição. A temporalidade das próprias restrições e características de versionamento não são consideradas.

Dessa forma, o mapeamento em relação ao trabalho de Castilho, Casanova e Furtado pode ser visto na Figura 5.5.

5.5 Mapeamento em Relação a BÖHLEN, Michael H.

No trabalho realizado por Böhlen (BÖHLEN, 1994), restrições de integridade são classificadas inicialmente como *temporais* e *não temporais* de acordo com a temporalidade dos dados por elas envolvidos. Sendo assim, nota-se claramente a equivalência entre essas duas classes definidas por Böhlen e as duas classes de mesmo nome presentes no primeiro nível do aspecto *temporalidade*. A temporalidade de restrições e suas características de versionamento não são consideradas.

Böhlen também classifica restrições *temporais* de acordo com diversos sub-critérios. Primeiramente, essas restrições são agrupadas de acordo com o tipo de tempo que utilizam, sendo então divididas em restrições de *tempo de validade*, *tempo de transação* e *bitemporais*. Na classificação proposta, uma divisão equivalente é feita através dos sub-critérios de *tipo de tempo* presentes no aspecto *temporalidade*. Além disso, Böhlen considera a abrangência de estados de restrições temporais, criando diversas classes que podem ser mapeadas para classes equivalentes do critério *abrangência de estados* presente no aspecto *substância*.

A Figura 5.6 ilustra de forma detalhada o mapeamento em relação à classificação criada por Böhlen.

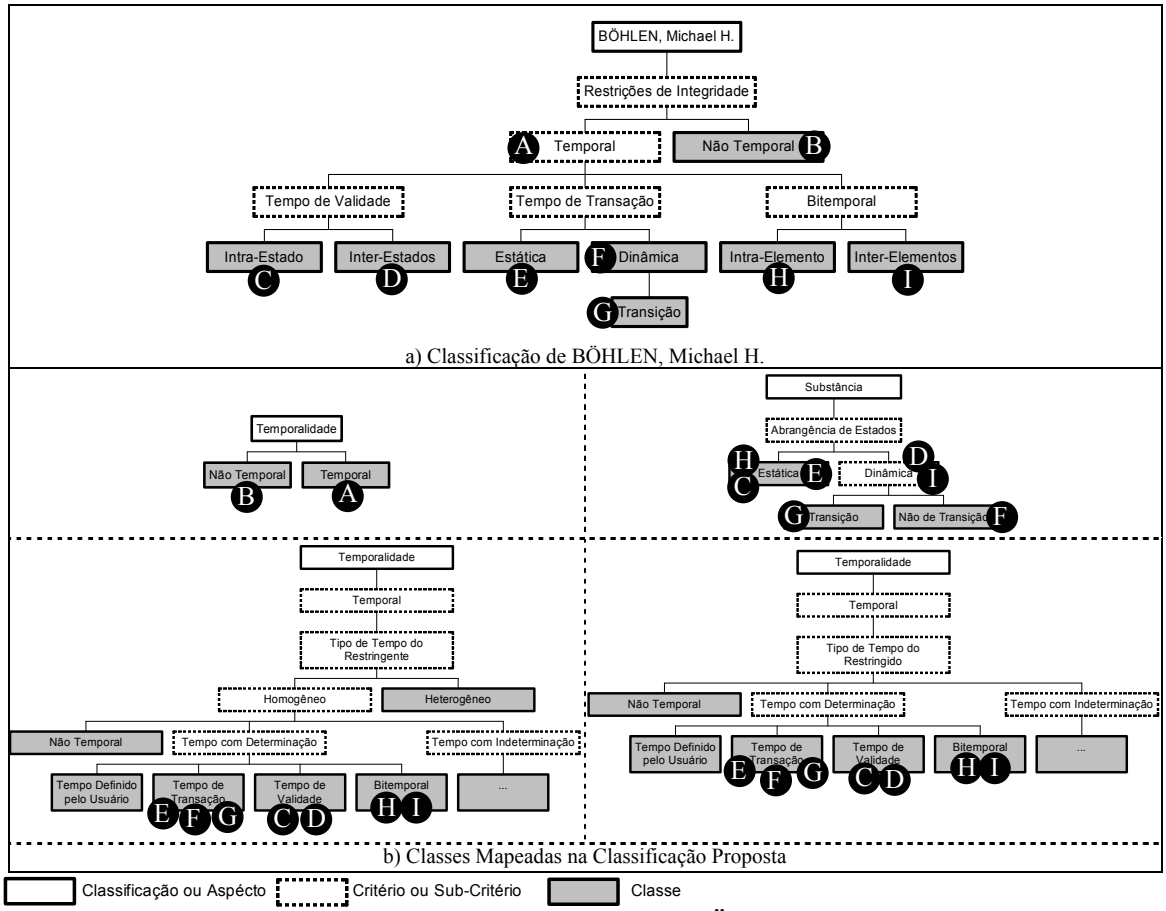


Figura 5.6: Mapeamento em relação a BÖHLEN, Michael H.

5.6 Mapeamento em Relação a ESCOFET, Carme Martín

Em seu trabalho (ESCOFET, 2001), Escofet analisa restrições de integridade *temporais* tomando como base os trabalhos de Böhlen (BÖHLEN, 1994) e Doucet

(DOUCET et al., 1997). Novamente, características de versionamento não são consideradas.

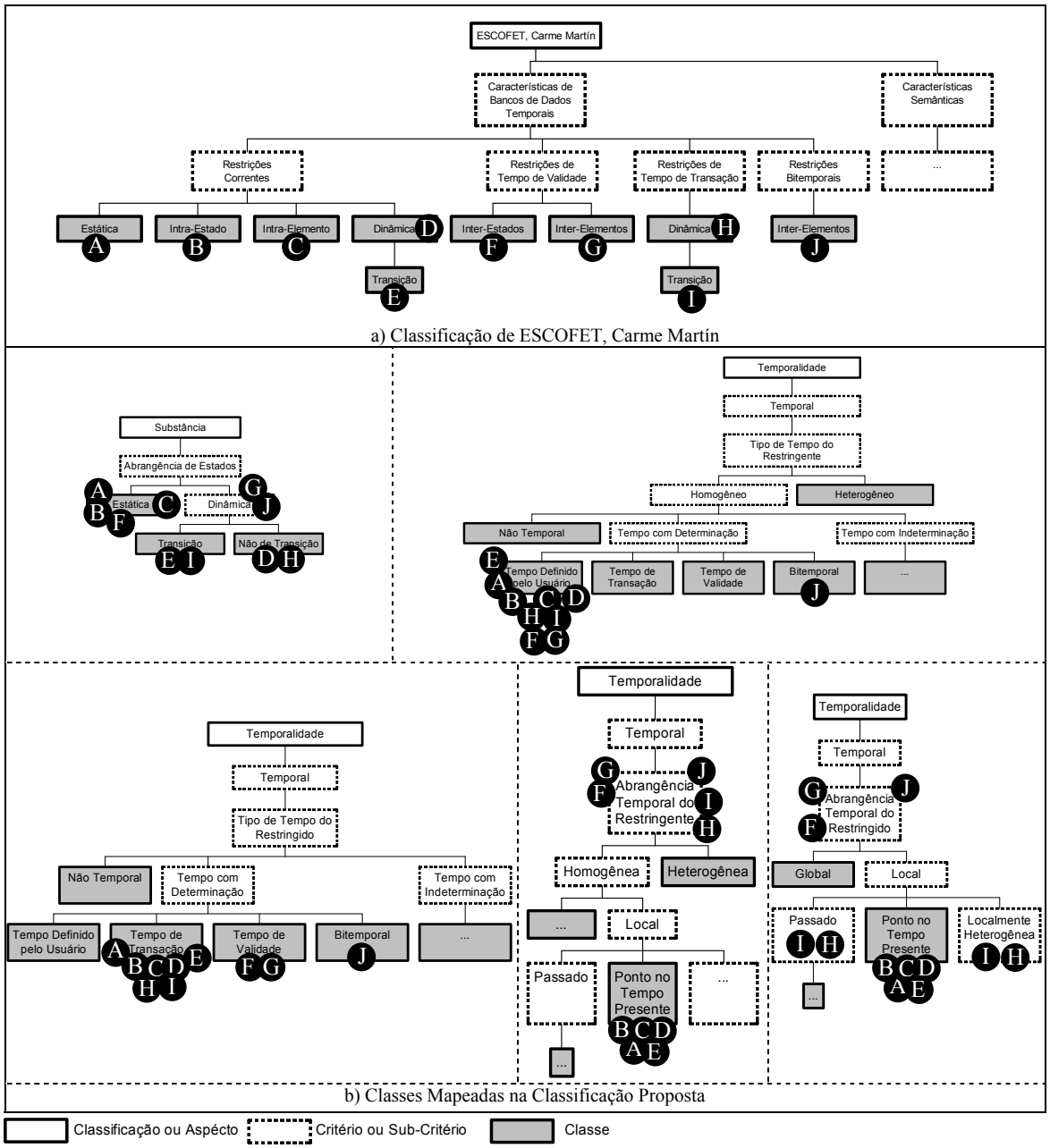


Figura 5.7: Mapeamento em relação a ESCOFET, Carme Martín

Considerando o trabalho de Böhlen, Escofet classifica restrições de acordo com o tipo de tempo que elas utilizam e com suas abrangências de estados. Entretanto, além dos tipos de tempo considerados por Böhlen, seu trabalho também considera restrições de integridade *baseadas no tempo corrente*. Quanto ao mapeamento, as classes provenientes dessa análise podem ser representadas na presente classificação através de classes do critério *abrangência de estados*, do aspecto *substância*, e dos sub-critérios de *tipo de tempo*, do aspecto *temporalidade*. Além disso, devido às classes definidas sobre restrições *baseadas no tempo corrente*, os sub-critérios de *abrangência temporal* também se fazem necessários no mapeamento. Apesar da boa classificação de tipo e abrangência temporal, a temporalidade das próprias restrições não é analisada. Através

da Figura 5.7, o mapeamento em relação à primeira parte da classificação de Escofet é apresentado de forma detalhada.

Com base no trabalho de Doucet, Escofet também classifica restrições temporais de acordo com suas características *semânticas* e com o tipo de tempo por elas utilizado. Suas restrições de *condição*, *síncronas*, *periódicas* e *dependentes de deadline* podem ser representadas na presente classificação através de classes do sub-critério *referência* presente no aspecto *temporalidade*, de acordo com a utilização de referências baseadas no *tempo atual*, em *tempo pré-definido* ou na *ocorrência de eventos*. Além disso, o tipo de tempo considerado por Escofet é facilmente representado pelos sub-critérios de *tipo de tempo* do mesmo aspecto, com base no *tempo com determinação*. A Figura 5.8 ilustra graficamente o mapeamento em relação à segunda parte da classificação de Escofet.

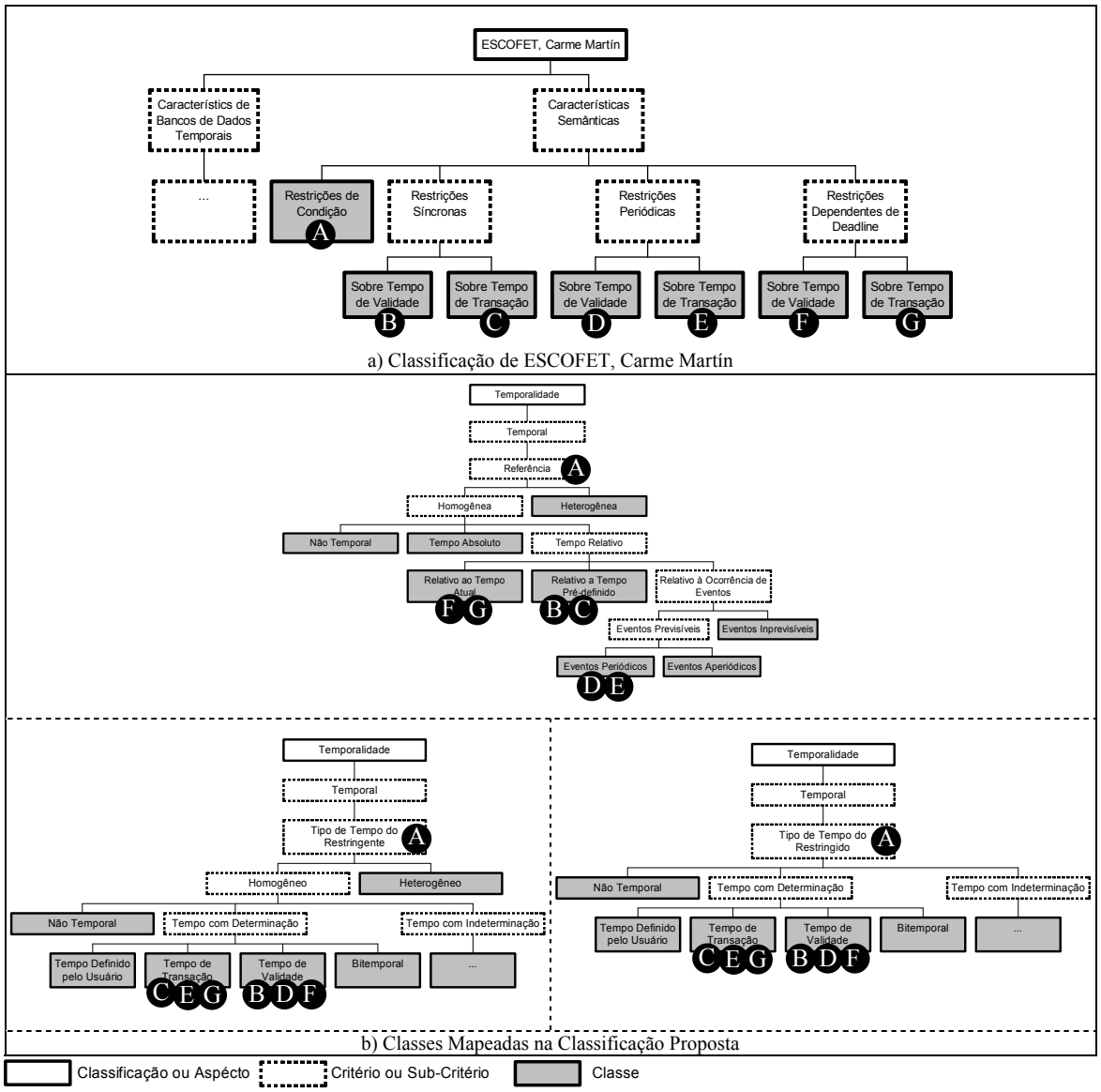


Figura 5.8: Mapeamento em relação a ESCOFET, Carme Martín (cont.)

5.7 Mapeamento em Relação a CHOMICKI e TOMAN

Em seus trabalhos (CHOMICKI, 1992; CHOMICKI, 1995; CHOMICKI; TOMAN, 1995), Chomicki e Toman classificam restrições de integridade para bancos de dados temporais. Em um primeiro nível são classificadas restrições *temporais* e *estáticas*, sendo que não é considerada a temporalidade da própria restrição e nem mesmo a possibilidade de restrições temporais que utilizem apenas um estado da base em sua validação. Dessa forma, como pode ser visto na Figura 5.9, são consideras equivalentes as restrições *temporais* de Chomicki e Toman e as restrições *dinâmicas* representadas pelo critério *abrangência de estados* do aspecto *substância*. Também são consideradas equivalentes as restrições *estáticas* de ambas classificações, por abrangerem um único estado da base de dados durante seu tempo de vida.

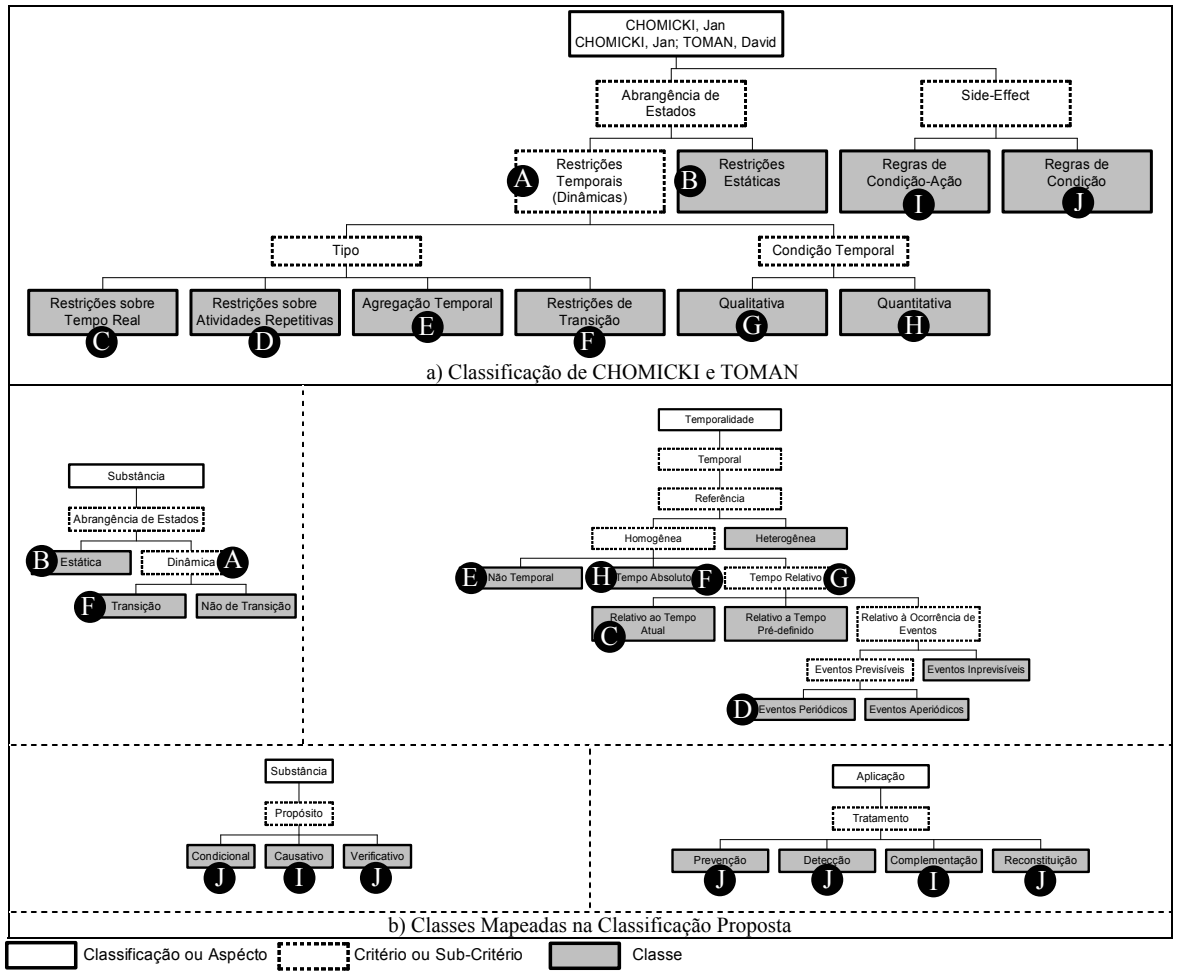


Figura 5.9: Mapeamento em relação a CHOMICKI e TOMAN

Além disso, esse trabalho classifica restrições temporais (*dinâmicas*) de acordo com seus *tipos* e com suas *condições temporais*. Quanto ao mapeamento, o tipo *transição* é representado pela classe de restrições de *transição* do critério *abrangência de estados*. As demais classes são mapeadas para classes presentes no sub-critério *referência* do aspecto *temporalidade*, de acordo com a utilização de referências *não temporais*, de *tempo absoluto*, de tempo *relativo ao tempo atual* e de tempo relativo à *ocorrência de eventos*. Por fim, Chomicki e Toman consideram a existência de restrições que possuem ou não ações de violação (*restrições de condição-ação*, *condição*), as quais são

facilmente representadas através dos critérios *propósito* e *tratamento* da classificação proposta.

5.8 Mapeamento em Relação a MEDEIROS, JOMIER e CELLARY

Medeiros, Jomier e Cellary (1992) analisam restrições de integridade no contexto de bancos de dados de versões sob diversos aspectos. Características temporais não são consideradas.

Assim como pode ser visto na Figura 5.10, seu primeiro critério é chamado de *dependência* e, segundo ele, restrições podem ser dependentes da *aplicação*, do *controle do versionamento* ou do *modelo*. Quanto à classificação proposta, classes equivalentes são encontradas no aspecto *origem* e no critério *veículo* do aspecto *aplicação*. Medeiros também analisa a abrangência de estados de restrições e considera as restrições *temporais* como um sub-conjunto das restrições *dinâmicas*. Dessa forma, as classes resultantes são mapeadas para classes do critério *abrangência de estados* e do aspecto *temporalidade* na classificação proposta.

Este trabalho demonstra a possibilidade de existência de restrições definidas sobre o *conteúdo* dos dados, o *comportamento* e a *estrutura* do esquema. O mapeamento para essas classes é definido com base nos critérios *aspecto restringente* e *aspecto restringido*, presentes na *substância*, através de seus aspectos *comportamento*, *estrutura* e *composição interna*.

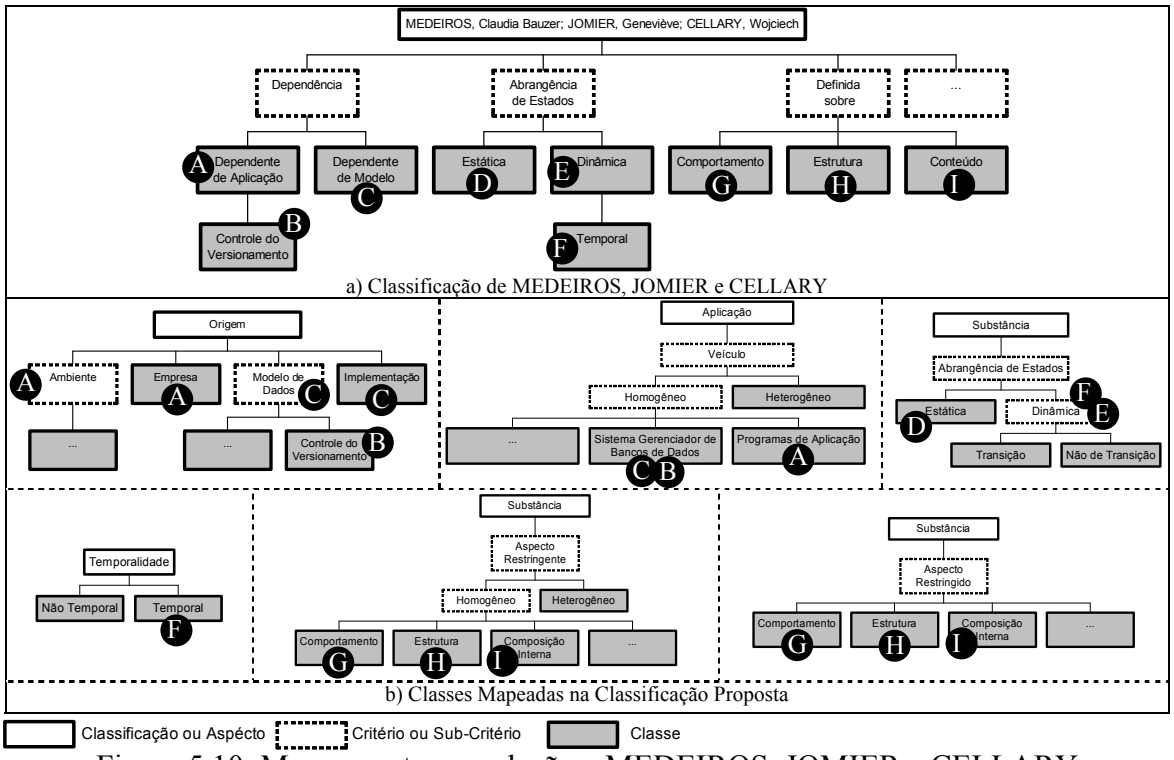


Figura 5.10: Mapeamento em relação a MEDEIROS, JOMIER e CELLARY

Como pode ser visto na Figura 5.11, a *abrangência de versionamento* de restrições de integridade também é considerada por Medeiros e possui classes equivalentes às classes dos critérios de *abrangência de versionamento* presentes no aspecto *versionamento*. Por fim, também são consideradas possíveis versões de restrições de integridade. Quanto ao mapeamento, considera-se a equivalência das restrições *versionadas* e *não versionadas* de Medeiros em relação às restrições de mesmo nome

classificadas no critério *versionamento da restrição*, presente no aspecto *especificação* da classificação proposta.

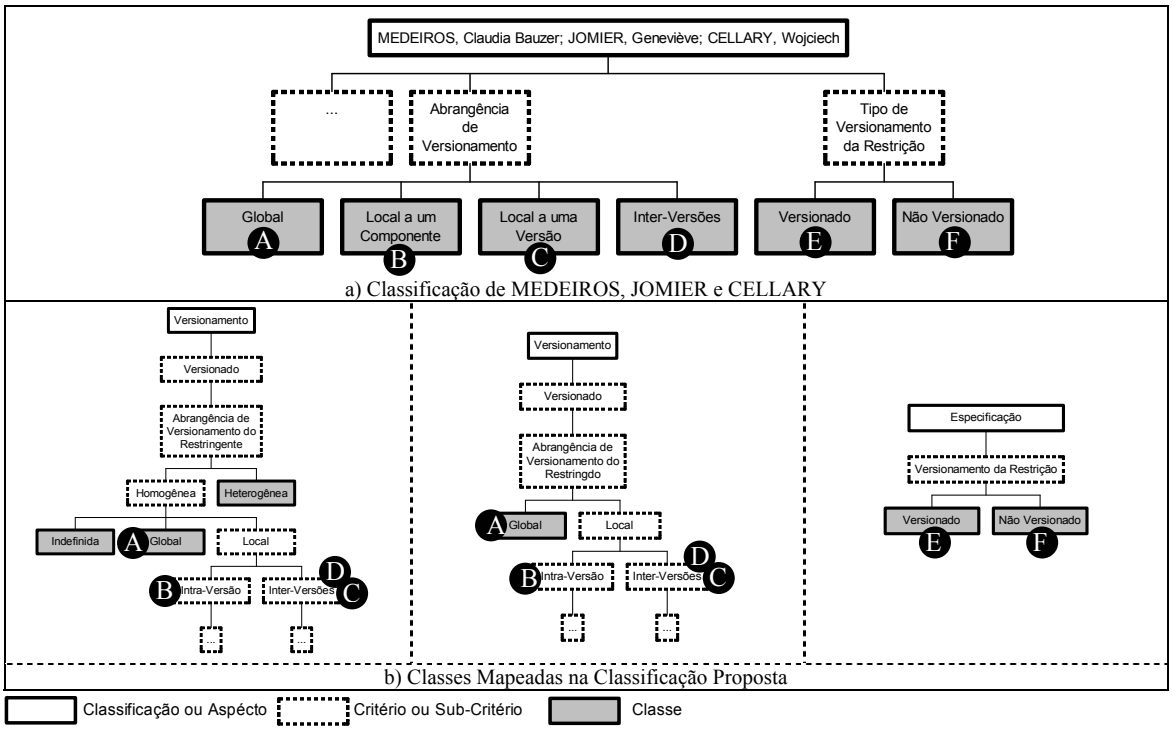


Figura 5.11: Mapeamento em relação a MEDEIROS, JOMIER e CELLARY (cont.)

5.9 Mapeamento em Relação a PLEXOUSAKIS e COWLEY

Em seus trabalhos (PLEXOUSAKIS, 1993; PLEXOUSAKIS, 1995; COWLEY; PLEXOUSAKIS, 2000), Plexousakis e Cowley tratam a manutenção de integridade de bases de conhecimento temporais e dedutivas. A possibilidade de utilização de indeterminação temporal também é considerada. Apesar da boa classificação, novamente não são consideradas características de versionamento e a temporalidade das próprias restrições. A Figura 5.12 ilustra graficamente o mapeamento em relação a este trabalho.

Em sua classificação são consideradas características referentes à temporalidade dos dados envolvidos por restrições de integridade. Quanto ao mapeamento, fica clara a equivalência entre as *restrições temporais* deste trabalho relacionado e as *restrições temporais* do aspecto *temporalidade* presente na classificação proposta.

Assim como no presente trabalho, Plexousakis e Cowley analisam restrições temporais de acordo com diversos aspectos. Primeiramente é analisada a abrangência de restrições em relação aos estados necessários para sua validação. Dessa forma, dentre as restrições temporais, nota-se a equivalência entre as classes *estáticas*, *dinâmicas* e de *transição* de ambos trabalhos. Além disso, este trabalho relacionado define a possibilidade de *restrições epistêmicas* que abrangem dois ou mais estados da base, considerando o tempo de transação e o tempo de validade. Quanto à presente classificação, essas restrições são representadas por restrições de *abrangência de estados dinâmica* com restringentes ou restringidos de *tipo de tempo bitemporal*.

Plexousakis e Cowley também analisam a possibilidade de *dependência de restrições sobre regras dedutivas* de uma base de conhecimento. As classes definidas por eles são equivalentes a classes definidas no critério *dependência de regras dedutivas*

presentes no aspecto *aplicação* da presente classificação. Além disso, seus trabalhos consideram a possibilidade de restrições *potenciais* e *não potenciais*, de acordo com a utilização de indeterminação temporal. Essas restrições são representadas por restrições com *tipo de tempo do restringido* ou *do restringente* baseado em *tempo com determinação* ou *tempo com indeterminação*.

Por fim, Plexousakis e Cowley consideram a possibilidade de restrições *estruturais* e *semânticas*. Na classificação proposta, restrições estruturais são representadas por restrições com *aspecto e tipo do restringido* ou *restringente* relacionado à *estrutura* da base de dados e ao *esquema*. Restrições semânticas são representadas por restrições com qualquer outro aspecto e tipo.

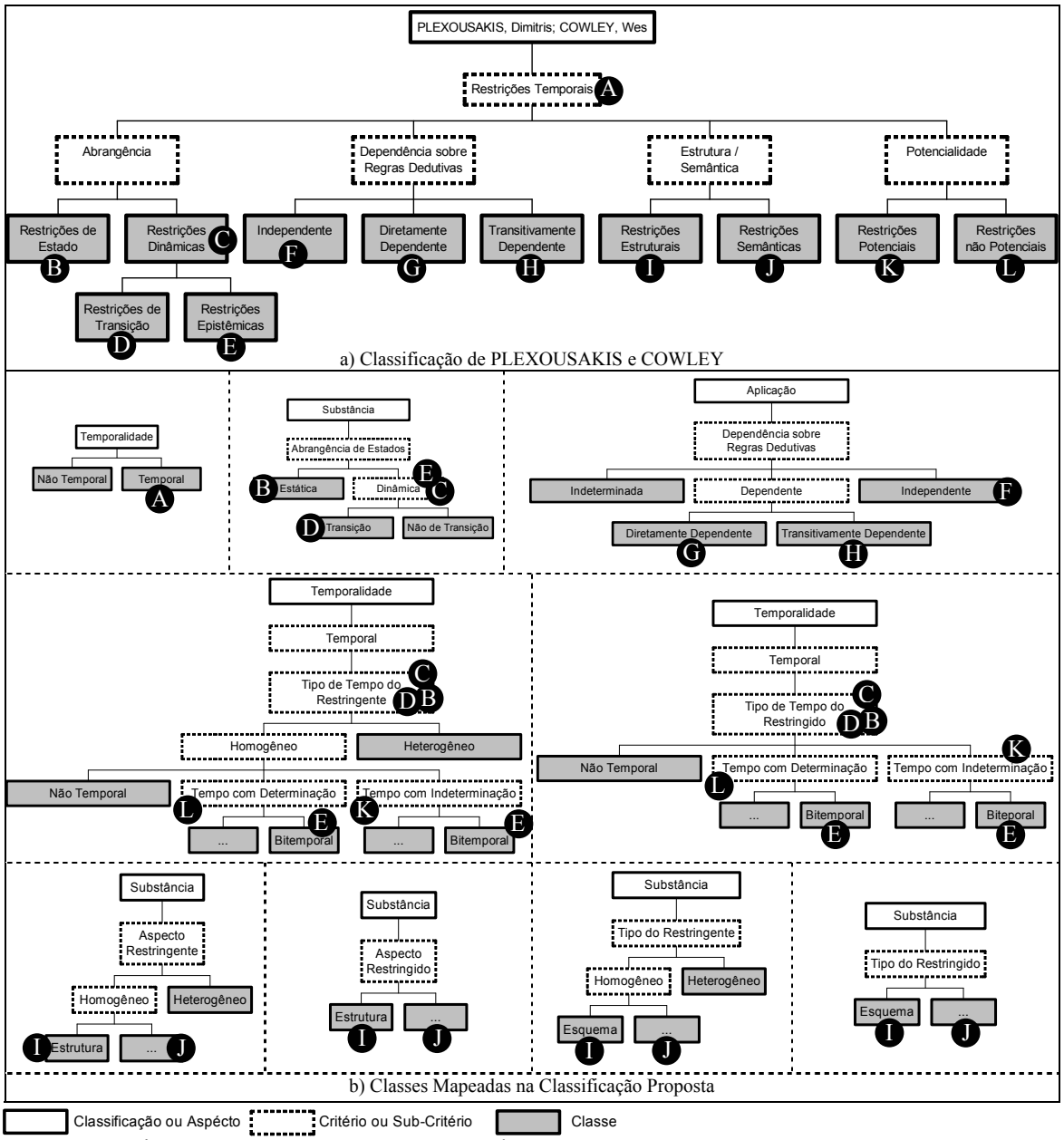


Figura 5.12: Mapeamento em relação a PLEXOUSAKIS e COWLEY

5.10 Considerações Finais

Este capítulo apresentou uma análise sobre a abrangência da classificação de restrições de integridade definida no capítulo anterior. Considerando a impossibilidade de prova formal da completude da classificação proposta, a alternativa escolhida para analisar sua abrangência foi comparar a presente classificação com outras encontradas na literatura. Essa comparação foi feita através de um mapeamento entre cada uma das classes definidas por outros trabalhos e classes equivalentes presentes na classificação proposta.

Um resumo da análise de abrangência da classificação proposta é apresentado através da Tabela 5.1. Nessa tabela, as classificações propostas nos trabalhos apresentados no Capítulo 3, nas quais se baseiam seus métodos de manutenção de integridade, são comparadas com a classificação proposta no Capítulo 4. Essa comparação indica os aspectos de restrições considerados e não considerados por cada classificação. Através dessa análise, percebe-se que os seguintes aspectos não são considerados por nenhum dos trabalhos relacionados: formação da condição restritiva, forma de declaração, tipo de acionamento e de inspeção, granularidade temporal, tipo de tempo e abrangência temporal de restrições. Percebe-se também que são tratadas em cada trabalho relacionado somente características específicas de restrições, pois nenhuma de suas classificações é genérica o suficiente para considerar a maioria dos aspectos inerentes a restrições.

O resultado desta experiência demonstra que, além de todas as características de restrições consideradas nesses trabalhos, a presente classificação considera características não consideradas por eles. Apesar de não estar provada formalmente sua completude, a classificação proposta é mais completa do que qualquer uma das outras apresentadas. Desta forma, acredita-se nas vantagens de sua utilização como base do desenvolvimento de trabalhos de especificação e verificação de restrições de integridade.

O capítulo seguinte apresentará uma linguagem de especificação de restrições de integridade sobre o Modelo Temporal de Versões (TVM). Logo após, com base na classificação apresentada, o poder de expressão da linguagem proposta será analisado a fim de definir quais classes de restrições esta linguagem pode especificar.

Tabela 5.1: Comparação entre as classificações de restrições

Aspectos e Critérios da Classificação Proposta / Trabalhos Relacionados	DOUCET, Anne et al.	RAM, D. Janaki et al.	DAYAL, Umeshwar et al.	CASTILHO, CASANOVA e FURTADO	BÖHLEN, Michael H.
Origem	NC	NC	NC	NC	NC
Abrangência de Estados	C	NC	C	C	C
Tipo e Formação dos Dados Envolvidos	NC	NC	NC	NC	NC
Formação da Condição Restritiva	NC	NC	NC	NC	NC
Aspecto Restringente e Restringido	NC	NC	NC	NC	NC
Propósito	NC	NC	C	NC	NC
Abrangência dos Dados Envolvidos e das Ligações entre eles	C	C	C	NC	NC
Completude	NC	C	NC	NC	NC
Forma de Declaração	NC	NC	NC	NC	NC
Dependência sobre Regras Dedutivas	NC	NC	NC	NC	NC
Ordem de Precedência	NC	C	C	NC	NC
Ativação e Desativação	NC	NC	C	NC	NC
Tipo de Acionamento	NC	NC	NC	NC	NC
Tratamento	NC	NC	C	NC	NC
Veículo	NC	NC	NC	NC	NC
Tipo de Inspeção	NC	NC	NC	NC	NC
Tipos de Ponto de Verificação	NC	NC	C	NC	NC
Referência entre os Dados Restringentes e Restringidos	C	NC	C	NC	NC
Granularidade Temporal	NC	NC	NC	NC	NC
Tipo de Tempo da Restrição	NC	NC	NC	NC	NC
Tipo de Tempo dos Dados Envolvidos	C	NC	C	C	C
Abrangência Temporal da Restrição	NC	NC	NC	NC	NC
Abrangência Temporal dos Dados Envolvidos	NC	NC	NC	NC	NC
Tipo de Versionamento da Restrição	C	NC	NC	NC	NC
Tipo de Versionamento dos Dados Envolvidos	C	NC	NC	NC	NC
Abrangência de Versionamento dos Dados Envolvidos	C	NC	NC	NC	NC

C – Considerado NC – Não Considerado

Tabela 5.1: Comparação entre as classificações de restrições (cont.)

Aspectos e Critérios da Classificação Proposta / Trabalhos Relacionados	ESCOFET, Carme Martín	CHOMICKI e TOMAN	MEDEIROS, JOMIER e CELLARY	PLEXOUSAKIS e COWLEY
Origem	NC	NC	C	NC
Abrangência de Estados	C	C	C	C
Tipo e Formação dos Dados Envolvidos	NC	NC	NC	C
Formação da Condição Restritiva	NC	NC	NC	NC
Aspecto Restringente e Restringido	NC	NC	C	C
Propósito	NC	C	NC	NC
Abrangência dos Dados Envolvidos e das Ligações entre eles	NC	NC	NC	NC
Completude	NC	NC	NC	NC
Forma de Declaração	NC	NC	NC	NC
Dependência sobre Regras Dedutivas	NC	NC	NC	C
Ordem de Precedência	NC	NC	NC	NC
Ativação e Desativação	NC	NC	NC	NC
Tipo de Acionamento	NC	NC	NC	NC
Tratamento	NC	C	NC	NC
Veículo	NC	NC	C	NC
Tipo de Inspeção	NC	NC	NC	NC
Tipos de Ponto de Verificação	NC	NC	NC	NC
Referência entre os Dados Restringentes e Restringidos	C	C	NC	NC
Granularidade Temporal	NC	NC	NC	NC
Tipo de Tempo da Restrição	NC	NC	NC	NC
Tipo de Tempo dos Dados Envolvidos	C	NC	C	C
Abrangência Temporal da Restrição	NC	NC	NC	NC
Abrangência Temporal dos Dados Envolvidos	C	NC	NC	NC
Tipo de Versionamento da Restrição	NC	NC	C	NC
Tipo de Versionamento dos Dados Envolvidos	NC	NC	C	NC
Abrangência de Versionamento dos Dados Envolvidos	NC	NC	C	NC

C – Considerado NC – Não Considerado

6 LINGUAGEM TVCL

Depois de classificadas as restrições de integridade, este capítulo apresenta a linguagem TVCL (*Temporal Versioned Constraint Language*), que permite a especificação de restrições de integridade sobre o modelo TVM. O intuito dessa extensão é permitir ao usuário explicitar de forma declarativa restrições de integridade que devam ser respeitadas pelos dados de uma base temporal de versões. Dessa forma, em conjunto com métodos para a manutenção de restrições, aplicações que utilizam bases sobre o TVM não mais terão a obrigação de tratar a integridade de dados.

As vantagens da manutenção de restrições inerente ao SGBD sobre a manutenção efetuada por programas de aplicação são claras: o desenvolvedor de aplicação pode se concentrar em *"quais restrições devem ser respeitadas"* ao invés de *"como elas deverão ser mantidas"*. Isso acarreta em programas de aplicação muito mais compactos e de fácil entendimento. Além disso, os programas de aplicação e a especificação de restrições formam módulos independentes, o que permite a construção de aplicações modulares, onde um ou mais módulos definem suas restrições de integridade (CHOMICKI; TOMAN, 1995).

6.1 Conceitos Introdutórios

Conforme visto no Capítulo 4, Santos (1980) define cinco componentes presentes em uma especificação completa de restrições de integridade: *restringente*, *restringido*, *condição restritiva*, *pontos de verificação* e *ações de violação*. A manutenção da integridade dos dados, em relação a uma restrição, é efetuada através da verificação da validade de sua condição restritiva em seus pontos de verificação. A Figura 6.1 ilustra graficamente esses componentes e a interação existente entre eles.

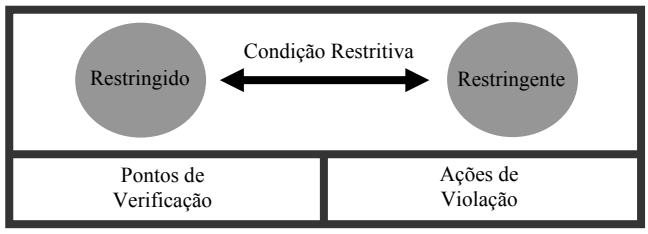


Figura 6.1: Componentes da especificação de restrições

Segundo Böhlen (1994), uma técnica simplista para a validação de uma restrição de integridade consiste em negá-la e submetê-la na forma de uma consulta ao banco de

dados. Caso a consulta retorne um resultado vazio, a restrição é satisfeita e, caso contrário, a restrição é violada. Com base nessa visão, a consulta é executada sempre que for alcançado um dos pontos de verificação da restrição. A Figura 6.2 ilustra como seria a aplicação dessa técnica sobre restrições de integridade em uma base de dados temporal de versões, considerando também suas possíveis versões e períodos de validade.

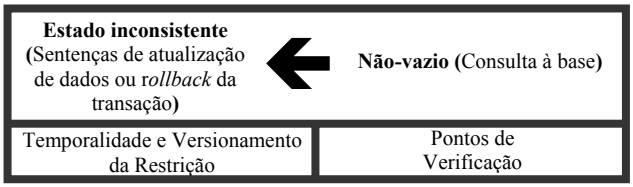


Figura 6.2: Técnica simplista de verificação

Na prática, a utilização desse método é claramente inviável devido a problemas de desempenho. Além disso, é útil a existência de mecanismos de verificação de restrições que não apenas decidam se um banco de dados é consistente ou não, mas também revelem as fontes e causas de inconsistências. (BÖHLEN, 1994).

Apesar disso, essa visão de validação de restrições de integridade pode ser utilizada com o intuito de permitir ao usuário especificar restrições sobre suas bases de dados. Com base nessa idéia, a especificação de uma restrição de integridade é feita de forma intuitiva pelo usuário, visto que ele define apenas os dados indesejáveis em sua base de dados (especificação declarativa) ao invés de definir formas de manutenção de integridade (especificação procedural). Vale notar que no contexto de linguagens de consulta a bases de dados relacionais, essa vantagem é conhecida e discutida há um longo tempo (CHOMICKI, 1995).

A linguagem de especificação de restrições TVCL é definida através da aplicação dessa idéia. Essa linguagem utiliza como base a linguagem de consulta (TVQL) do modelo TVM e parte de uma linguagem de especificação, versionamento e modificação de esquemas (linguagem de atualização de objetos) chamada TVL/SE (GALANTE; EDELWEISS; SANTOS, 2003), definida sob um modelo de versionamento de esquemas. A especificação de uma restrição é composta pelos seguintes componentes:

- consulta TVQL buscando dados inconsistentes;
- possível lista de operações de atualização de objetos TVL/SE;
- conjunto de pontos de verificação;
- controles sobre a temporalidade e o versionamento da restrição;
- controles sobre a ordem de precedência de verificação da restrição.

Em relação aos componentes definidos por Santos, os elementos referenciados na consulta TVQL correspondem aos integrantes dos conjuntos restringente e restringido de restrições de integridade. A condição de seleção utilizada nessa consulta corresponde à negação da condição restritiva da restrição. A lista de operações TVL/SE na especificação corresponde às possíveis ações de violação de uma restrição. Por fim, os pontos de verificação são apresentados explicitamente na especificação.

Novos integrantes devem ser incorporados à especificação da linguagem a fim de controlar as possíveis características de temporalidade e de versionamento das restrições em si. Esses devem especificar o tipo de versionamento e os períodos de validade da própria restrição. Novos integrantes também se fazem necessários a fim de controlar a ordem de precedência de validação de uma restrição em relação a outra(s).

É importante deixar claro que a linguagem proposta tem apenas o intuito de permitir ao usuário do TVM especificar restrições de integridade que até o momento só podem ser especificadas em linguagem coloquial e mantidas através de programas de aplicação. A manutenção das restrições especificadas por essa linguagem não é tratada nesse trabalho.

6.2 Definição da Linguagem

A especificação de restrições de integridade na linguagem TVCL é possível através de sentenças cuja estrutura obedece as regras definidas na BNF (*Backus Naur Form*) da linguagem. A versão simplificada dessa BNF é apresentada a seguir[1].

```
<constraint> ::= ( create constraint <constraintName> <constraintMiddle>|
   create TV constraint <constraintName> <constraintMiddle>
   [ <validity> ] [ <versioningScope> ] )
<constraintMiddle> ::= when  "{" <verifyingPoints> "}"
   if exists "(" <TVQLQuery> ")"
   then ( rollback | alert |
        "(" <TVLSEStatement> { "," <TVLSEStatement> }* ")" )
   [ follows "{" <constraintName> { "," <constraintName> }* "}" ]
   [ precedes "{" <constraintName> { "," <constraintName> }* "}" ]
<validity> ::=  validity <timeInterval>
<versioningScope> ::= versioning scope
   ( allVersions | "{" <versionState> {"," <versionState>}* "}" )
```

Através da cláusula *create constraint*, são definidos o nome da restrição e o tipo de sua temporalidade e versionamento. O conjunto de pontos de verificação de uma restrição é definido na cláusula *when*.

Também é determinada uma consulta TVQL para a restrição (cláusula *if exists*). Sua formulação deve definir uma busca sobre dados que, caso existam, violarão a restrição em questão. Através da cláusula *then*, é permitido ao usuário especificar as providências que devem ser tomadas caso dados inconsistentes sejam encontrados pela consulta. Essas providências podem ser: (i) a reconstituição da base de dados a um estado não influenciado pela transação que violou a restrição (*rollback*); (ii) a execução de uma lista de operações de reparo (operações da linguagem de atualização de objetos TVL/SE), ou (iii) o simples disparo de um alerta informando ao usuário a violação da restrição (*alert*). Vale notar que a primeira opção é válida apenas para restrições cujo conjunto de pontos de verificação é constituído apenas por pontos baseados em operação.

Através da cláusula opcional *validity*, possível apenas em restrições temporais de versões, é permitida a especificação de um intervalo temporal referente ao seu período de validade sobre a base de dados. Além disso, a cláusula *versioning scope* permite que a restrição restrinja versões de objetos em determinados estados, de acordo com seu grau de evolução, além de versões correntes e configuradas.

Por fim, as cláusulas *follows* e *precedes* permitem a definição de uma ordem de precedência na verificação de restrições sobre uma mesma base de dados. Essas cláusulas definem, respectivamente, as restrições que devem ser verificadas antes da verificação da restrição em questão e as que devem ser verificadas após sua verificação.

Vale notar que os itens especificados entre chaves (cláusulas *when*, *follows*, *precedes* e *versioning scope*) representam conjuntos matemáticos não vazios. Além disso, as sentenças da linguagem de atualização de objetos (<*TVLSEStatement*>)

[1] A BNF completa da linguagem encontra-se no Apêndice do trabalho.

apresentadas entre parênteses na cláusula *then* representam uma lista com uma ou mais sentenças a serem executadas na ordem definida na especificação.

As seções seguintes apresentam de forma detalhada as principais características da linguagem.

6.2.1 Controles de Temporalidade e de Versionamento

A cláusula *create constraint* da linguagem TVCL permite a especificação de restrições tradicionais ou restrições temporais de versões. A criação de restrições temporais de versões é definida através da utilização da palavra reservada *TV* no momento de sua especificação. No caso da omissão dessa palavra, restrições tradicionais são definidas. Os inícios de especificações a seguir demonstram respectivamente a criação de restrições tradicionais e de restrições temporais de versões.

```
create constraint Restricao ...
create TV constraint Restricao ...
```

Restrições tradicionais não possuem nenhum controle sobre sua própria temporalidade e versionamento. Diferentemente, restrições temporais de versões possuem implicitamente tempo de validade e tempo de transação, permitindo assim a armazenagem de seus históricos de alterações. Essas restrições podem ainda possuir validade relacionada a parte do tempo de vida da base de dados ou todo o tempo de vida. Além disso, versões deste tipo de restrição podem ser definidas a fim de restringir de forma distinta versões de objetos com diferentes níveis de desenvolvimento em um mesmo período de tempo.

O modelo TVM define que, de acordo com seu grau de desenvolvimento, versões de objetos podem estar nos seguintes estados: *estável*, *em trabalho*, *consolidado* ou *desativado*. Visto isso, versões de uma restrição podem restringir de forma distinta versões de objetos em diferentes estados. Os estados restringidos por uma versão de restrição de integridade são listados através da cláusula opcional *versioning scope* da linguagem TVCL, através das palavras reservadas *stableVersions*, *workingVersions*, *consolidatedVersions* e *deactivatedVersions* . Respectivamente, a fim de considerar apenas versões configuradas ou correntes de objetos, as palavras reservadas *configuredVersions* e *currentVersions* também podem ser especificadas. Além disso, é possível a utilização da palavra reservada *allVersions* a fim de definir que versões de objetos em todos os estados serão restringidas por uma versão de restrição.

É importante notar que, apesar do fato de que no TVM apenas versões de objetos no estado em trabalho possam ser alteradas, restrições sobre diferentes estados possuem sentido. A principal função destas restrições é definir características obrigatórias para versões e um determinado estado, de forma que uma versão só possa ser promovida a um novo estado caso ela possua as características exigidas pelo mesmo.

Os exemplos a seguir ilustram respectivamente parte da especificação de uma versão de restrição sobre as versões consolidadas e estáveis de objetos e outra sobre todas as versões de objetos.

```
... versioning scope {consolidatedVersions, stableVersions}
... versioning scope allVersions
```

Vale notar que as versões de uma restrição de integridade devem restringir versões de objetos em diferentes estados. Assim, não é possível a existência de duas ou mais

versões de uma única restrição de integridade que restrinjam versões de objetos em um mesmo estado. Além disso, caso a cláusula *versioning scope* seja omitida no momento de criação de uma versão de restrição, assume-se que ela restringirá versões de objetos em todos os estados não restringidos pelas outras versões dessa restrição.

O controle do versionamento de restrições de integridade é feito de forma análoga ao controle definido para objetos versionados e versões do modelo TVM. No momento de criação de uma restrição de integridade temporal de versões, é criada uma versão única. Após a criação de uma nova versão para a restrição, passa a se controlar sua hierarquia de derivação através de um grafo acíclico dirigido.

A Figura 6.3 ilustra graficamente a representação de restrições tradicionais e restrições temporais de versões. As restrições *R1* e *R2* são temporais de versões, diferentemente da restrição *R3* que é tradicional. A restrição *R2* possui as versões *V1*, *V2*, *V3* e *V4*, cuja hierarquia de derivação é representada através do grafo apresentado. Apesar de estarem representadas da mesma forma, os tratamentos recebidos pelas restrições *R1* e *R3* devem ser distintos. O histórico de alterações, os períodos de validade e a abrangência de versionamento da restrição *R1* devem ser considerados. Já a restrição *R3* é válida durante todo o tempo de vida da base, todas as versões de objetos e possui armazenado apenas seu estado atual. O mesmo tratamento aplicado à restrição R1 deve ser aplicado a cada uma das versões da restrição *R2*.

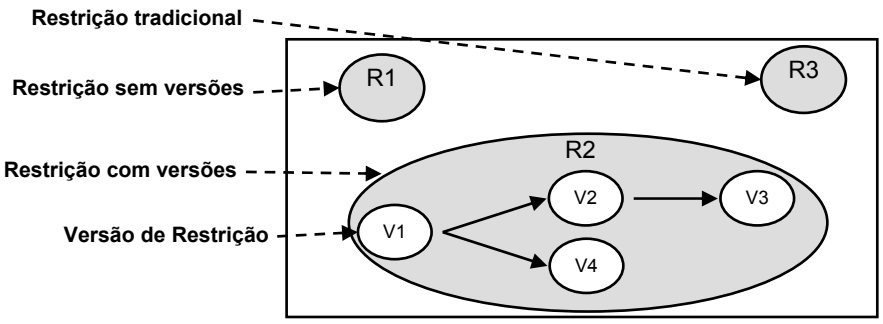


Figura 6.3: Representação de restrições e suas versões

O controle da temporalidade de restrições de integridade temporais de versões também é baseado no modelo TVM. Ele é feito de forma análoga ao controle definido para manipulação de objetos versionados, versões de objetos, atributos e relacionamentos temporais do modelo. É utilizado o tempo ramificado para restrições temporais de versões e o tempo linear para cada uma de suas versões. Dessa forma, são permitidas diferentes versões válidas de uma mesma restrição em um determinado instante, mas cada uma das versões possui um único estado atual.

Versões de restrições possuem associados rótulos temporais referentes aos seus intervalos de tempo de transação e validade. Através dos rótulos temporais *ivTime*, *fvTime*, *itTime*, *ftTime*, são representados respectivamente os tempos de validade inicial e final e os tempos de transação inicial e final.

O tempo de transação é controlado de forma automática pelo SGBD. No momento de criação de uma versão de restrição, seu tempo de transação inicial recebe um valor referente ao instante de tempo atual (*now*) e seu tempo final de transação recebe um valor nulo (*null*). Antes de uma alteração sobre a versão, seu tempo de transação final é definido pelo instante de tempo atual (*now*). Caso o tempo de validade final da versão antiga não tenha sido definido, uma cópia é criada com o tempo de validade final igual ao instante anterior ao tempo atual (*now*-1) e tempo de transação final nulo (*null*). A

alteração é efetuada sobre outra cópia da versão anterior cujo tempo de transação inicial e final recebem respectivamente o instante atual (*now*) e o valor nulo (*null*).

É permitida ao usuário a definição do tempo de validade inicial e final de versões de restrições. Caso o tempo de validade inicial não seja especificado, é assumido o instante de tempo atual (*now*), informando que a validade se inicia no mesmo instante. Quando o tempo de validade final é omitido, assume-se o valor nulo (*null*), indicando que o estado da versão será valido até que ele seja alterado ou o usuário defina um tempo de validade final para ele.

Além dos tempos de validade e de transação de suas versões, restrições de integridade temporais de versões (com todas as suas versões) também possuem tempos de validade e de transação iniciais e finais representados pelos rótulos temporais *ivTime*, *fvTime*, *itTime*, *ftTime*. O controle sobre esses tempos é feito de forma análoga ao controle da temporalidade de objetos versionados no modelo TVM. Além disso, as regras de integridade temporal entre os tempos de objetos versionados e versões no modelo TVM também devem ser respeitadas em relação aos tempos relacionados a restrições temporais de versões e suas versões.

6.2.2 Pontos de Verificação

Através da cláusula *when* da linguagem TVCL são especificados os pontos de verificação de restrições de integridade, ou seja, os momentos onde sua validação deve ser efetuada. A linguagem TVCL permite a especificação de pontos de verificação baseados na execução de operações que alteram o estado da base de dados, ou baseados na ocorrência de eventos, especificados através de expressões temporais.

Pontos de verificação baseados em operações podem estar relacionados à ocorrência de operações de inserção, alteração ou remoção de objetos ou versões de objetos. Através das palavras reservadas *insert*, *update* e *delete* é permitida a definição de pontos referentes à execução de uma operação de atualização específica. Além disso, através da palavra reservada *change*, definem-se pontos de verificação relacionados à execução de qualquer uma das operações de atualização citadas. Vale notar que a mudança de estado de uma versão de objeto á considerada como uma atualização ocorrida sobre ela.

Além da definição do tipo de operação considerado por um ponto de verificação, também devem ser definidos quais elementos da base de dados serão considerados. Quando especificado somente o nome de uma classe temporal de versões, apenas atualizações sobre as versões correntes de objetos são consideradas. Caso seja acrescentada a palavra reservada *versions* ao nome dessa classe, atualizações sobre todas as versões de objetos são consideradas. Além disso, caso a especificação seja sobre uma classe tradicional, atualizações sobre todos os seus objetos são considerados e não é permitida a utilização da palavra reservada *versions*.

Vale notar que, quando considerada apenas a operação de inserção, a utilização da palavra reservada *versions* não é possível, visto que no momento de inserção de um novo objeto versionado uma única versão é criada.

Considerado a possibilidade de alteração de dados no presente, no passado e no futuro quanto ao temo de validade de versões de objetos, também é possível a utilização da palavra reservada *ever* antes da especificação de uma operação. Sua utilização permite definir pontos de verificação referentes à execução de alterações sobre dados válidos em qualquer período. A omissão da palavra *ever* especifica pontos de verificação referentes apenas à execução de alterações sobre os dados atualmente válidos.

A fim de exemplificar a definição de pontos de verificação, considere a existência de uma classe temporal de versões chamada *funcionarios*, assim como pode ser visto na Figura 6.4. O histórico de datas de admissão e demissão de funcionários de uma empresa é armazenado através de atributos temporais dessa classe.

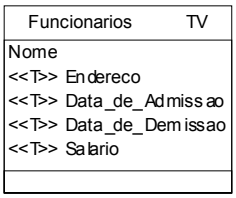


Figura 6.4: Representação gráfica da classe *funcionarios*

Nesse contexto, considere as seguintes definições de pontos de verificação de uma restrição sobre a classe *funcionarios*:

```
... when {on delete Funcionarios} ...

... when {on delete Funcionarios.versions} ...

... when {on ever change Funcionarios.versions} ...
```

De acordo com o primeiro exemplo, a restrição em questão será verificada sempre que ocorrer a remoção da versão corrente e atualmente válida de um dos objetos da classe *funcionarios*. A restrição do segundo exemplo será verificada sempre que ocorrer a remoção de qualquer uma das versões atualmente válidas de objetos da classe *funcionarios*. De forma mais genérica, a restrição do terceiro exemplo será verificada sempre que ocorrer uma inserção de um novo funcionário, ou uma alteração ou remoção de qualquer uma das versões de objetos da classe *funcionarios*, válidas em qualquer período de tempo.

Considerando apenas a operação de alteração, é possível ainda a definição de pontos de verificação relacionados a mudanças em atributos ou relacionamentos específicos. Dessa forma, a restrição definida no exemplo abaixo é verificada sempre que houver uma alteração sobre o valor do atributo *endereco* presente em qualquer uma das versões atualmente válidas de objetos da classe *funcionarios*.

```
... when {on update Funcionarios.versions.endereco} ...
```

Pontos de verificação baseados em operação podem ainda ser imediatos ou protelados, definidos respectivamente pela utilização ou omissão das palavras reservadas *immediately on* em sua especificação. Um ponto de verificação imediato refere-se ao momento imediatamente após a execução da operação de atualização de dados em questão, enquanto um ponto protelado refere-se ao momento de término de execução de uma transação completa, a qual envolve a operação de atualização de dados (SANTOS, 1980; DOUCET et al., 1997). Os exemplos a seguir demonstram respectivamente a definição de restrições com ponto de verificação imediato e protelado sobre alterações em qualquer uma das versões de objetos da classe *funcionarios*, válidas em qualquer período.

```
... when {immediately on ever update Funcionarios.versions} ...

... when {on ever update Funcionarios.versions} ...
```

Além dos pontos de verificação baseados na ocorrência de operações de atualização de dados, a linguagem TVCL permite especificar pontos de verificação baseados na ocorrência de eventos temporais. Esses pontos, determinados na cláusula *when* através de expressões temporais, são alcançados sempre que o resultado de sua expressão se altera de um estado falso (*false*) para um estado verdadeiro (*true*).

As expressões utilizadas são as mesmas expressões temporais aceitas na cláusula *where* de uma consulta TVQL, considerando as extensões propostas nesse trabalho. É permitida a utilização dos operadores lógicos *and* e *or* a fim de separar comparações simples ou sub-expressões do mesmo tipo através de um procedimento recursivo. Além disso é permitida a utilização do operador de negação *not*.

Dentre as comparações aceitas estão comparações de tempo pré-definidas para instantes. Essas utilizam os operadores =, >, <, >=, <= e <> a fim de comparar instantes definidos por propriedades temporais, instantes iniciais e finais de validade e transação de objetos versionados, versões, atributos ou relacionamentos, ou instantes apresentados explicitamente através de constantes.

Comparações de tempo pré-definidas para intervalos também são aceitas. Essas utilizam os operadores *INTERSECT*, *OVERLAP* e *EQUAL* a fim de permitir a comparação entre intervalos definidos explicitamente por constantes, além de intervalos de validade e de transação de objetos versionados, versões, atributos ou relacionamentos.

Por fim, comparações de tempo entre instantes e intervalos também são permitidas através da utilização dos operadores *BEFORE*, *INTO* e *AFTER*. Essas comparações aceitam todos os intervalos e instantes permitidos nos outros dois tipos de comparações.

Deve-se notar que, diferentemente das expressões utilizadas em consultas TVQL, o acesso a propriedades e características de objetos e versões de objetos só é possível através da utilização conjunta do nome de sua classe. O uso de um apelido (*alias*) no lugar do nome da classe também não é permitido. Entretanto, a utilização a palavra reservada *versions* após o nome da classe é possível. Na linguagem TVQL, essa restrição não ocorre, pois os objetos e versões utilizados são especificados na cláusula *from*, onde também podem ser definidos seus apelidos.

Para considerar todo o histórico de alterações de atributos e relacionamentos utilizados na expressão temporal, a palavra reservada *ever* pode ser utilizada no inicio da expressão. Nesse caso, a função *present* utilizada na TVQL pode ser necessária quando desejado considerar apenas os dados atuais em parte de uma expressão.

Uma unidade temporal deve ser especificada logo após cada expressão que representa um ponto de verificação baseado em eventos. Essa unidade se refere à freqüência de verificação de restrições. As freqüências possíveis são: segundos, minutos, horas, dias, meses e anos, respectivamente representadas pelas palavras reservadas *secondly*, *minutely*, *hourly*, *daily*, *monthly*, e *yearly*. Essas palavras devem ser especificadas logo após a palavra reservada *evaluate*. Vale ressaltar que a definição de uma freqüência de verificação é considerada de extrema importância a fim de possibilitar a verificação de restrições com pontos de verificação baseados em eventos. Além disso, o usuário deve estar ciente que a escolha dessa freqüência estará intimamente relacionada ao desempenho do sistema.

A fim de permitir a exemplificação de pontos de verificação baseados em eventos, considere novamente a classe *funcionarios*. Nesse contexto, a seguinte restrição *"O salário de um funcionário após um período de experiência de três meses não pode ser menor do que o salário base de sua categoria."* deve ser verificada em um ponto especificado como segue:

```
... when{ ( incDate(funcionarios.versions.data_de_admissao,month,3.0)
          >= now evaluate daily ) } ...
```

De acordo com o ponto de verificação definido, essa restrição deve ser verificada no momento de término do período de três meses de experiência de algum funcionário da empresa, considerando que a validade da expressão seja verificada diariamente. Para tanto, todas as versões de objetos da classe funcionários são consideradas e apenas o estado atual de valores de seus atributos e relacionamentos é analisado. Vale notar que é utilizada a função *incDate*, definida na Seção 6.3.3, que retorna um instante referente à data de admissão de um funcionário acrescido de três meses.

6.2.3 Controles sobre a Ordem de Precedência de Verificação

Através das cláusulas opcionais *follows* e *precedes* da linguagem TVCL, é possível a definição de uma ordem de precedência de verificação de restrições de integridade sobre uma mesma base de dados. Nessas cláusulas são listadas respectivamente as restrições que devem ser verificadas antes da restrição em questão e as restrições cuja verificação deve ocorrer após sua verificação. Nota-se que a ordenação considerada é classificada como *inter-transações*, ou seja, a ordem definida deve ser respeitada por todas as verificações de restrições que ocorram sobre a base, independentemente de quais transações que as acionaram.

Conforme descrito durante a classificação de restrições, a ordem de precedência de verificação só faz sentido em relação a restrições com propósito causativo, ou seja, restrições que possuem uma lista de ações de atualização de dados definidas para serem executadas logo após sua violação. Em relação à linguagem TVCL, uma restrição de propósito causativo possui obrigatoriamente sentenças da linguagem de atualização de objetos TVL/SE relacionadas à sua violação.

Em relação à definição dessa ordenação, algumas regras devem ser obedecidas, a fim de evitar inconsistências durante a manutenção de restrições com ordem de precedência. Deve-se garantir que:

- todas as restrições citadas na cláusula *precedes* de uma restrição *X* devem citá-la em suas cláusulas *follows*;
- dentre as restrições citadas na cláusula *follows* e *precedes* de uma restrição *X*, não pode estar a própria restrição *X*;
- uma restrição *X* não pode aparecer em ambas cláusulas *follows* e *precedes* de uma restrição *Y*.

Com base nessas regras, considere os seguintes exemplos de especificação de restrições:

```
create TV constraint R0 ...
precedes {R1}

create TV constraint R1 ...
follows {R0}
precedes {R2, R3}

create TV constraint R2 ...
follows {R1}

create TV constraint R3 ...
follows {R1}
```

De acordo com as especificações, fica definido que a verificação da restrição *R1* deve ocorrer obrigatoriamente após a verificação da restrição *R0*. Além disso, as verificações de ambas restrições *R2* e *R3* só podem ocorrer depois de efetuada a verificação da restrição *R1*. A ordem de precedência estipulada por esses exemplos pode ser representada através do grafo presente na Figura 6.5.

É importante notar que, a fim de evitar a ocorrência de *deadlock* durante a verificação de restrições com ordem de precedência, todos os grafos criados com base nas cláusulas *follows* e *precedes* devem obrigatoriamente ser *acíclicos dirigidos*.

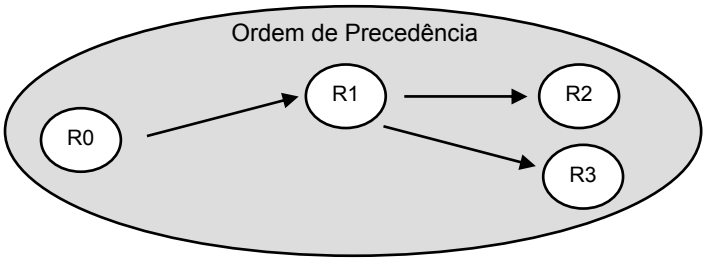


Figura 6.5: Representação de ordem de precedência de verificação

6.2.4 Exemplos de Especificação

Considere a existência de um banco de dados utilizado por uma aplicação de controle de matrículas em uma universidade. Parte dessa base de dados é ilustrada pelo diagrama de classes do modelo TVM na Figura 6.6.

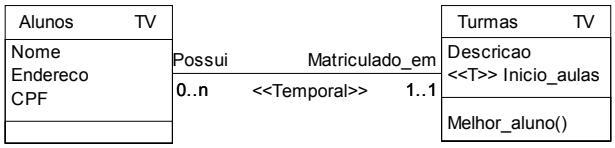


Figura 6.6: Diagrama de classes da aplicação exemplo

Nessa base de dados existem as classes temporais de versões *alunos* e *turmas*. Através de um relacionamento temporal (*matriculado_em*) entre essas classes, são definidos os alunos que participam de uma determinada turma. A cardinalidade desse relacionamento é definida de forma que cada aluno deve obrigatoriamente estar matriculado em uma única turma. Nesse contexto, a restrição *"Cada turma deve ter no máximo quarenta alunos"*, definida a partir do dia 10 de fevereiro de 2005, pode ser especificada da seguinte maneira:

```
create TV constraint tamanhoTurmas
when { immediately on update alunos.versions.matriculado_em }
   if exists( select t.tvoid
                from turmas.versions t
                where ( select count(*)
                        from turmas.versions t2, alunos.versions a
                        where t2.tvoid = t.tvoid and
                        a.matriculado_em.tvoid = t2.tvoid) > 40 )
   then rollback
validity [10/02/2005..]
versioning scope {workingVersions}
```

Dessa forma, a restrição será violada se houver mais de quarenta alunos matriculados em versões em trabalho de algum objeto da classe *turmas*. Apenas os valores atuais de versões de objetos são considerados e a restrição passa a ser válida a partir do dia 10/02/2005. Fica especificado que a restrição em questão será verificada

sempre que houver alguma alteração sobre o relacionamento *matriculado_em* válido atualmente em uma versão em trabalho de objetos da classe *alunos*. Além disso, essa verificação será feita em caráter imediato, ou seja, a verificação ocorrerá imediatamente após a operação que alterou o relacionamento e não após o término da transação em questão. Além disso, caso ocorra uma violação da restrição, a transação causadora deverá ser desfeita (*rollback*).

O acesso ao identificador (OID) da turma onde um aluno está matriculado é feito pela expressão *a.matriculado_em.tvoid*. Esse acesso se baseia na utilização de expressões de caminho, incorporadas à linguagem TVQL por Galante (2003) e descritas na Seção 6.3.1.

Considerando a existência da classe *funcionarios*, utilizada anteriormente, um novo exemplo de restrição é apresentado. Nesse contexto, a seguinte restrição *"O salário de um funcionário após um período de experiência de três meses não pode ser menor do que 300."* pode ser especificada como segue:

```
create TV constraint salarioExperiencia
when { ( incDate(funcionarios.versions.data_de_admissao,month,3.0) >= now
         evaluate monthly ) }
   if exists( select f.tvoid
                from funcionarios.versions f
                where incDate(f.data_de_admissao,month,3.0) >= now and
                      f.salario < 300 )
   then ( update funcionarios
          set salario = 300
          where tvoid in (select tvoid from inconsistentObjects) )
versioning scope {workingVersions}
```

De acordo com o ponto de verificação definido, essa restrição deve ser verificada no momento de término do período de três meses de experiência de algum funcionário da empresa, considerando que a expressão temporal seja avaliada mensalmente. Para tanto, somente versões em trabalho de objetos da classe funcionários são consideradas e apenas o estado atual de valores de seus atributos e relacionamentos é analisado. Por fim, caso a restrição seja violada, os valores dos salários atuais das versões de objetos que violaram a restrição receberão automaticamente o valor mínimo 300.

6.3 Utilização de Consultas TVQL em Restrições

A linguagem TVQL, definida a fim de permitir consultas a bases de dados criadas sobre o modelo TVM, é utilizada durante a especificação de restrições de integridade. É importante notar que, nesse contexto, são permitidas apenas consultas cujos retornos sejam identificadores de objetos ou versões de objetos (OID) de uma única classe do modelo de dados. Além disso, independentemente de suas formulações, consultas TVQL referentes a restrições temporais de versões possuem sempre um escopo limitado ao período de tempo de validade de cada versão de restrição e ao seu escopo de versionamento, definidos nas cláusulas *validity* e *versioning scope*. Dessa forma é possível dizer que a consulta TVQL, especificada em uma versão de restrição de integridade, não é definida sobre a base como um todo, mas apenas sobre a "visão" (*view*) da base de dados afetada pela versão de restrição em questão.

Considerando a utilização de linguagens de consulta para permitir a especificação de restrições de integridade, Chomicki e Toman (1995) afirmam que o poder de expressão de uma linguagem de especificação de restrições depende do poder de expressão da linguagem de consulta utilizada como base. Assim, fica claro que o poder de expressão da linguagem TVCL depende fortemente do poder de expressão da TVQL.

Conforme pôde ser notado na introdução à TVQL, apresentada no Capítulo 2, essa linguagem possui o poder de expressar consultas que envolvam dados atuais ou históricos de diferentes versões de objetos de uma base de dados temporal de versões. Entretanto, por ser uma linguagem baseada em SQL, algumas características comuns a linguagens de consulta a bases de dados orientadas a objetos não estão presentes. Dentre elas, estão a possibilidade de navegação entre relacionamentos de objetos e a possibilidade de chamada de métodos em consultas.

Com isso, a fim de aumentar o poder de expressão da linguagem de especificação de restrições, algumas extensões à linguagem TVQL são necessárias. Essas extensões visam possibilitar a utilização de características de uma linguagem de consulta orientada a objetos, como a linguagem OQL definida para o modelo de dados do padrão ODMG. Além disso, a utilização de novas funções de manipulações de instantes de tempo se faz necessária em alguns casos. As principais características dessas extensões são descritas a seguir.

Nota-se ainda a possibilidade de acrescentar à TVLQ outras características de linguagens de consulta orientadas a objetos e funcionalidades sobre a manipulação de tempos e versões.

6.3.1 Expressões de Caminho

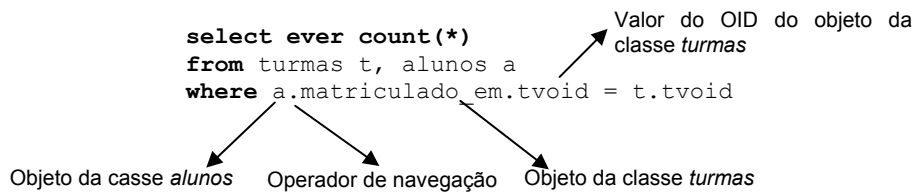
A possibilidade de navegação de relacionamentos através de expressões de caminho foi incorporada à linguagem TVQL por Galante (2003). A estrutura dessas expressões é similar à linguagem de consulta OQL (*Object Query Language*) (CATTELL et al., 2000). A navegação de relacionamentos é representada pelo símbolo ponto (.) e pode ser utilizada na cláusula *where* entre o nome de uma classe (ou alias de classe) e o nome de um relacionamento pertencente a ela. Sua utilização também é possível de forma direta na cláusula *from* ou na especificação dos dados de retorno da consulta (clausula *select*).

É permitido o acesso ao(s) objeto(s) relacionado(s) a um determinado objeto. Considere um relacionamento *R* entre as classes *A* e *B*. Caso esse relacionamento seja do tipo *1:1* ou *n:1*, o retorno da expressão *A.R* é um único objeto da classe *B*, relacionado ao objeto em questão da classe *A*. Nesse caso, é possível por exemplo o acesso ao identificador desse objeto através da expressão *A.R.tvoid*. Caso o relacionamento *R* seja do tipo *1:n* ou *n:n*, o retorno da expressão *A.R* é o conjunto de objetos, de uma mesma classe, relacionados ao objeto em questão da classe *A*. Nesse caso o acesso aos dados referentes a um único objeto do conjunto de retorno só é possível através da utilização de operadores de conjuntos (*sets*) e listas (*bags*), definidos para a linguagem OQL.

Em relacionamentos parciais (cardinalidade mínima igual a zero) existe ainda a possibilidade de retorno de um valor nulo (*null*), que ocorre sempre que um objeto da classe *A* não estiver relacionado a nenhum objeto da classe *B*.

De forma análoga ao acesso aos atributos de um determinado objeto ou versão de objeto, a temporalidade dos relacionamentos considerados pela consulta depende diretamente da utilização das palavras reservadas *ever*, *versions* e da função *present* definidas para a TVQL.

A fim de exemplificar a utilização de expressões de caminho, considere novamente as classes temporais de versões *alunos* e *turmas* e o relacionamento temporal definido entre elas *matriculado_em* (Figura 6.6). Considerando que cada aluno deve obrigatoriamente estar matriculado em uma única turma, a seguinte consulta TVQL é analisada:

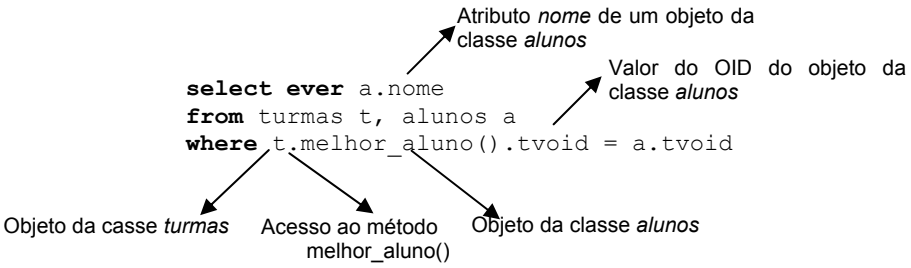


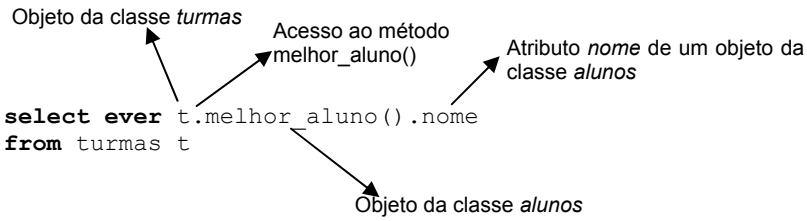
Através do acesso ao OID da turma onde cada aluno está matriculado, o resultado dessa consulta é uma lista com os números de alunos matriculados em cada uma das turmas da universidade, considerando todo o seu histórico de alterações durante o tempo de vida da base de dados e apenas versões correntes de objetos das classes *turmas* e *alunos*.

6.3.2 Chamada a Métodos

Outra característica acrescentada à linguagem TVQL é a possibilidade de efetuar chamadas a métodos pertencentes às classes do modelo de dados. Essa funcionalidade, definida para a linguagem OQL, permite apenas o acesso a métodos que possuam algum valor de retorno (*funções*) e esse valor pode ser utilizado na cláusula *where* de consultas ou diretamente no retorno dos dados da consulta (clausula *select*). O acesso a um método também é especificado através do símbolo ponto (**.**).

Considere novamente as classes temporais de versões *alunos* e *turmas*. Através da chamada do método *melhor_aluno()* da classe *turmas*, é possível o retorno de um único objeto referente ao melhor aluno de uma turma. Considere agora as seguintes consultas:





Na primeira consulta, o acesso ao método *melhor_aluno()* de objetos da classe *turmas* é feito na cláusula *where* e o identificador do objeto referente ao melhor aluno é comparado ao identificador de todos os alunos. A segunda consulta utiliza o acesso ao método *melhor_aluno()* diretamente no retorno dos dados, evitando assim a comparação dos identificadores de alunos. De qualquer forma, as duas consultas retornam uma lista com os nomes dos melhores alunos de cada turma da universidade, considerando todo o seu histórico e alterações e apenas versões correntes de objetos.

6.3.3 Funções de Manipulação de Instantes

Além do acréscimo à TVQL das características de linguagens de consulta orientadas a objetos, também se faz necessário o acréscimo de novas funções de manipulação de instantes a fim de aumentar o poder de expressão dessa linguagem de consulta e, conseqüentemente, aumentar o poder de expressão da linguagem de especificação de restrições TVCL.

As seguintes funções de manipulação de instantes são acrescidas à linguagem TVQL e podem ser utilizadas nas cláusulas *where* ou diretamente no retorno de dados:

- ***incDate (baseDate**: instant, **granularity**: tGranularity, **value**: float) : instant.* Essa função retorna um instante referente ao instante base, recebido através do parâmetro *baseDate*, incrementado ao valor passado pelo parâmetro *value* sob a granularidade temporal definida no parâmetro *granularity*. As possíveis granularidades são definidas pelas constantes: *second*, *minute*, *hour*, *day*, *week*, *month*, *year*, *decade*, *century*.
- ***decDate (baseDate**: instant, **granularity**: tGranularity, **value**: float) : instant.* Essa função é praticamente análoga à função anterior, exceto por retornar um instante referente ao instante base decrementado do valor passado pelo parâmetro *value* sob a granularidade temporal definida no parâmetro *granularity*.
- ***dayOfWeek (baseDate**: instant) : tDaysOfWeek.* Essa função recebe um instante através do parâmetro de entrada *baseDate* e retorna uma constante que indica o dia da semana referente a esse instante. As possíveis constantes de retorno são: *monday*, *tuesday*, *wednesday*, *thursday*, *friday*, *saturday*, *sunday*.
- ***isLeapYear(baseDate**: instant) : boolean.* Essa função recebe um instante através do parâmetro de entrada *baseDate* e retorna um valor do tipo *boolean* que indica se o ano em questão é bissexto.

Vale notar que todos os parâmetros existentes nas funções citadas são somente de entrada (passagem por valor).

6.4 Utilização de Sentenças TVL/SE em Restrições

A especificação de restrições de integridade pode estar relacionada a uma lista de ações de violação que alteram o estado da base de dados a fim de recompor sua integridade, imediatamente após a violação de uma restrição. Restrições com ações de violação são comuns em casos onde a violação independe de alterações no estado da base de dados (pontos de verificação baseados em eventos) e, conseqüentemente, não existe a possibilidade de retorno a um estado anterior (*rollback*).

Na linguagem TVCL, a especificação de ações de violação é feita na cláusula *then* através da utilização de parte de uma linguagem chamada TVL/SE (linguagem de atualização de objetos).

6.4.1 Linguagem de Atualização de Objetos TVL/SE

A linguagem de especificação, versionamento e modificação de esquemas (TVL/SE) proposta pelo modelo TVSE (GALANTE, 2003) pode ser dividida e três partes distintas: especificação e versionamento de esquemas, modificação de esquemas e atualização de objetos. De acordo com o foco do presente trabalho, esta seção apresentará somente as características da linguagem TVL/SE que estão diretamente relacionadas à atualização de objetos, ou seja, a linguagem de atualização de objetos do modelo TVSE (GALANTE; EDELWEISS; SANTOS, 2003). Além disso, é considerada

a existência de uma única versão de esquema, visto que o modelo TVM não considera a possibilidade de versionamento de esquemas. Maiores detalhes podem ser encontrados nas referências originais.

A linguagem de manipulação de dados permite ao usuário expressar modificações simultâneas sobre os dados extensionais de diferentes versões de esquema controladas pelo modelo TVSE. A versão simplificada da BNF da linguagem pode ser vista a seguir:

```
<query> ::= ( <insert> | <update> | <delete> ) <schemaVersionSelection>
<insert> ::= insert into <className>
   ["("<attribName>"="<expression> [validity interval<defInterval>]
   {","<attribName>"="<expression> [validity interval<defInterval>]}*")"]
   values "(" <constant> {"," <constant>}*")"
   [validity interval <defInterval>]
   [validity interval <defInterval>]
<delete> ::= delete from <className>
   [where <condition>][fvalidity "=" <instantValue>]
<update> ::= update <className>
   set <attribName> "=" <expression>[validity interval <defInterval>]
   {"," <attribName>"="<expression> [validity interval <defInterval>]}*
   where <condition>
```

O intuito da cláusula *<schemaVersionSelection>* é identificar as versões de esquema consideradas na atualização de dados. Entretanto, os exemplos de atualização de dados apresentados a seguir trabalharão apenas com uma versão de esquema (versão corrente), representada na cláusula *<schemaVersionSelection>* pela opção *current*.

Na cláusula *WHERE <condition>* das operações *UPDATE* e *DELETE*, o usuário deve definir quais objetos serão considerados pela alteração ou remoção. A linguagem TVQL, definida para o modelo TVM e apresentada anteriormente, é utilizada para o retorno dos dados.

As operações de manipulação (*INSERT*, *UPDATE* e *DELETE*) são definidas para efetuar as alterações de objetos. Em relação ao tempo de validade, essas operações podem modificar o presente, o passado e o futuro dos dados extensionais de bases de dados, possibilitando modificações prospectivas e retrospectivas.

Para exemplificar a utilização das operações de manipulação, considere a existência de uma classe temporal versionada chamada *Pais*. Os atributos *regime_politico*, *nome*, *presidente* e *renda_per_capita* fazem parte de seu conjunto de atributos e somente o atributo *nome* é do tipo não temporal.

A operação *INSERT* é implementada a fim de possibilitar a criação da primeira versão de um objeto. Através da cláusula opcional *VALIDITY INTERVAL*, o tempo de validade do novo objeto pode ser especificado e, caso o objeto possua atributos temporais, o usuário poderá fornecer seus intervalos de validade de forma recursiva.

De acordo com a realidade apresentada, a inserção do país ‘Brasil’ com o regime político ‘Democracia’, o presidente ‘Lula’ e renda per capita igual a dois mil seria efetuada através do seguinte comando:

```
insert into Pais
(nome = 'Brasil'
 regime_politico = 'Democracia'
 presidente = 'Lula' validity interval [01/01/2004..]
 renda_per_capita = 2000 validity interval [01/01/2004..])
validity interval [22/04/1500..]
current
```

Note que as validades do objeto criado, de seu presidente e de sua renda per capita foram definidas explicitamente, enquanto o intervalo de validade do regime político recebe o valor padrão [*now* ..] por não ter sido especificado.

Quanto à utilização da operação *UPDATE*, existem duas situações possíveis dependendo da temporalidade do atributo atualizado. Quando um atributo não temporal é modificado, uma nova versão do objeto que o possui é criada. Diferentemente, quando um atributo temporal é modificado, ambos valores (valor antigo e valor novo) farão parte da versão do objeto em questão. Nesse caso, o intervalo de validade do novo valor deve ser obrigatoriamente informado.

Suponha agora que a renda per capita atual da versão corrente do objeto 'Brasil' tenha que ser alterada para o valor um mil e quinhentos. Dessa forma, a sentença de atualização seria:

```
update Pais
set renda_per_capita = 1500
validity interval [01/02/2005..]
where (select renda_per_capita
       from Pais
       where nome = 'Brasil')
current
```

Quanto à exclusão de dados, existem três situações possíveis: a exclusão de um atributo temporal, de uma versão de objeto ou de um objeto versionado. Em qualquer um dos casos, a operação *DELETE* efetua uma exclusão lógica, de forma que o tempo final de validade seja fechado, recebendo o valor referente ao momento em que ocorreu a exclusão. Quando um atributo temporal é removido, seu tempo de validade final recebe o valor da cláusula *fValidity* especificada pelo usuário (veja o exemplo a seguir). Quando essa cláusula não é especificada, o tempo presente (*now*) é assumido. Quando uma versão de objeto é removida, seu tempo de vida é fechado. Caso existam atributos ou relacionamentos temporais, seus tempos finais de validade também recebem o tempo validade final da versão. De forma análoga, quando um objeto versionado é removido, seu tempo de vida é finalizado e conseqüentemente todas as suas versões e os atributos temporais das versões são removidos.

Suponha agora a existência do país 'Itália' representado através de um objeto da classe *Pais*. Suponha também que o valor atual do atributo *presidente* da última versão do objeto tenha que ser excluído no dia trinta e um de janeiro de 2006. O comando que efetuaria essa exclusão seria:

```
delete from Pais
where (select p.presidente
       from Pais.versions p
       where p.nome = 'Itália' and p.tvoid = lastVersion
             and p.fvTime = null)
fvalidity = 31/01/2006
current
```

Nesse comando, a cláusula *SELECT* é uma consulta TVQL que retorna o presidente atual da última versão do objeto 'Itália'. Visto que a classe *Pais* é temporal de versões, o valor *null* presente no tempo de validade final da última versão do objeto indica que seu intervalo de validade está aberto, ou seja, a versão ainda não foi excluída.

6.4.2 Representação de Ações de Violação

Considerando que este trabalho não trata a possibilidade de versões de esquemas, a utilização da cláusula <*schemaVersionSelection*> da linguagem de atualização de objetos foi suprimida.

A cláusula *WHERE* <*condition*> das operações *UPDATE* e *DELETE* também sofre algumas modificações. O intuito dessa cláusula é identificar quais objetos serão considerados pela atualização de dados, através da utilização de uma consulta TVQL para o retorno dos dados. A fim de melhorar seu entendimento e se aproximar do padrão SQL, uma mudança sintática foi feita de forma a obrigar a definição de uma expressão lógica, ao invés de uma simples consulta, nesta cláusula. O uso de sub-consultas TVQL é possível nestas expressões. Essas sub-consultas sofrem as seguintes alterações: (i) as extensões da linguagem TVQL apresentadas anteriormente também podem ser utilizadas, e (ii) a consulta TVQL pode ter em sua cláusula *from* a palavra reservada *inconsistentObjects*. Essa palavra representa os identificadores (OID) do conjunto de versões de objetos ou objetos tradicionais retornados pela consulta utilizada na verificação da restrição (cláusula *if exists*).

Por fim, a cláusula *SET* da operação *UPDATE* também foi estendida a fim de possibilitar a utilização de expressões de caminho e de chamadas a métodos com base nos objetos retornados pela consulta de sua cláusula WHERE <condition>.

De acordo com as definições do modelo TVM, somente versões de objetos no estado *em trabalho* aceitam alterações de dados. Dessa forma, é importante notar que as ações de violação de restrições de integridade devem estar relacionadas apenas a versões de objetos *em trabalho*. Assim, é comum que a cláusula *versioning scope* da especificação de uma restrição deste tipo restrinja sua ação apenas a versões de objetos nesse estado.

De forma a exemplificar restrições com ações de violação especificadas através da linguagem TVCL, considere novamente as classes *alunos* e *turmas* utilizadas em exemplos anteriores. Considere ainda a inclusão à classe *turmas* do atributo temporal *Lotação_Excedida*, de tipo *boolean*, com intuito de identificar as turmas com lotação de alunos excedida. A Figura 6.7 ilustra a nova realidade dessas classes.

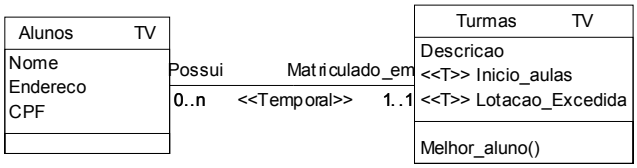


Figura 6.7: Novo diagrama de classes da aplicação exemplo

De acordo com essa nova realidade a mesma restrição *"Cada turma deve ter no máximo quarenta alunos"*, definida a partir do dia 10 de fevereiro de 2005, pode ser especificada de maneira diferente:

```
create TV constraint tamanhoTurmas
when { immediately on update alunos.versions.matriculado_em }
   if exists( select t.tvoid from turmas.versions t
                where (select count(*)
                       from turmas.versions t2, alunos.versions a
                       where t2.tvoid = t.tvoid and
                       a.matriculado_em.tvoid = t2.tvoid) > 40 )
   then ( update Turmas set Lotacao_Excedida = true
          where tvoid in (select tvoid from inconsistentObjects))
validity [10/02/2005..]
versioning scope {workingVersions}
```

Nesse caso, ao invés da reconstituição da base a um estado anterior (*rollback*), a violação da restrição implica em uma atualização sobre as versões de turmas que causarem a violação, indicando que as mesmas excederam sua lotação máxima.

6.5 Análise de Abrangência da Linguagem

O intuito dessa seção é analisar o poder de expressão da linguagem TVCL, com base na classificação de restrições de integridade apresentada no Capítulo 4. Conforme pode ser observado na Tabela 6.1, a abrangência da linguagem TVCL é analisada em relação às classes definidas em todos os aspectos e critérios da classificação de restrições.

Dessa forma, é possível notar que a utilização conjunta da linguagem TVCL e do diagrama de classes do modelo TVM permite especificar restrições de integridade que se enquadram na maioria das classes previstas na classificação. Nota-se que os principais tipos de restrições não considerados são restrições com origem no modelo de dados, incompletas, relacionadas a diferentes versões de esquema, relacionadas à estrutura ou ao comportamento do esquema, baseadas no tempo com indeterminação e baseadas na utilização de regras dedutivas.

Tabela 6.1: Análise de abrangência da linguagem TVCL

Aspecto	Critério	Análise	Observação
Origem	**-**	PC	A especificação de restrições com origem no modelo de dados não é possível.
Substância	**Abrangência de Estados**	CC	São possíveis restrições sobre um ou mais estados da base.
	Tipo do Restringente e do Restringido	PC	Restringentes e restringidos de tipo esquema não são possíveis.
	Formação do Restringente e do Restringido	CC	A utilização de objetos com formação primitiva e construída é possível.
	Formação da Condição Restritiva	CC	Condições simples, disjuntivas, conjuntivas e múltiplas são possíveis.
	Aspecto Restringente e Restringido	PC	Restrições sobre o comportamento ou a estrutura de esquemas não são possíveis. Além disso, restrições sobre a cardinalidade e o domínio dos dados são definidas no diagrama de classes do TVM.
	Propósito	CC	Completamente abrangido pelas opções da cláusula *then*.
	Abrangência do Restringente e do Restringido	PC	Abrangências relacionadas a mais de um esquema não são previstas.
	Abrangência de Ligação do Restringente e do Restringido	CC	São previstas abrangências de ligação indefinidas, globais e locais a objetos, atributos, relacionamentos, métodos, temporalidades e versionamentos.
	Completude	PC	A especificação de restrições incompletas não é prevista.
Especificação	**Declaração**	PC	Em conjunto com o diagrama de casses do TVM, são possíveis declarações implícitas, explícitas e derivadas. Entretanto, declarações explícitas procedurais não são possíveis.
	Temporalidade da Restrição	PC	O tempo com indeterminação não pode ser utilizado.
	Abrangência de Temporalidade da Restrição	CC	Restrições podem abranger qualquer parte do tempo de vida da base.
	Versionamento da Restrição	CC	São aceitas restrições tradicionais e versionadas.
Aplicação	**Dependência sobre Regras Dedutivas**	NA	A utilização de bases dedutivas não é considerada.
	Ordem de Precedência	PC	Somente ordenações inter-transações são possíveis.
	Ativação e Desativação	NC	-
	Tipo de Ativação e Desativação	NC	-
	Acionamento	PC	Somente o acionamento automático é considerado.
	Tratamento	NA	-
	Veículo	NA	-
	Inspeção	NA	-
	Ponto de Verificação	PC	Não são considerados pontos baseados em eventos internos ou externos não temporais.
Temporalidade	**Tipo de Tempo do Restringente e do Restringido**	PC	O tempo com indeterminação não é considerado.
	Referência	CC	Referências de todos os tipos citados na classificação são possíveis.
	Abrangência Temporal do Restringente e do Restringido	CC	É permitida a abrangência temporal de validade ou de transação no passado, presente e futuro.
	Granularidade Temporal	CC	São possíveis granularidades homogêneas e heterogêneas.
Versionamento	**Tipo de Versionamento do Restringente e do Restringido**	CC	Restringentes e restringidos podem ser versionados ou não versionados.
	Abrangência de Versionamento do Restringente e do Restringido	CC	São possíveis restrições sobre todas as versões ou versões específicas, de acordo com seu estado ou características de configuração.

CC – Completamente Compreendido PC – Parcialmente Compreendido NC – Não Compreendido
NA – Não Aplicável

6.6 Considerações Finais

Este capítulo apresentou a linguagem de especificação de restrições TVCL. Através dessa linguagem, é possível a especificação de restrições de integridade sobre bancos de dados temporais de versões construídos sobre o modelo TVM. Esse acréscimo ao modelo permite que usuários especifiquem restrições de integridade sobre suas bases de dados de uma forma intuitiva, especificando declarativamente quais dados são considerados inconsistentes em relação à restrição especificada. Toda a complexidade envolvida no processo de manutenção de integridade fica transparente ao usuário.

É importante notar que, apesar de estar definida como uma extensão do modelo TVM, a linguagem TVCL não possui utilização restrita a esse modelo. Essa linguagem pode ser aplicada a bases de dados construídas sobre qualquer modelo que aplique os conceitos de tempo e de versão. Além disso, através de pequenas alterações, sua utilização é possível em bases tradicionais, somente temporais ou somente de versões.

A linguagem TVCL se baseia na utilização da linguagem e consulta do modelo TVM e da linguagem de atualização de dados de um modelo de versionamento de esquemas. É permitida a especificação de restrições tradicionais e restrições temporais de versões. Quanto à temporalidade e o versionamento dos dados de bases temporais de versões, ambos tipos de restrição podem restringir propriedades, atributos e relacionamentos de versões específicas de objetos com validade no presente, passado ou futuro.

Restrições podem ainda possuir pontos de verificação baseados na ocorrência de alterações sobre o estado da base de dados ou sobre na ocorrência de eventos, especificados através de expressões temporais. Suas verificações podem respeitar uma ordem pré-definida e, quando violadas, restrições podem estar relacionadas à execução de ações de reparo que alteram o estado da base de dados, ao disparo de um simples alerta, ou à execução de operações que desfazem completamente as transações causadoras de violação (*rollback*).

Vale notar que o intuito dessa linguagem de especificação é permitir ao usuário do TVM especificar restrições de integridade que até o momento só poderiam ser expressas através de linguagem coloquial. A manutenção destas restrições não é tratada neste trabalho.

Após apresentada a linguagem TVCL, sua abrangência foi analisada com base na classificação de restrições apresentada anteriormente. O resultado dessa análise, indica que o conjunto de restrições especificáveis implicitamente no diagrama de classes do TVM unido ao conjunto de restrições especificáveis através da linguagem TVCL pode ser considerado bastante expressivo, visto que apenas restrições com características muito particulares ou não aplicáveis ao modelo TVM não podem ser especificadas.

7 ESTUDO DE CASO

Este capítulo apresenta um estudo de caso sobre a classificação e a linguagem de especificação de restrições de integridade propostas nos capítulos anteriores. Seu conteúdo é composto por uma breve descrição da realidade de uma empresa fictícia que tem a necessidade de desenvolvimento de uma aplicação que auxilie a execução de suas tarefas. A partir desse exemplo, são apresentadas, em linguagem coloquial, diversas restrições aplicáveis sobre a base de dados da aplicação. Dentre elas, estão restrições implícitas na modelagem de dados da aplicação e restrições que devem ser explicitamente declaradas. Restrições do segundo tipo são então especificadas através da linguagem TVCL, proposta no Capítulo 6, e classificadas com base na classificação apresentada no Capítulo 4.

7.1 Exemplo de Aplicação

A empresa *NewSoft*, uma empresa conceituada a nível regional, tem como principal atividade o desenvolvimento e manutenção de softwares administrativos. Devido a facilidades na reutilização de código, padronização, desenvolvimento em grupo e subdivisão de grandes projetos, a empresa tem como norma e padrão a ser seguido, que todo desenvolvimento por ela efetuado deve respeitar os princípios da orientação a objetos. Hierarquicamente, a empresa possui um gerente geral, analistas de sistema, programadores, além de funcionários gerais como de limpeza, segurança, etc.

Para cada projeto em andamento, deve-se ter pelo menos um analista e um programador, que devem participar do grupo de trabalho responsável pelo projeto. Além do coordenador geral, um dos analistas, do grupo responsável pelo projeto, coordena o mesmo e é desejada uma média de dez programadores para cada analista em um grupo de trabalho. Analistas e programadores podem participar de mais de um grupo de trabalho ao mesmo tempo e até mesmo não participarem de nenhum grupo por algum tempo, fato comum em períodos de mudança.

Cada projeto é composto por módulos, representados por objetos, que por sua vez também podem ser compostos por um conjunto de outros objetos. Devem existir sobre cada objeto diversas informações que dizem respeito à função do mesmo, heranças, variáveis de instância, métodos entre outras. Além disso, deverão existir outras informações úteis referentes aos projetos, como data de início, previsão de conclusão, duração, percentual de conclusão, gasto, linguagem de programação utilizada além de dados sobre os padrões, componentes e bibliotecas utilizadas.

Também deverá existir um cadastro relacionado aos dados pessoais de cada funcionário, além de informações sobre salário (subdividido em níveis), férias, licenças, data(s) de admissão, data(s) de demissão (se pertinente), experiência e formação profissional. Além disso, um cadastro de clientes é desejado com informações básicas sobre os mesmos.

7.2 Diagrama de Classes

A Figura 7.1 ilustra o diagrama de classes UML utilizado pela aplicação e construído sob o modelo TVM. Nesse diagrama, as variáveis de instância das classes são apresentadas enquanto seus métodos são omitidos. A granularidade temporal utilizada é baseada em minutos e, sempre que possível, são aplicados os conceitos de tempo e de versão a fim de simplificar a modelagem de dados.

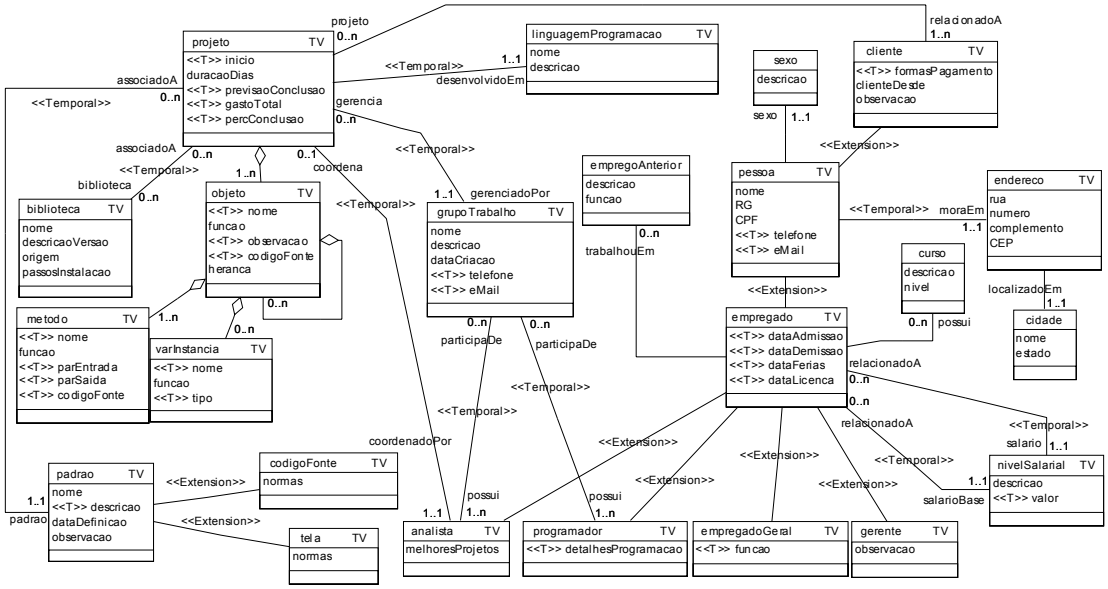


Figura 7.1: Exemplo de modelagem temporal de versões

7.3 Exemplos de Restrições

Através da definição do diagrama de classes da aplicação exemplo, algumas restrições são implicitamente definidas. Essas são decorrentes de regras do modelo TVM ou da própria modelagem de dados apresentada no diagrama de classes da aplicação. Exemplos dessas restrições são apresentados em linguagem coloquial:

- *o tempo associado aos atributos de um objeto versionado deverá estar contido em seu tempo de vida*: decorrente de regras da temporalidade do modelo TVM;
- *versões consolidadas de objetos não podem ser desativadas*: decorrente do diagrama de estados de versões do modelo TVM;
- *versões em trabalho, após derivadas, deverão ser promovidas a estáveis*: decorrente do diagrama de estados de versões do modelo TVM;
- *versões de objetos deverão ser criadas no estado em trabalho*: decorrente do diagrama de estados de versões do modelo TVM;
- *somente analistas e programadores podem participar de grupos de trabalho*: decorrente dos relacionamentos criados no diagrama de classes da aplicação;
- *uma pessoa deve estar associada a um e somente um endereço em um mesmo instante de tempo*: decorrente da cardinalidade dos relacionamentos criados no diagrama de classes da aplicação;
- *nenhum objeto pode ser agregado de si próprio*: decorrente de regras do modelo TVM;
- *o nome de uma pessoa não pode ultrapassar 50 caracteres*: decorrente da definição de domínio dos atributos definidos no diagrama de classes da aplicação.

Além das restrições implicitamente declaradas, diversas restrições de integridade podem ser necessárias na aplicação exemplo. Essas devem ser explicitamente declaradas através de alguma linguagem ou método de especificação. A seguir são apresentados, em linguagem coloquial, exemplos de restrições deste tipo. Estas são então especificadas e classificadas nas seções seguintes.

1. o salário de um empregado não pode ser menor do que o piso salarial de sua categoria, após três meses de contratação;
2. somente versões consolidadas de projetos terão gasto total definido;
3. o analista coordenador de um projeto deverá participar do grupo de trabalho que o gerencia;
4. todo empregado deve ter um salário definido;
5. é desejado que a duração real de projetos não ultrapasse sua previsão em mais do que 20 dias;
6. deseja-se que, no final do ano de 2005, os salários de todos os funcionários tenham aumentado pelo menos na mesma proporção que o índice de inflação registrado no ano;
7. são permitidos apenas sexos do tipo 'Masculino' ou 'Feminino';
8. após determinada, a duração de um projeto não pode sofrer alteração.

7.4 Especificação das Restrições Exemplo

Esta seção apresenta a especificação das restrições de integridade apresentadas na seção anterior cujas declarações não estão implícitas na modelagem de dados. As especificações são baseadas na linguagem TVCL, considerando a definição do diagrama de classes da aplicação exemplo. Seguem as especificações de restrições:

1. o salário de um empregado não pode ser menor do que o piso salarial de sua categoria, após três meses de contratação.

```
create TV constraint salarioBase
when { ( incDate(empregado.versions.dataAdmissao,month,3.0) >= now
         evaluate monthly ) }
   if exists( select e.tvoid
                 from empregado.versions e
                 where incDate(e.dataAdmissao,month,3.0) >= now and
                       e.salario.valor < e.salarioBase.valor )
   then ( update empregado
          set salario.valor = salarioBase.valor
          where tvoid in (select tvoid from inconsistentObjects) )
versioning scope {workingVersions}
```

Dessa forma, a restrição especificada será avaliada sempre que o período de experiência de três meses de algum empregado terminar, considerando que a expressão temporal seja verificada mensalmente. Para isso, são considerados apenas os dados atuais de versões em trabalho de objetos das classes *empregado* e *nivelSalarial*. A ação de violação indica que quando violada a restrição, o valor de salário atual das versões de objetos que violaram a restrição receberão automaticamente o valor do salário base de sua categoria. Por fim, devido à possibilidade de alteração no período de experiência de empregados (três meses), essa restrição é definida como temporal de versões a fim de guardar seu histórico de alterações.

2. somente versões consolidadas de projetos terão gasto total definido.

```
create constraint gastoTotal
when { on ever update projeto.versions.gastoTotal }
   if exists( select ever p.tvoid
                 from projeto.versions p
                 where (not p.isconsolidated) and
                        p.gastoTotal is not null )
   then rollback
```

Dessa forma, a restrição é verificada em caráter postergado sempre que o valor do gasto total de algum projeto for atualizado. São consideradas todas as versões de objetos da classe *projeto* e alterações no passado, no presente e no futuro. Por fim, caso ocorra uma violação, a transação que a causou deve ser desfeita.

3. o analista coordenador de um projeto deverá participar do grupo de trabalho que o gerencia.

```
create TV constraint analistaCoordenador
   when { immediately on update analista.versions.participaDe ,
          immediately on update analista.versions.coordena ,
          immediately on update grupoTrabalho.versions.gerencia }
      if exists( select p.tvoid
                    from projeto.versions p, grupoTrabalho.versions g,
                         analista.versions a
                    where p.coordenadoPor.tvoid = a.tvoid and
                          p.gerenciadoPor.tvoid = g.tvoid and
                          g not in a.participaDe )
      then rollback
validity [31/03/2005..]
```

A restrição especificada será verificada em caráter imediato sempre que houver alguma atualização sobre os relacionamentos *participaDe*, *coordena* ou *gerencia* atualmente válidos em qualquer versão de objetos das classes *analista* e *grupoTrabalho*. Caso ocorra uma violação, a transação em questão deve ser desfeita. Além disso, a restrição é definida como temporal de versões a fim de possibilitar que sua validade se inicie apenas no dia 31 de março de 2005.

4. todo empregado deve ter um salário definido.

```
create constraint salario
when { on ever change empregado,
       on ever change nivelSalarial }
   if exists( select ever e.tvoid
                 from empregado e
                 where ( e.salario.tvoid is null or
                         e.salario.valor is null))
   then rollback
```

A restrição em questão será verificada em caráter postergado sempre que houver alguma alteração de dados válidos no presente, no passado ou no futuro das versões correntes de objetos das classes *empregado* ou *nivelSalarial*. Além disso, as transações causadoras de violações deverão ser desfeitas.

5. é desejado que a duração real de projetos não ultrapasse sua previsão em mais do que vinte dias.

```
create TV constraint conclusao
   when { immediately on insert projeto,
          immediately on update projeto.versions }
      if exists( select p.tvoid
                 from projeto.versions p
                 where p.percConclusao = 100 and
                       incDate(p.inicio,day,p.duracaoDias) >
                       incDate(p.previsaoConclusao,day,20.0))then alert
validity [02/02/1999..]
```

Nesse caso, a restrição especificada deve ser verificada em caráter imediato sempre que houver alguma inserção na classe *projeto* ou atualização de qualquer uma das versões de objetos dessa classe. O estado atual de todas as versões de objetos é considerado. Quando violada, a restrição não impede o armazenamento dos dados, pois é apenas disparado um alerta ao administrador da base. Além disso, a mesma é válida desde o dia 2 de fevereiro de 1999.

6. deseja-se que, no final do ano de 2005, os salários de todos os funcionários tenham aumentado pelo menos na mesma proporção que o índice de inflação registrado no ano.

```
create TV constraint aumentoSalario
when { ( 31/12/2005 = now evaluate daily ) }
   if exists( select ever e.tvoid
              from empregado e, empregado e1
              where e.CPF = e1.CPF and
                    01/01/2005 into e.salario.vinteral and
                    31/12/2005 into e1.salario.vinteral and
                    e1.salario.valor < (e.salario.valor * 1.1) )
   then alert
validity [01/01/2005..31/12/2005]
versioning scope {currentVersions}
```

Considerando um índice de inflação de 10 pontos percentuais, a restrição acima será verificada apenas no dia 31 de dezembro de 2005 e irá considerar as versões correntes de objetos da classe *empregado*. Sua expressão temporal será verificada diariamente. Assim como o exemplo de restrição anterior, essa restrição não impedirá o armazenamento de dados, pois apenas alertará o administrador da base sobre sua violação. Por fim, sua validade é referente ao ano de 2005.

7. são permitidos apenas sexos do tipo ‘Masculino’ ou ‘Feminino’.

```
create constraint tipoSexo
when { immediately on update sexo.descricao }
   if exists( select s.tvoid
              from sexo s
              where s.descricao <> 'Masculino' and
                    s.descricao <> 'Feminino'  ) then rollback
```

A restrição em questão afeta apenas objetos da classe *sexo*. Considerando que essa classe não possui características de tempo e de versão, somente dados atuais de versões únicas de objetos são considerados. Apesar de possível, não existe controle de temporalidade e versionamento da restrição, pois a mesma é considerada estável. A

verificação será feita em caráter imediato sempre que a descrição de algum objeto da classe *sexo* for atualizada. Por fim, caso ocorra uma violação, a transação em questão deve ser desfeita.

8. após determinada, a duração de um projeto não pode sofrer alteração.

```
create TV constraint duracaoAlterada
   when { immediately on update projeto.versions.duracaoDias }
      if exists( select p1.tvoid
                 from projeto.versions p, projeto.versions p1
                 where p1.tvoid = p1.lastversion and
                       p1.isSucessorOf(p) and
                       p.duracaoDias <> p1.duracaoDias and
                       p.duracaoDias is not null )
      then ( delete from projeto
             where tvoid in (select tvoid from inconsistentobjects) )
versioning scope {workingVersions}
```

Essa restrição considera o fato de que uma alteração sobre o atributo não temporal *duracaoDias* cria uma nova versão de objeto da classe *projeto*. Sua verificação é efetuada sempre que ocorrer alguma atualização sobre a duração de versões em trabalho de objetos da classe *projeto*. Caso a restrição seja violada, as versões de projeto com a duração alterada são removidas. Vale notar que, esta mesma restrição poderia ser especificada ordenando que as transações causadoras de violação fossem desfeitas (*rollback*).

7.5 Classificação das Restrições Exemplo

Com base na classificação de restrições de integridade proposta no Capítulo 4, as restrições especificadas na seção anterior são classificadas. Além de sua especificação, a classificação a seguir considera características do diagrama de classes da aplicação exemplo e características do modelo de dados utilizado (TVM).

Antes da classificação das restrições, é necessária uma análise sobre os integrantes de seus conjuntos restringido e restringente. O resultado dessa análise é apresentado na Tabela 7.1 que ilustra os integrantes destes conjuntos relacionados a cada uma das restrições especificadas. Vale notar que o conjunto restringente de uma restrição de integridade pode ser composto por entidades ou relacionamentos, definidos no diagrama de classes, ou valores constantes. O conjunto restringido pode possuir apenas entidades ou relacionamentos.

Tabela 7.1: Conjuntos restringido e restringente das restrições especificadas

	Restrição 1	**Restrição 2**	**Restrição 3**	**Restrição 4**
Conjunto Restringido	{salario / relacionadoA^{r}}	{projeto^{e}}	{coordena / coordenadoPor^{r}}	{salario / relacionadoA^{r}}
Conjunto Restringente	{salarioBase / relacionadoA^{r}, nivelSalarial^{e}, empregado^{e}, 3^{c}}	{projeto^{e}}	{gerencia / gerenciadoPor^{r}, participaDe / possui^{r}}	{nivelSalarial^{e}}
	Restrição 5	**Restrição 6**	**Restrição 7**	**Restrição 8**
Conjunto Restringido	{projeto^{e}}	{salario / relacionadoA^{r}}	{sexo^{e}}	{projeto^{e}}
Conjunto Restringente	{projeto^{e}, 20^{c}}	{empregado^{e}, nivelSalarial^{e}, 10^{c}}	{sexo^{e}, 'Masculino'^{c}, 'Feminino'^{c}}	{projeto^{e}}

r – Relacionamento e – Entidade c – Valor Constante

Depois de apresentados os integrantes de seus conjuntos restringido e restringente, as restrições especificadas são classificadas através da Tabela 7.2. As linhas dessa tabela representam os aspectos e critérios definidos para a classificação de restrições, enquanto colunas representam as restrições de integridade especificadas anteriormente. O conteúdo das células representa as classes em que cada restrição se enquadra e o caminho percorrido (sub-critérios) na árvore que representa cada critério.

Tabela 7.2: Classificação das restrições especificadas

Aspecto	Critério	Restrição 1	Restrição 2	Restrição 3	Restrição 4
Origem	**-**	Ambiente / Legal	Empresa	Empresa	Empresa
Substância	**Abrangência de Estados**	Estática	Dinâmica / Não de Transição	Estática	Dinâmica / Não de Transição
	Tipo do Restringente	Heterogêneo	Homogêneo / Entidade	Homogêneo / Relacionamento	Homogêneo / Entidade
	Tipo do Restringido	Relacionamento	Entidade	Relacionamento	Relacionamento
	Formação do Restringente	Heterogênea	Homogênea / Primitiva	Homogênea / Primitiva	Homogênea / Primitiva
	Formação do Restringido	Primitiva	Primitiva	Primitiva	Primitiva
	Formação da Condição Restritiva	Composta / Conjuntiva	Composta / Conjuntiva	Composta / Conjuntiva	Composta / Disjuntiva
	Aspecto Restringente	Homogêneo / Composição Interna	Homogêneo / Versionamento	Homogêneo / Composição Interna	Homogêneo / Composição Interna
	Aspecto Restringido	Composição Interna	Composição Interna	Composição Interna	Composição Interna
	Propósito	Causativo	Condicional	Condicional	Condicional
	Abrangência do Restringente	Heterogênea	Homogênea / Local / Local a Domínios / Intra-Domínio	Homogênea / Local / Local a Relacionamentos / Inter-Relacionamentos	Homogênea / Local / Local a Domínios / Intra-Domínio
	Abrangência do Restringido	Local / Local a Relacionamentos / Intra-Relacionamento	Local / Local a Domínios / Intra-Domínio	Local / Local a Relacionamentos / Intra-Relacionamento	Local / Local a Relacionamentos / Intra-Relacionamento
	Abrangência de Ligação do Restringente	Heterogênea	Homogênea / Local / Local a Versionamentos / Intra-Versionamento	Homogênea / Local / Local a Relacionamentos / Inter-Relacionamentos	Local / Local a Atributos / Inter-Atributos
	Abrangência de Ligação do Restringido	Local / Local a Relacionamentos / Intra-Relacionamento	Local / Local a Atributos / Intra-Atributo	Local / Local a Relacionamentos / Intra-Relacionamento	Local / Local a Relacionamentos / Intra-Relacionamento
	Completude	Completa	Completa	Completa	Completa
Especificação	**Declaração**	Explícita / Declarativa	Explícita / Declarativa	Explícita / Declarativa	Explícita / Declarativa
	Temporalidade da Restrição	Tempo com Determinação / Bitemporal	Não Temporal	Tempo com Determinação / Bitemporal	Não Temporal
	Abrangência de Temporalidade da Restrição	Global	Global	Local / Futuro / Intervalo Temporal	Global
	Versionamento da Restrição	Versionado	Não-Versionado	Versionado	Não-Versionado
Aplicação	**Dependência sobre Regras Dedutivas**	Indeterminada	Indeterminada	Indeterminada	Indeterminada

Tabela 7.2: Classificação das restrições especificadas (cont.)

Aspecto	Critério	Restrição 1	Restrição 2	Restrição 3	Restrição 4
	Ordem de Precedência	Livre	Livre	Livre	Livre
	Ativação e Desativação	Restrita / Homogênea / Restrita à Existência / Relacionamento	Restrita / Homogênea / Restrita à Existência / Valor de Atributo	Restrita / Homogênea / Restrita à Existência / Relacionamento	Heterogênea
	Tipo de Ativação e Desativação	Independente	Independente	Independente	Independente
	Acionamento	Homogêneo / Automático	Homogêneo / Automático	Homogêneo / Automático	Homogêneo / Automático
	Tratamento	Complementação	Reconstituição	Reconstituição	Reconstituição
	Veículo	Homogêneo / Sistema Gerenciador de Bancos de Dados	Homogêneo / Sistema Gerenciador de Bancos de Dados	Homogêneo / Sistema Gerenciador de Bancos de Dados	Homogêneo / Sistema Gerenciador de Bancos de Dados
	Inspeção	Heterogênea	Heterogênea	Heterogênea	Homogênea / Direta
	Ponto de Verificação	Homogêneo / Baseado em Evento / Externo / Temporal	Homogêneo / Baseado em Operação / Protelado	Homogêneo / Baseado em Operação / Imediato	Homogêneo / Baseado em Operação / Protelado
Temporalidade	**-**	Temporal	Temporal	Temporal	Temporal
	Tipo de Tempo do Restringente	Heterogêneo	Homogêneo / Tempo com Determinação / Bitemporal	Homogêneo / Tempo com Determinação / Bitemporal	Homogêneo / Tempo com Determinação / Bitemporal
	Tipo de Tempo do Restringido	Tempo com Determinação / Bitemporal	Tempo com Determinação / Bitemporal	Tempo com Determinação / Bitemporal	Tempo com Determinação / Bitemporal
	Referência	Homogênea / Tempo Relativo / Relativo a Tempo Pré-definido	Homogênea / Não Temporal	Homogênea / Não Temporal	Homogênea / Não Temporal
	Abrangência Temporal do Restringente	Heterogênea	Homogênea / Global	Homogênea / Local / Futuro / Intervalo Temporal	Homogênea / Global
	Abrangência Temporal do Restringido	Local / Ponto no Tempo Presente	Global	Local / Futuro / Intervalo Temporal	Global
	Granularidade Temporal	Heterogênea	Homogênea	Homogênea	Homogênea
Versionamento	**-**	Versionado	Versionado	Não Versionado	Versionado
	Tipo de Versionamento do Restringente	Heterogêneo	Homogêneo / Versionado	-	Homogêneo / Versionado
	Tipo de Versionamento do Restringido	Não Versionado	Versionado	-	Não Versionado
	Abrangência de Versionamento do Restringente	Heterogênea	Homogênea / Global	-	Global
	Abrangência de Versionamento do Restringido	Local / Inter-Versões / Homogênea entre Versões / Versões não Configuradas / Versões em Trabalho	Global	-	Global

Tabela 7.2: Classificação das restrições especificadas (cont.)

Aspecto	Critério	Restrição 5	Restrição 6	Restrição 7	Restrição 8
Origem	**-**	Empresa	Empresa	Ambiente / Natural	Empresa
Substância	**Abrangência de Estados**	Estática	Dinâmica / Não de Transição	Estática	Estática
	Tipo do Restringente	Heterogêneo	Heterogêneo	Heterogêneo	Homogêneo / Entidade
	Tipo do Restringido	Entidade	Relacionamento	Entidade	Entidade
	Formação do Restringente	Heterogênea	Heterogênea	Heterogênea	Homogênea / Primitiva
	Formação do Restringido	Primitiva	Primitiva	Primitiva	Primitiva
	Formação da Condição Restritiva	Composta / Conjuntiva	Composta / Conjuntiva	Composta / Conjuntiva	Composta / Conjuntiva
	Aspecto Restringente	Homogêneo / Composição Interna	Heterogêneo	Homogêneo / Composição Interna	Heterogêneo
	Aspecto Restringido	Composição Interna	Composição Interna	Composição Interna	Composição Interna
	Propósito	Verificativo	Verificativo	Condicional	Causativo
	Abrangência do Restringente	Heterogênea	Heterogênea	Heterogênea	Homogênea / Local / Local a Domínios / Intra-Domínio
	Abrangência do Restringido	Local / Local a Domínios / Intra-Domínio	Local / Local a Relacionamentos / Intra-Relacionamento	Local / Local a Domínios / Intra-Domínio	Local / Local a Domínios / Intra-Domínio
	Abrangência de Ligação do Restringente	Heterogênea	Heterogênea	Heterogênea	Heterogênea
	Abrangência de Ligação do Restringido	Local / Local a Atributos / Intra-Atributo	Local / Local a Relacionamentos / Intra-Relacionamento	Local / Local a Atributos / Intra-Atributo	Local / Local a Atributos / Intra-Atributo
	Completude	Completa	Completa	Completa	Completa
Especificação	**Declaração**	Explícita / Declarativa	Explícita / Declarativa	Explícita / Declarativa	Explícita / Declarativa
	Temporalidade da Restrição	Tempo com Determinação / Bitemporal	Tempo com Determinação / Bitemporal	Não Temporal	Não Temporal
	Abrangência de Temporalidade da Restrição	Local / Localmente Heterogênea / Intervalo Temporal	Local / Localmente Heterogênea / Intervalo Temporal	Global	Global
	Versionamento da Restrição	Versionado	Versionado	Não Versionado	Não-Versionado
Aplicação	**Dependência sobre Regras Dedutivas**	Indeterminada	Indeterminada	Indeterminada	Indeterminada
	Ordem de Precedência	Livre	Livre	Livre	Livre
	Ativação e Desativação	Restrita / Homogênea / Restrita à Existência / Valor de Atributo	Restrita / Homogênea / Restrita à Existência / Objeto	Restrita / Homogênea / Restrita à Existência / Objeto	Restrita / Homogênea / Restrita à Existência / Valor de Atributo
	Tipo de Ativação e Desativação	Independente	Independente	Independente	Independente

Tabela 7.2: Classificação das restrições especificadas (cont.)

Aspecto	Critério	Restrição 5	Restrição 6	Restrição 7	Restrição 8
	Acionamento	Homogêneo / Automático	Homogêneo / Automático	Homogêneo / Automático	Homogêneo / Automático
	Tratamento	Detecção	Detecção	Reconstituição	Complementação
	Veículo	Homogêneo / Sistema Gerenciador de Bancos de Dados	Homogêneo / Sistema Gerenciador de Bancos de Dados	Homogêneo / Sistema Gerenciador de Bancos de Dados	Homogêneo / Sistema Gerenciador de Bancos de Dados
	Inspeção	Heterogênea	Heterogênea	Homogênea / Direta	Heterogênea
	Ponto de Verificação	Homogêneo / Baseado em Operação / Imediato	Homogêneo / Baseado em Evento / Externo / Temporal	Homogêneo / Baseado em Operação / Imediato	Homogêneo / Baseado em Operação / Imediato
Temporalidade	**-**	Temporal	Temporal	Não Temporal	Temporal
	Tipo de Tempo do Restringente	Heterogêneo	Heterogêneo	-	Homogêneo / Tempo com Determinação / Bitemporal
	Tipo de Tempo do Restringido	Tempo com Determinação / Bitemporal	Tempo com Determinação / Bitemporal	-	Tempo com Determinação / Bitemporal
	Referência	Homogênea / Não Temporal	Heterogênea	-	Homogênea / Não Temporal
	Abrangência Temporal do Restringente	Heterogênea	Heterogênea	-	Homogênea / Global
	Abrangência Temporal do Restringido	Local / Localmente Heterogênea / Intervalo Temporal	Local / Localmente Heterogênea / Intervalo Temporal	-	Global
	Granularidade Temporal	Heterogênea	Homogênea	-	Homogênea
Versionamento	**-**	Versionado	Versionado	Não Versionado	Versionado
	Tipo de Versionamento do Restringente	Heterogêneo	Heterogêneo	-	Homogêneo / Versionado
	Tipo de Versionamento do Restringido	Versionado	Não Versionado	-	Versionado
	Abrangência de Versionamento do Restringente	Heterogênea	Homogênea / Global	-	Heterogênea
	Abrangência de Versionamento do Restringido	Global	Global	-	Local / Inter-Versões / Homogênea entre Versões / Versões não Configuradas / Versões em Trabalho

7.6 Considerações Finais

Este capítulo apresentou um estudo de caso baseado na classificação de restrições de integridade, proposta no Capítulo 4, e na linguagem de especificação TVCL, apresentada no Capítulo 6.

Foi descrita a realidade de uma empresa fictícia com a necessidade de desenvolvimento de uma aplicação. A partir desse exemplo, foram apresentados, em linguagem coloquial, exemplos de restrições aplicáveis sobre a base de dados da aplicação exemplo. Dentre as restrições, estão restrições implícitas na modelagem de

dados e restrições que devem ser explicitamente declaradas. Restrições do segundo conjunto foram especificadas através da linguagem TVCL e classificadas com base na classificação proposta.

Este estudo demonstra de forma prática a usabilidade da linguagem TVCL e da classificação de restrições. Dessa forma, restrições de integridade sobre bases de dados baseadas no modelo TVM podem ser especificadas de forma não ambígua. Além disso, através da análise propiciada pela classificação, métodos distintos para a manutenção de restrições podem ser criados de acordo com características das próprias restrições.

8 CONCLUSÕES E TRABALHOS FUTUROS

Esta dissertação apresentou um estudo sobre restrições de integridade em bases de dados com suporte a tempo e versões. Restrições foram analisadas em detalhes através de uma classificação que considera os aspectos origem, substância, especificação, declaração, temporalidade e versionamento. A abrangência da classificação foi analisada através da comparação com diversas outras classificações encontradas na literatura. Também foi apresentada uma linguagem de especificação de restrições sobre bancos de dados baseados no modelo TVM. Com base na classificação de restrições definida, o poder de expressão desta linguagem foi analisado. Por fim, um estudo de caso demonstrou a utilização prática da classificação e da linguagem de especificação de restrições.

A Tabela 8.1 apresenta as principais características deste trabalho, considerando os mesmos critérios utilizados na Tabela 3.1, que ilustra uma comparação entre todos os trabalhos relacionados apresentados no Capítulo 3. Acredita-se que, dessa forma, a comparação entre o presente trabalho e os trabalhos relacionados se torne mais fácil.

Tabela 8.1: Análise do trabalho

Características	Trabalho
Público Alvo	Geral
Forma de Apresentação da Classificação	Explícita
Modelo de Dados	Orientado a Objetos
Tipo de Banco de Dados	Temporal de Versões
Método de Especificação de Restrições	Linguagem TVCL
Método de Manutenção de Restrições	ND
Linguagem de Atualização de Dados	Linguagem TVL/SE (parte de atualização de objetos)
Otimização do Processo de Manutenção	ND
Técnica de Propagação de Mudanças em Restrições	Baseada na manutenção automática da temporalidade e do versionamento das próprias restrições.
Implementação	ND

ND – Não Definido

8.1 Principais Contribuições

Acredita-se que uma das principais contribuições do trabalho seja a classificação de restrições de integridade. Através dessa classificação é possível uma análise profunda de diversas características inerentes a restrições de integridade com tempo e versões. É importante enfatizar que, apesar do fato que a classificação de restrições possa ser utilizada de diversas maneiras, seu principal alvo é o projetista do SGBD, pois com base na classificação, torna-se possível a criação de técnicas distintas para a manutenção de restrições com características comuns, de forma a se obter vantagens sobre elas.

Através da análise de abrangência da classificação, foi demonstrado que ela é mais completa do que outras encontradas na literatura, pois considera os aspectos

considerados por elas além de aspectos não considerados anteriormente. Assim, acredita-se que a classificação proposta possa servir de base para a manutenção da integridade de bancos de dados com tempo e versões.

A classificação pode ainda ser adaptada para bases de dados tradicionais, somente temporais ou somente de versões, visto que os critérios relacionados aos conceitos de tempo e de versão são agrupados em aspectos específicos.

A linguagem de especificação de restrições TVCL é considerada outra das principais contribuições do presente trabalho. Através dessa linguagem torna-se possível a especificação não ambígua de restrições de integridade em bases de dados definidas sobre o modelo TVM. Essa linguagem considera a temporalidade e o versionamento dos dados envolvidos além da possibilidade de aplicação destes conceitos sobre as próprias restrições. Por fim, através da análise do poder de expressão da linguagem, é possível saber quais classes de restrições podem e quais não podem ser especificadas.

Como principal produção científica do presente trabalho, foram publicados os seguintes artigos:

- Cordeiro, Robson L. F.; Edelweiss, Nina; Galante, Renata de M.; Santos, Clesio S. dos. **TVCL – Temporal Versioned Constraint Language.** Proc. 20th Brazilian Symposium on Databases – SBBD 2005, Uberlândia, October 2005.
- Cordeiro, Robson L. F.; Santos, Clesio S. dos; Edelweiss, Nina; Galante, Renata de M. **Classificação de Restrições de Integridade em Bancos de Dados Temporais de Versões.** Proc. 19th Brazilian Symposium on Databases – SBBD 2004, Brasília, October 2004.
- Cordeiro, Robson L. F.; Santos, Clesio S. dos; Edelweiss, Nina **Integrity Constraints for Temporal Versions Model:** Classification, Modeling and Verification. III Workshop de Teses e Dissertações em Banco de Dados – III WTDBD, Brasília, October 2004.

8.2 Trabalhos Futuros

Diversos trabalhos podem ser realizados com base no material produzido na presente pesquisa. Considera-se que o principal foco para trabalhos futuros seja a manutenção de restrições de integridade classificadas e especificadas através das técnicas propostas.

A maioria dos métodos existentes para a manutenção de restrições de integridade assume a existência de um estado consistente da base de dados antes da ocorrência de uma operação que altere esse estado. A partir disso, esses métodos se concentram nos efeitos relativos à execução da operação em questão, a fim de verificar a consistência do novo estado da base (BÖHLEN, 1994). Esses métodos são conhecidos como métodos incrementais de verificação (*incremental consistency checkers*).

A adaptação de métodos incrementais para a verificação de restrições com a presença de tempo e versões parece bastante óbvia. Entretanto, acredita-se que a utilização desses métodos é possível apenas para a manutenção de restrições com pontos de verificação baseados em operação. A utilização de métodos incrementais para a manutenção de restrições com pontos de verificação baseados em eventos não é possível, pois sua violação independe de uma alteração de estado da base. Isso significa que, um estado considerado consistente em um determinado momento pode se tornar inconsistente após alguns instantes, sem que ocorra qualquer alteração sobre o mesmo.

Devido à possibilidade de falhas humanas durante a utilização de bases de dados, a manutenção de restrições deve ainda considerar a existência de operações de correção sobre alteração indevidas de dados. Essas operações, feitas apenas por usuários

autorizados, podem desrespeitar certas restrições a fim de possibilitar a correção de erros.

Além do desenvolvimento de técnicas para a manutenção de restrições, outras propostas de trabalhos futuros são apresentadas a seguir:

- verificação do minimalismo da classificação - além da verificação da completude da classificação proposta, é considerada importante a verificação de seu minimalismo. Essa análise visa permitir a classificação completa de restrições de integridade com o mínimo possível de aspectos e critérios, minimizando sua complexidade sem que ocorra perda em abrangência;
- especificação de restrições definidas no modelo TVM - diversas restrições definidas, em linguagem coloquial, por Moro (2001) podem ser especificadas através da linguagem TVCL a fim de evitar possíveis ambigüidades;
- analisar novas restrições do modelo TVM - além das restrições definidas para o TVM, novas restrições sobre seus relacionamentos e objetos temporais de versões podem ser definidas a fim de aumentar o grau de confiabilidade do modelo. Uma base para a definição dessas novas restrições pode ser encontrada no trabalho de Elmasri, Kouramajian e Fernando (1993);
- tratar de problemas de restrições em cascata - trabalhos que visam evitar a ocorrência de restrições que se anulem e restrições com a ocorrência de sobreposição de tempos;
- implementação de técnicas para a classificação automática de restrições - o desenvolvimento e técnicas que permitam a classificação automática ou semi-automática de restrições especificadas através da TVCL. A implementação dessas técnicas poderia facilitar grandemente a utilização da classificação como base de processos de manutenção de integridade de dados.

REFERÊNCIAS

AGRAWAL, R. et al. Object versioning in Ode. In: IEEE INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 7., 1991, Kobe. **Proceedings**… Los Alamitos, IEEE Computer Society, 1991. p.446-455.

BEECH, D.; MAHBOD, B. Generalized Version Control in an Object-Oriented Database. In: IEEE INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 4., 1988, Los Angeles. **Proceedings…** Los Angeles: IEEE Computer Society, 1988. p.14-22.

BETTINI, C.; WANG, X. S.; JAJODIA, S. Satisfiability of Quantitative Temporal Constraints with Multiple Granularities. In: INTERNATIONAL CONFERENCE ON PRINCIPLES AND PRACTICE OF CONSTRAINT PROGRAMMING, CP, 3., 1997. **Proceedings…** Linz, Austria:[s.n.], 1997. p. 435-449.

BJÖRNERSTEDT, A.; HULTÉN, C. Version Control in an Object-Oriented Architecture. In: KIM, W.; LOCHOVSKY, F.H. (Ed.). **Object-Oriented Concepts, Databases, and Applications.** New York: ACM Press, 1989. p. 451-485.

BÖHLEN, M. H. **Valid Time Integrity Constraints.** Tucson, AZ: Department of Computer Science – University of Arizona, 1994. (Technical Report 94-30).

BÖHLEN, M. H.; MARTI, R. On the Completeness of Temporal Database Query Languages. In: INTERNATIONAL CONFERENCE ON TEMPORAL LOGIC, 1., 1994. **Proceedings**… [S.l.:s.n.], 1994. p.283-300.

BUCHMANN, A. P.; CARRERA, R. S.; VAZQUEZ-GALINDO, M. A. A Generalized Constraint and Exception handler for an Object-Oriented CAD-DBMS. In: INTERNATIONAL WORKSHOP ON OBJECT-ORIENTED DATABASE SYSTEMS, OODBS, 2., 1986. **Proceedings…** Pacific Grove, California, USA:[s.n.], 1986. p. 38-49.

CAMOLESI, L. J.; TRAINA, C. J. Evolução de Esquemas de Dados: um panorama amplo de aspectos técnicos e gerenciais. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 11., 1996, São Carlos, SP. **Minicursos/Tutoriais. . .** São Carlos: Universidade Federal de São Carlos, 1996. p.1–19.

CASTILHO, J. M. V. de; CASANOVA, M. A.; FURTADO, A. L. A Temporal Framework for Database Specifications. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 8., 1982. **Proceedings…** Mexico City, Mexico:[s.n.], 1982. p. 280-291.

CASTILHO, J. M. V. de. **Especificações Formais e Sistemas de Bancos de Dados.** Buenos Aires: Kapelusz, 1987. 157p.

CASTRO, C. de; GRANDI, F.; SCALAS, M. R. Schema Versioning for Multitemporal Relational Databases. **Information Systems**, [S.l.], v.22, n.5, p.249–290, July 1997.

CATTELL, R.; BARRY, D. K. (Ed.). **The Object Data Standard:** ODMG 3.0. San Francisco: Morgan Kaufmann, 2000. 280p.

CHOMICKI, J. Real-Time Integrity Constraints. In: INTERNATIONAL SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS, ACM SIGACT-SIGMOD-SIGART, 11., 1992. **Proceedings…** San Diego, California, United States:[s.n.], 1992. p. 274-282.

CHOMICKI, J. Efficient Checking of Temporal Integrity Constraints Using Bounded History Encoding. **ACM Transactions on Database Systems**, [S.l.], v.20, n.2, p. 149-186, Jun. 1995

CHOMICKI, J; TOMAN, D. Implementing Temporal Integrity Constraints Using an Active DBMS. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.7, n.4, p. 566-582, Feb. 1995.

CONRADI, R.; WESTFECHTEL, B. Version Models for Software Configuration Management. **ACM Computing Surveys**, New York, v.30, n.2, p.232-282, June 1998.

CORDEIRO, R. L. F. **Otimização de Consultas em Bancos de Dados Temporais de Versões**. 2003. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

COWLEY, W.; PLEXOUSAKIS, D. Temporal Integrity Constraints with Indeterminacy. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 26., 2000. **Proceedings…** Cairo, Egypt:[s.n.], 2000. p. 441-450.

DADAM, P.; LUM, V.; WERNER, H. D. Integration of Time Versions into a Relational Database System. In: CONFERENCE ON VERY LARGE DATABASES, VLDB, 10., 1984, Singapore. **Proceedings…** San Mateo: Morgan Kauffman, 1984. p. 509-522.

DAYAL, U. et al. The HiPAC Project: Combining Active Databases and Timing Constraints. **ACM SIGMOD Record**, [S.l.], v.17, n.1, p. 51-70, Mar. 1988.

DOUCET, A. et al. Integrity Constraints and Versions. In: INTERNATIONAL WORKSHOP ON FOUNDATIONS OF MODELS AND LANGUAGES FOR DATA AND OBJECTS, FMLDO, 6., 1996. **Proceedings…** Schloss Dagstuhl, Germany:[s.n.], 1996. p. 25-39.

DITTRICH, K. R.; LORIE, R. A. A Versions Support for Engineering Database Systems. **IEEE Transactions on Software Engineering**, New York, v.14, n.4, p. 429-437, Apr. 1988.

DOUCET, A. et al. Integrity Constraints in Multiversion Databases. In: BRITISH NATIONAL CONFERENC ON DATABASES, BNCOD, 14., 1996. **Proceedings…** Edinburgh, UK:[s.n.], 1996. p. 56-73. (Lecture Notes in Computer Science, v.1094).

DOUCET, A.; MONTIES, S. Versions of Integrity Constraints in Multiversion Databases. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 8., 1997, Toulouse, France. **Proceedings. . .** Berlin: Springer-Verlag, 1997. p.252–261. (Lecture Notes in Computer Science, v.1308).

DOUCET, A. et al. Using Database Versions to Implement Temporal Integrity Constraints. In: CONSTRAINT DATABASE AND APPLICATIONS, CDB, 2., 1997. **Proceedings…** Delphi, Greece:[s.n.], 1997. p. 219-233.

EDELWEISS, N.; CASTILHO, J. M. V. de; OLIVEIRA, J. P. M. de. A Temporal Logic Language For Temporal Conditions Definition. In: INTERNATIONAL CONFERENCE OF THE CHILEAN COMPUTER SCIENCE SOCIETY, 13., 1993. **Proceedings…** La Serena, Chile:[s.n.], 1993. p. 163-178.

EDELWEISS, N.; OLIVEIRA, J. P. M. de. **Modelagem de Aspectos Temporais de Sistemas de Informação.** Recife: Brasil, 1994. 162p.

ELMASRI, R.; KOURAMAJIAN, V.; FERNANDO, S. Temporal Database Modeling: an object oriented approach. In: INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, CIKM, 2., 1993. **Proceedings…** Washington, DC, USA:[s.n.], 1993. p. 574-585.

ESCOFET, C. M.; SISTAC, J. Integrity Constraints Checking in Historical Deductive Databases. In: INTERNATIONAL WORKSHOP ON THE DEDUCTIVE APPROACH TO INFORMATION SYSTEMS AND DATABASES, DAISD, 5., 1994. **Proceedings…** Aiguablava, Costa Brava, Catalonia:[s.n.], 1994. p. 299-324.

ESCOFET, C. M.; SISTAC, J. An Integrity Constraint Checking Method for Temporal Deductive Databases. In: INTERNATIONAL WORKSHOP ON TEMPORAL REPRESENTATION AND REASONING, TIME, 3., 1996. **Proceedings…** Key West, Florida, USA:[s.n.], 1996.

ESCOFET, C. M.; SISTAC, J. Applying Transition Rules to Bitemporal Deductive Databases for Integrity Constraint Checking. In: INTERNATIONAL WORKSHOP ON LOGIC IN DATABASES, LID, 1., 1996. **Proceedings…** San Miniato, Italy:[s.n.], 1996. p. 117-134.

ESCOFET, C. M. **Analyzing Temporal Integrity Constraints to Obtain the Minimum Number of Transition Rules.** Barcelona, Catalunya: Department of Llenguatges i Sistemes Informàtics – Universitat Politècnica de Catalunya, 2001. (Technical Report LSI-01-52-R).

FREEMAN-BENSON, B. N.; BORNING, A. Integrating Constraints with an Object-Oriented Language. In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, ECOOP, 6., 1992. **Proceedings…** Utrecht, The Netherlands:[s.n.], 1992. p. 268-286.

GALANTE, R. de M.; EDELWEISS, N.; SANTOS, C. S. dos; MOREIRA, A. F. Data Modification Language for Full Support of Temporal Schema Versioning. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 18., 2003. **Anais. . .** [S.l.: s.n.], 2003.

GALANTE, R. de M. **Modelo Temporal de Versionamento com Suporte à Evolução de Esquemas**. 2003. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

GETZ, M.; LIPECK, U. W. "Temporal" Integrity Constraints in Temporal Databases. In: INTERNATIONAL WORKSHOP ON TEMPORAL DATABASES, 2., 1995. **Proceedings…** Zürich, Switzerland:[s.n.], 1995. p. 77-92.

GOLENDZINER, L. **Um Modelo de Versões para Banco de Dados Orientados a Objetos**. 1995. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

GOONETILLAKE, J. S.; CARNDUFF, T. W.; GRAY, W. A. An integrity constraint management framework in engineering design. **Computers in Industry**, [S.l.], v.48, n.1, p. 29-44, May 2002.

JENSEN, C.S. et al. The Consensus Glossary of Temporal Database Concepts - February 1998 Version. In: ETZION, O; JAJODIA, S; SRIPADA, S (Ed.). **Temporal Databases Research and Practice**. Berlin: Springer-Verlag, 1998. p. 367-405.

JENSEN, C. S. **Temporal Database Management.** 1999. Tese (Doutorado em Ciência da Computação) — Department of Computer Science, Aalborg University, Aalborg.

KATZ, R. H. Toward a Unified Framework for Version Modeling in Engineering Databases. **ACM Computing Surveys**, New York, v.22, n.4, p. 375-408, Dec. 1990.

KIM, W.; BERTINO, E.; GARZA, J. F. Composite objects revisted. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, ACM-SIGMOD, Oregon, 1989. **Proceedings…** New York: ACM Press, 1989. p.337-347.

MANOLA, F. A.; DAYAL, U. PDM: An Object-Oriented Data Model. In: INTERNATIONAL WORKSHOP ON OBJECT-ORIENTED DATABASE SYSTEMS, OODBS, 1., 1986, Pacific Grove, California, USA. **Proceedings…** Pacific Grove: IEEE Computer Society, 1986. p.18-25.

MEDEIROS, C. B.; JOMIER, G.; CELLARY, W. **Maintaining Integrity Constraints across Versions in a Database.** Campinas, SP: Departamento de Ciência da Computação – Universidade Estadual de Campinas, 1992. (Relatório Técnico DCC-08/92).

MEIRI, I. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. **Artificial Intelligence**, [S.l.], v.87, n.1-2, p. 343-385, Nov. 1996.

MORO, M.M. **Modelo Temporal de Versões**. 2001. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

MORO, M. M. et al. **Linguagem de Consultas para o Modelo Temporal de Versões**. 2001. Relatótio de Pesquisa (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto alegre.

MORO, M. M.; SAGGIORATO, S. M.; EDELWEISS, N.; SANTOS, C. S. dos. Adding Time to an Object-Oriented Versions Model. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 12., 2001, Munich, Germany. **Proceedings. . .** Berlin: Springer-Verlag, 2001. p.805–814. (Lecture Notes in Computer Science, v.2113).

MORO, M. M.; EDELWEISS, N.; ZAUPA, A. P.; SANTOS, C. S. dos. TVQL - Temporal Versioned Query Language. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 13., 2002, Aix-en-Provence, France. **Proceedings. . .** Berlin: Springer-Verlag, 2002. p.618–627. (Lecture Notes in Computer Science, v.2453).

MYLOPOULOS, J. et al. Telos: representing Knowledge About Information Systems. **ACM Transactions on Information Systems**, [S.l.], v.8, n.4, p. 325-362, Oct. 1990.

NIELSON, H. R.; NIELSON, F. **Semantics with Applications:** a formal introduction. [S.l.]: Wiley Professional Computing, 1992. 252p.

PLEXOUSAKIS, D. Integrity Constraint and Rule Maintenance in Temporal Deductive Knowledge Bases. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 19., 1993. **Proceedings…** Dublin, Ireland:[s.n.], 1993. p. 146-157.

PLEXOUSAKIS, D. Compilation and Simplification of Temporal Integrity Constraints. In: INTERNATIONAL WORKSHOP ON RULES IN DATABASE SYSTEMS, RIDS, 2., 1995. **Proceedings…** Athens, Greece:[s.n.], 1995. p. 260-276.

RAM, D. J. et al. Constraint Meta-Object: A New Object Model for Distributed Collaborative Designing. **IEEE Transactions on Systems, Man, and Cybernetics**, [S.l.], v.27, n.2, p. 208-221, Mar. 1997.

SANTOS, C. S. dos. **Caracterização Sistemática de Restrições de Integridade em Bancos de Dados**. 1980. Tese (Doutorado em Ciências, menção Informática) – Departamento de Informática, PUC, Rio de Janeiro.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database System Concepts.** 4th ed. Boston: McGraw-Hill, 2002. 1064p.

SILVA, M. A. P. e. Dynamic Integrity Constraints Definition and Enforcement in Databases: A Classification Framework. In: INTERNATIONAL CONFERENCE ON INTEGRITY AND INTERNAL CONTROL IN INFORMATION SYSTEMS, IICIS, 1., 1997. **Proceedings…** Zurich, Switzerland:[s.n.], 1997. p. 65-87.

SRINATH, S.; RAMAKRISHNAN, R.; RAM, J. A Generic Model for Semantics-Based Versioning in Projects. **IEEE Transactions on Systems, Man, and Cybernetics**, [S.l.], v.30, n.2, p. 108-122, Mar. 2000.

WEN-CHI, H.; ZHANG, Z. Enhancing Database Correctness: A Statistical Approach. **ACM SIGMOD Record**, San Jose, California, v.24, n.2, p. 223-232, June 1995.

ZAUPA, A. P. **Suporte a Consultas no Ambiente Temporal de Versões**. 2002. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

APÊNDICE BNF DA LINGUAGEM TVCL

A especificação da BNF (Backus Naur Form) obedece a seguinte notação:

- <> para delimitar metavariáveis (símbolos não terminais);
- | para separar duas alternativas;
- [] para itens opcionais;
- {}* para itens opcionais repetitivos (zero ou mais vezes);
- {}+ para itens obrigatórios repetitivos (uma ou mais vezes)
- () para delimitar um conjunto de opções;
- Os símbolos terminais são apresentados em negrito e delimitados por aspas quando compostos por um único caractere.

Vale notar que os elementos não terminais *<TVQLQuery>* e *<TVLSEStatement>* não estão expandidos, pois representam respectivamente consultas da linguagem TVQL, descrita na Seção 2.3.4, e sentenças de modificação de dados da linguagem TVL/SE, descrita na Seção 6.4.1. A BNF completa dessas linguagens pode ser encontrada em (MORO et al., 2001; GALANTE, 2003)

```
<Constraint> ::= ( create constraint <constraintName> <constraintMiddle> |
   create TV constraint <constraintName> <constraintMiddle>
   [<validity>] [<versioningScope>] )
<constraintMiddle> ::= when  "{" <verifyingPoints> "}"
   if exists "(" <TVQLQuery> ")"
   then ( rollback | alert |
   "(" <TVLSEStatement> { "," <TVLSEStatement>}* ")" )
   [follows "{" <constraintName> {"," <constraintName>}* "}"]
   [precedes "{" <constraintName> {"," <constraintName>}* "}" ]
<validity> ::= validity <timeInterval>
<versioningScope> ::= versioning scope
   (allVersions | "{" <versionState> {"," <versionState>}* "}")
<versionState> ::= ( stableVersions | workingVersions | consolidatedVersions
   | deactivatedVersions | configuredVersions | currentVersions )
<verifyingPoints> ::= <verifyingPoint> { "," <verifyingPoint> }*
<verifyingPoint> ::= ( <operationBased> | <eventBased> )
<operationBased> ::= [immediately] on [ever] <operarion>
<operation> ::= ( insert <className> |
   (update | delete | change) <className> [.versions] |
   update <className> [.versions] "." (<TVMRelationship> | <TVMAttribute>) )
<eventBased> ::= "(" [ever] <temporalExpression> evaluate <temporalUnit> ")"
<temporalUnit> ::= ( secondly | minutely | hourly | daily | monthly | yearly )
<temporalExpression> ::= <logicalTerm> <LO>
<LO> ::= { or <temporalExpression> <LO> }*
<logicalTerm> ::= <logicalFactor> <LT>
<LT> ::= { and <logicalTerm> <LT> }*
<logicalFactor> ::= [ not ] <logicalElement>
<logicalElement> ::= ( "(" <temporalExpression> ")"
   | PRESENT "(" <temporalExpression> ")"
   | <tInstantExpression>
   | <tIntervalExpression> )
```

```
<tInstantExpression> ::= <tInstantTerm> <comparisonOp> <tInstantTerm>
<tIntervalExpression> ::= ( <tIntervalTerm> <intervalOp> <tIntervalTerm>
   | <mixTerm> <mixOp> <tIntervalTerm> )
<tInstantTerm> ::= ( <propertyName> | <instantValue>
   | <preDefInstant> | <preDefLifeTime> | <instantFunction>)
<tIntervalTerm> ::= ( <preDefInterval> | <timeInterval> )
<mixTerm> ::= ( <tIntervalTerm> | <tInstantTerm> )
<comparisonOp> ::= ( "><" | "<" | >= | <= | "=" | <> )
<intervalOp> ::= ( intersect | overlap | equal )
<mixOp> ::= ( after | into | before )
<propertyName> ::= <head> "." ( <TVMAttribute> | <TVMMethod> )
<head> ::= <className> [.versions] [ "." ( <TVMRelationship> | <TVMMethod> ) ]
<preDefInstant> ::= <propertyName> "." <instants>
<instants> ::= ( tiInstant | tfInstant | viInstant | vfInstant )
<preDefLifeTime> ::= <head> "." <lifeTimes>
<lifeTimes> ::= ( iLifetime | fLifetime )
<preDefInterval> ::= <propertyName> "." <intervals>
<intervals> ::= ( tInterval | vInterval )
<instantFunction> ::= ( incDate( <tInstantTerm> ","
   <granularity> "," <floatValue> ")" |
   decDate( <tInstantTerm> "," <granularity> "," <floatValue> ")" )
<granularity> ::= ( second | minute | hour | day |
   week | month | year | decade | century )
<TVMMethod> ::= <identifier> "(" [<params>] ")"
<params> ::= <identifier> {"," <identifier>}*
<constraintName> ::= <identifier>
<TVMRelationship> ::= <identifier>
<TVMAttribute> ::= <identifier>
<className> ::= <identifier>
<timeInterval> ::= ( "[" <initialInstant> .. [<finalInstant>] "]" |
   "[" .. <finalInstant> "]" )
<initialInstant> ::= <instantValue>
<finalInstant> ::= <instantValue>
<instantValue> ::= ( <dateValue> [ "," <hourValue>] | now )
<dateValue> ::= <number> "/" <number> "/" <number> <number>
<hourValue> ::= <number> ":" <number>
<floatValue> ::= <digit> {<digit>}* "." <digit> {<digit>}*
<number> ::= <digit> <digit>
<identifier> ::= ( <letter> {<ID>}* | "_" {<ID>}* )
<ID> ::= ( <letter> | <digit> | "_" )
<digit> ::= ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" )
<letter> ::= ( "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L"
   | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y"
   | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l"
   | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y"
   | "z" )
```