ADRIANA PEREIRA DE MEDEIROS

ESPECIFICAÇÃO DECLARATIVA E IMPLEMENTAÇÃO DE APLICAÇÕES HIPERMÍDIA NA WEB

DISSERTAÇÃO DE MESTRADO

Departamento de Informática

Rio de Janeiro, 08 de junho de 2001

ADRIANA PEREIRA DE MEDEIROS

ESPECIFICAÇÃO DECLARATIVA E IMPLEMENTAÇÃO DE APLICAÇÕES HIPERMÍDIA NA WEB

Dissertação apresentada ao Departamento de Informática da PUC-RJ como parte dos requisitos para a obtenção do título de Mestre em Informática: Ciência da Computação.

Orientador: Daniel Schwabe.

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 08 de junho de 2001.

Este trabalho é dedicado a Deus e aos meus queridos pais, Maria de Lourdes e Manoel, por tudo o que eles representam em minha vida.

Meus agradecimentos

- a Deus, pela luz divina que ilumina todos os dias da minha vida;
- aos meus pais, irmãos e afilhado, Lourdes, Manoel, Alciléa, Alcimar e Vitor Gabriel, pela compreensão, pelo amor e carinho de sempre;
- ao meu orientador, Prof. Daniel Schwabe, pelo conhecimento, pela atenção, confiança, dedicação e amizade;
- ao Nilton, pela compreensão, pela atenção e pelo carinho constante;
- aos amigos Mark Douglas, Fernanda Lima, Laufer, Angela, Patrícia Vilain, Natacha, Geiza, Bazilio e Patrícia, pela ajuda, pela alegria e companheirismo de todos os dias;
- ao Prof. Bruno Feijó, pela amizade e contribuição para a continuidade desse trabalho;
- aos inesquecíveis professores e amigos, Jorge Domingos, Renato Doria, José Helayel, Alexandre Lucas e Tavares, pelo incentivo e força para a realização desse trabalho;
- aos funcionários do Departamento de Informática da PUC-Rio, pela colaboração;
- ao CNPq, pelo apoio financeiro fornecido durante o curso.

Resumo

O desenvolvimento de uma aplicação hipermídia compreende duas etapas principais: a especificação da aplicação por um método de projeto (por exemplo, OOHDM) e sua implementação em alguma linguagem (utilizando um ambiente de suporte).

Geralmente, a especificação do projeto de uma aplicação hipermídia é composta por modelos, cujas informações devem ser mapeadas para o ambiente de implementação. Em muitos ambientes de implementação, esse mapeamento é realizado de forma manual pela tradução dos modelos para o código da aplicação.

Este trabalho apresenta a linguagem declarativa OOHDM-ML para especificação de aplicações hipermídia, projetadas de acordo com as primitivas do método OOHDM, e o ambiente OOHDM-XWeb, criado para apoiar a implementação dessas aplicações. Este ambiente tem como principal objetivo gerar de forma automática a descrição do projeto OOHDM de uma aplicação hipermídia em um formato próprio para o ambiente de implementação OOHDM-Web 2.0, a partir de sua especificação OOHDM-ML. Essa descrição é gerada por uma folha de estilo chamada OOHDM-Translation implementada em XSLT, uma linguagem de transformação para documentos XML.

Abstract

Hypermedia application development is composed of two main steps: the application specification by a design method (such as OOHDM) and its implementation in a programming language (using a support environment).

The design specification of a hypermedia application is generally composed of models whose information must be mapped onto the implementation environment primitives. In many environments this mapping is done through the manual translation of the models into application code .

This paper presents the declarative language OOHDM-ML, allowing the specification of hypermedia applications designed with OOHDM, and the OOHDM-XWeb environment, created to support the implementation of these applications. The main objective of this environment is to automatically generate the OOHDM design description of a hypermedia application into the format required by the OOHDM-Web 2.0 environment, using its OOHDM-ML specification. This description is generated by the OOHDM-Translation stylesheet implemented in XSLT, a transformation language for XML documents.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Motivação

O desenvolvimento de uma aplicação hipermídia compreende duas etapas principais: a especificação da aplicação através de um método de projeto (por exemplo, OOHDM [Rossi 1996] [Schwabe 1998]) e sua implementação em alguma linguagem (através de um ambiente de suporte).

Geralmente, a especificação do projeto de uma aplicação hipermídia é composta por modelos, cujas informações devem ser mapeadas para o ambiente de implementação. Em muitos ambientes de implementação, esse mapeamento é realizado de forma manual através da tradução dos modelos para o código da aplicação.

A utilização de uma linguagem declarativa para especificar o projeto de uma aplicação hipermídia, traz algumas vantagens ao processo de desenvolvimento web.

Linguagens declarativas podem possuir uma sintaxe simples, são compreensíveis ao ser humano e fáceis de processar pela máquina. Se definidas no nível adequado de abstração, elas permitem que o projetista concentre-se no processo de autoria da aplicação, sem precisar se preocupar com os aspectos relacionados à implementação. Além disso, o processo de desenvolvimento da aplicação torna-se mais seguro, pois a especificação criada pode ser validada por um conjunto de regras pré-definidas e mapeada automaticamente para um ambiente de implementação, utilizando-se um processador.

Atualmente, já existem algumas linguagens declarativas para a especificação do projeto de aplicações hipermídia, baseado em algum modelo ou método de projeto, como por exemplo a linguagem WebML. No entanto, essas linguagens tratam de aspectos relacionados a implementação que, muitas vezes, fazem com que o projetista antecipe decisões de projeto, que podem dificultar o processo de autoria da aplicação.

Existem, atualmente, diversos métodos e modelos para a especificação de aplicações hipermídia tais como HDM [Garzotto 1993], RMM [Isakowitz 1995], OOHDM e o EORM [Lange 1994]. O método OOHDM (Object Oriented Hypermedia Design Method) tem se mostrado o método mais maduro, devido a sua ampla utilização pela comunidade hipermídia para projetos de aplicações em diversos países. Assim, o método utilizado neste trabalho para a especificação de aplicações hipermídia será o OOHDM.

Inicialmente, as aplicações hipermídia projetadas com o método OOHDM não possuíam uma especificação declarativa que permitisse mapear as informações de projeto da aplicação diretamente para um ambiente de implementação. Este mapeamento era realizado de forma manual através de consultas visuais aos modelos conceitual, de navegação, aos cartões de especificação de contexto e às estruturas de acesso, levando à tradução para o código da aplicação.

Com a evolução do ambiente de desenvolvimento OOHDM-Web [Moura 1999] foi criado o Projeto de Navegação, que é a descrição do projeto de uma aplicação hipermídia em estruturas de dados da linguagem Lua [Ierusalimschy 1996]. Esta descrição permite mapear as informações de projeto da aplicação diretamente para o ambiente OOHDM-Web. No entanto, essa descrição não é amigável, é restrita ao ambiente OOHDM-Web e não contempla todos os aspectos do método OOHDM.

Com a criação da metalinguagem XML (Extensible Markup Language) [W3C 1998] pelo W3C (World Wide Web Consortium) [W3C], surgiu a possibilidade de definir uma linguagem

declarativa que permita gerar a especificação do projeto OOHDM de uma aplicação hipermídia independente do ambiente de implementação, e tornar a descrição das primitivas do projeto mais amigável ao ser humano.

A utilização da metalinguagem XML para descrever o projeto OOHDM de uma aplicação hipermídia é uma boa alternativa para a resolução dos problemas apresentados, por diversos motivos:

- ela oferece uma sintaxe padrão que permite a criação de uma estrutura específica para descrever projetos OOHDM;
- permite descrever o projeto utilizando um conjunto de elementos apropriados, que mantêm o nível de abstração próximo ao modelo da aplicação, de forma completamente independente do ambiente de implementação;
- permite que a descrição de projetos OOHDM seja mais amigável ao ser humano, uma vez que os "tags" utilizados são mais próximos ao modelo da aplicação. Ademais, a descrição do projeto poderá ser realizada com o auxílio de uma interface gráfica, oferecida por uma grande variedade de editores XML já disponíveis no mercado.

A definição de uma linguagem declarativa para especificar o projeto OOHDM de uma aplicação hipermídia possibilita, ainda, a descrição do projeto contemplando todos os aspectos do método OOHDM e a construção de tradutores para mapear a especificação declarativa do projeto automaticamente para um ambiente de implementação, que são as principais motivações desse trabalho.

1.2 Objetivo

O objetivo deste trabalho é definir uma linguagem que permita especificar de forma declarativa o projeto de uma aplicação hipermídia e, a partir desta especificação, obter a aplicação final implementada. Para isto, será utilizado o método OOHDM.

Este trabalho apresenta a linguagem de marcação OOHDM-ML, criada para permitir a especificação declarativa do projeto OOHDM de aplicações hipermídia para a Web, e o ambiente OOHDM-XWeb para apoiar a implementação dessas aplicações.

A linguagem OOHDM-ML foi criada definindo-se uma DTD (Document Type Definition), que contém as regras que serão usadas para a criação de documentos XML contendo a especificação do projeto OOHDM de aplicações hipermídia.

O ambiente OOHDM-XWeb foi criado com o objetivo de gerar, de forma automática, a descrição do projeto OOHDM da aplicação hipermídia para o ambiente de desenvolvimento OOHDM-Web 2.0. Esta descrição é gerada a partir da especificação declarativa da aplicação em um documento XML válido de acordo com as regras definidas na DTD OOHDM-ML.

1.3 Estrutura da Dissertação

Esta dissertação inicia apresentando, no segundo capítulo, a linguagem OOHDM-ML que é o ponto de partida deste trabalho. Inicialmente, é apresentada a metalinguagem XML, utilizada neste trabalho para a especificação da linguagem de marcação OOHDM-ML. Em seguida, é feita uma breve descrição do método OOHDM e das atividades que ele propõe para o processo de desenvolvimento de aplicações hipermídia. Por fim, é feita uma descrição da DTD OOHDM-ML que contém a especificação da linguagem OOHDM-ML, ou seja, a definição das regras para a especificação declarativa de projetos OOHDM de aplicações hipermídia.

O terceiro capítulo apresenta o ambiente OOHDM-XWeb, criado para apoiar a implementação de aplicações hipermídia especificadas com a linguagem OOHDM-ML. Inicialmente, é apresentado um resumo da especificação da linguagem XSLT, utilizada neste trabalho para a criação do programa OOHDM-Translation. Em seguida, apresentamos o ambiente de desenvolvimento OOHDM-Web 2.0 e a especificação do projeto OOHDM-Web que ele utiliza. Por fim, apresentamos o programa OOHDM-Translation, responsável pela geração automática do projeto OOHDM-Web.

O quarto capítulo apresenta dois exemplos. O primeiro exemplo mostra a utilização da gramática definida na DTD OOHDM-ML para criar um documento OOHDM-ML válido a partir das definições do projeto OOHDM da aplicação. O segundo exemplo mostra a utilização da folha de estilo OOHDM-Translation para gerar o projeto OOHDM-Web da aplicação, que será usado pelas funções da biblioteca do ambiente OOHDM-Web 2.0.

No quinto capítulo são apresentadas as conclusões sobre este trabalho, indicando suas principais contribuições, alguns trabalhos relacionados e sugestões de trabalhos futuros.

No apêndice serão apresentados a listagem da DTD OOHDM-ML, um glossário para os nomes utilizados no projeto OOHDM-Web, as especificações OOHDM-ML completas das aplicações utilizadas como exemplo e o código fonte do programa OOHDM-Translation.

2 A Linguagem OOHDM-ML

A linguagem de marcação OOHDM-ML, criada utilizando-se a metalinguagem XML, consiste de um conjunto de regras (DTD) que serão usadas para a criação de documentos XML contendo a especificação do modelo conceitual e dos aspectos de navegação de uma aplicação hipermídia, projetada de acordo com as primitivas do método OOHDM.

A utilização desta linguagem torna o processo de especificação do projeto OOHDM mais amigável, uma vez que os "tags" da OOHDM-ML são mais próximos da especificação OOHDM. Ademais, a criação do documento XML poderá ser realizada com o auxílio de um editor XML.

Neste capítulo, apresentamos inicialmente a metalinguagem XML utilizada para criação da linguagem de marcação OOHDM-ML. Em seguida, apresentamos o método OOHDM. Por fim, apresentamos um resumo da especificação da linguagem OOHDM-ML descrevendo seus principais elementos e características.

2.1 A Metalinguagem XML

Desde fevereiro de 1998, quando foi publicada como recomendação pela W3C, a metalinguagem XML tem sido a origem de vários esforços para a criação de novas linguagens de marcação e recomendações associadas. XML, um subconjunto simplificado da SGML (Standard Generalized Markup Language) [ISO 1986], foi criada com o objetivo de garantir um crescimento aberto e padronizado na Web.

XML é uma metalinguagem para a especificação da estrutura de documentos que permite descrever o conteúdo dos dados de forma estruturada. Ela consiste de regras que alguém pode seguir para criar uma linguagem de marcação com seus próprios conjuntos de "tags". Essas regras controlam como os documentos estão estruturados e garantem que um único programa compacto, muitas vezes chamado "parser", pode processar todas essas novas linguagens [Bosak 1999], além de permitir a troca de informação entre diferentes sistemas.

A especificação XML descreve uma classe de objetos de dados chamados documentos XML. Cada documento XML possui marcações que especificam sua estrutura física e sua estrutura lógica.

A estrutura física do documento XML corresponde à organização das suas unidades de armazenamento, denominadas entidades, que podem ser externas ou internas. Entidades externas são arquivos que contém marcações XML. Utilizando essas entidades é possível construir documentos XML em módulos, como ilustrado na figura 2.1.

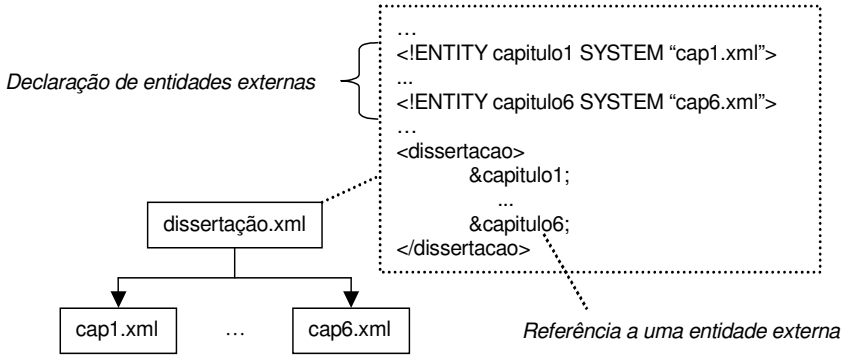


Figura 2.1 – Estrutura física de um documento XML utilizando entidades externas

A figura 2.1 mostra o exemplo de um documento XML para descrever esta dissertação. Para que o documento não fique muito extenso, pode-se especificar cada capítulo como um arquivo físico separado. O arquivo dissertação.xml é a entidade raiz e os arquivos contendo as marcações referentes aos capítulos são entidades externas declaradas no arquivo da entidade raiz.

As entidades internas são um tipo de atalho para um pedaço de marcação XML ou dados contidos dentro do documento. Normalmente, se referem a símbolos reservados para a marcação. Por exemplo, *x > y* representa x > y.

A estrutura lógica de um documento XML é indicada pelas marcações que ele contém. Marcações correspondem a *start-tags*, *end-tags*, *empty-element tags*, referências a entidades, comentários, seções CDATA, declarações de tipo de documento e instruções de processamento, conforme detalhado a seguir.

- **Elementos**

Os elementos são os principais construtores de XML, através dos quais a estrutura hierárquica de um documento é criada. Um documento XML contém um ou mais elementos. Cada elemento possui um tipo, identificado por um nome, e pode ter um conjunto de especificações de atributos a ele associado. Cada especificação de atributo compõem-se de um par nome/valor. A figura 2.2 ilustra um exemplo de um documento XML.

```
<?xml version="1.0"?>
<!DOCTYPE catalogo SYSTEM "catalogo.dtd">

<catalogo mes="abril">
    <produto>
        <nome>Camisa  Polo</nome>
        <preco>R$18,00</preco>
        <item codigo="001" cor="branca"/>
        <item codigo="002" cor="azul"/>
    </produto>
    <produto>
        <nome>Camisa  Social</nome>
        <preco>R$25,00</preco>
        <item codigo="003" cor="bege"/>
        <item codigo="004" cor="branca"/>
        <item codigo="005" cor="amarela"/>
    </produto>
        …
</catalogo>
```

Figura 2.2 – Documento XML para especificação de um catálogo

Um elemento não vazio tem seu conteúdo delimitado por sua *start-tag*, por sua vez delimitada por "<" e ">", e sua *end-tag* delimitada por "</" e ">", como em *<nome>Camisa Polo</nome>*. Um elemento vazio contém apenas atributos e precisa apenas da *empty-element tag* que começa com "<"e termina com "/>", como em *<item codigo="001" cor="branca"/>*. Todo o texto entre a start-tag e end-tag de um elemento faz parte do seu conteúdo. A figura 2.2 ilustra essa notação.

- **Atributos**

Um elemento pode ser classificado qualitativamente através de atributos. Um atributo é um par (nome="valor") presente na start-tag do elemento, logo após o seu nome. Em XML, os valores dos atributos devem estar entre aspas, simples ou duplas. Um atributo não pode

aparecer mais de uma vez no mesmo elemento. Na figura 2.2, um dos elementos "*item*" especifica os atributos "*codigo*" e "*cor*" com os valores "*001*" e "*branca*", respectivamente.

- **Entidades**

Como mencionado anteriormente, a estrutura física de um documento XML consiste de unidades de armazenamento denominadas entidades. Todas as entidades possuem um conteúdo e são identificadas por um nome.

As entidades podem ser internas ou externas. Os valores das entidades internas são atribuídos diretamente na sua declaração. A especificação XML predefine cinco entidades internas: "lt", "gt", "amp", "apos" e "quot". Quando referenciadas, essas entidades internas são substituídas, respectivamente, pelos seguintes caracteres[1]: <, >, &, ' e ".

Entidades externas referenciam unidades de armazenamento separadas (Figura 2.1), que podem conter texto ou dados binários. Quando uma entidade externa possui apenas dados binários seu conteúdo não é passado para o parser e, conseqüentemente, não é analisado. Porém, quando a entidade possui texto, seu conteúdo é passado para o parser e analisado como parte integrante do documento.

Uma referência a uma entidade interna ou externa em um documento XML é da forma "*&nome da entidade;"* como em *&capitulo6;* na figura 2.1. Com exceção das entidades internas pré-definidas, antes de referenciar qualquer entidade é preciso declará-la, sendo que o nome utilizado para referenciar a entidade deve ser o mesmo nome utilizado na sua declaração.

- **CDATA**

Uma seção CDATA permite a inclusão de trechos em um documento XML que, contendo caracteres reservados para marcação, devem ter seu conteúdo ignorado pelo processador. Seções CDATA podem ocorrer em qualquer lugar no documento onde é permitida a ocorrência de dados. Seu formato é *<![CDATA[…]]>*, como ilustrado abaixo:

<![CDATA[*p = &q; b = (i<=3);]]>

Qualquer conjunto de dados pode estar contido nas seções CDATA, exceto a seqüência "*]]>*". Portanto, não é permitido o aninhamento de seções CDATA.

- **Comentários**

Comentários podem ser usados para fazer alguma anotação no documento XML. Eles são delimitados por "*<!--*" e "*-->*", como em *<!-- Isto é um comentário -->*, podem ser de qualquer tamanho, ocorrer em qualquer lugar do documento e conter qualquer conjunto de dados[2], exceto a sequência "--".

Comentários não fazem parte do conteúdo do documento, consequentemente, os processadores podem ou não tornar seus conteúdos disponíveis para as aplicações.

- **Instruções de Processamento**

As instruções de processamento são da forma *<? nome_ip…?>*. Elas não fazem parte do conteúdo do documento XML devendo ser passadas para a aplicação pelo processador. As

[1] Os caracteres <, > e & só devem aparecer em um documento XML, em sua forma literal, se fizerem parte da marcação. Nos outros casos, devem ser substituídos pelas entidades correspondentes.
[2] Dentro dos comentários, os caracteres <, > e & podem ser usados normalmente pois, nestes casos, nenhuma marcação é interpretada.

aplicações devem processar apenas as instruções que reconhecerem, ignorando as demais. Todas as instruções que começam com "*xml*" são reservadas para a linguagem.

Documentos XML podem, e devem, começar com uma instrução de processamento denominada declaração XML, como em *<?xml version = "1.0" ?>* (Figura 2.2). Nesta declaração, o número da versão é provido para futuras extensões da linguagem.

- **Prólogo**

O prólogo de um documento XML é formado por dois componentes principais: a declaração XML e a declaração de tipo de documento.

Conforme visto anteriormente, a declaração XML é um tipo especial de instrução de processamento que indica à aplicação que o documento processado é um documento XML e qual a versão utilizada. A declaração XML, se presente, é sempre a primeira marcação existente no documento.

A declaração de tipo de documento contém, ou referencia, ou ambos, um conjunto de marcações declarativas que definem a gramática do documento, isto é, sua definição de tipo de documento (DTD). Esta declaração deve aparecer entre a declaração XML e o começo do elemento raiz[3].

No exemplo da figura 2.2, as regras para o documento XML podem ser encontradas no arquivo referenciado pela URL "*catalogo.dtd*", denominada subconjunto DTD externo. Isto é indicado pela tag *DOCTYPE* e pelo identificador *SYSTEM* que especifica a localização do arquivo que contém a DTD.

A declaração do tipo de documento também pode conter algumas ou todas as regras dentro dela mesmo, ou seja, no próprio documento XML, como ilustrado na figura 2.3. Neste caso, todas as regras para o documento estão dentro da declaração de tipo de documento e antes do começo do elemento raiz, denominando-se subconjunto DTD interno.

```
<?xml version="1.0"?>
<!DOCTYPE catalogo [
   <!ELEMENT catalogo (produto+)>
   <!ATTLIST catalogo
                  mes   CDATA #IMPLIED>
   <!ELEMENT produto (nome, preco, item+)>
   <!ELEMENT nome (#PCDATA)>
   <!ELEMENT preco (#PCDATA)>
   <!ELEMENT  item EMPTY>
   <!ATTLIST item
               codigo   ID       #REQUIRED
               cor      CDATA  #REQUIRED>
]>
<catalogo mes="abril">
    <produto>
        <nome>Camisa  Polo</nome>
        <preco>R$18,00</preco>
        <item codigo="001" cor="branca"/>
        <item codigo="002" cor="azul"/>
     </produto>
        …
</catalogo>
```

Figura 2.3 – Documento XML contendo um subconjunto DTD interno

[3] Elemento raiz é o elemento que engloba todos os outros elementos, dados ou comentários do documento XML.

Geralmente, a declaração de tipo de documento pode conter ambos os subconjuntos DTD externo e interno. Normalmente, o subconjunto DTD externo é usado para referenciar uma DTD padrão, e o subconjunto DTD interno é usado para declarar as características específicas do documento em questão. Uma observação importante é que as regras especificadas no subconjunto DTD interno serão lidas antes e terão precedência sobre as declarações do subconjunto DTD externo.

- **Definição de Tipo de Documento - DTD**

Uma DTD (Document Type Definition) corresponde às marcações declarativas que definem a gramática do documento. Ela especifica quais elementos e atributos são válidos em um documento XML, e em qual contexto eles são válidos. A figura 2.4 apresenta a DTD que define a gramática do documento XML para especificação de um catálogo, ilustrado na figura 2.2.

```
<!-- catalogo.dtd -->
<!ELEMENT catalogo (produto+)>
<!ATTLIST catalogo
          mes   CDATA #IMPLIED>
<!ELEMENT produto (nome, preco, item+)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT preco (#PCDATA)>
<!ELEMENT  item EMPTY>
<!ATTLIST  item
          codigo   ID       #REQUIRED
          cor      CDATA  #REQUIRED>
```

Figura 2.4 – DTD utilizada para validar o documento XML da figura 2.2.

A declaração de elementos *<!ELEMENT nome_elemento conteúdo>* identifica os nomes dos elementos e a natureza de seu conteúdo. O conteúdo do elemento pode ser especificado pelas palavras "*EMPTY*", "*ANY*" ou "*#PCDATA*" que indicam, respectivamente, que o elemento é vazio, que o elemento pode conter qualquer conteúdo e que o conteúdo do elemento é formado por uma seqüência de caracteres.

O conteúdo de um elemento pode ser composto por outros elementos. Neste caso, deve-se especificar, entre parênteses, todos os elementos permitidos seguidos ou não por algum qualificador. O qualificador pode ser um sinal de adição ("+"), indicando que o elemento deve ocorrer uma ou mais vezes, o asterisco ("*") indicando que o elemento pode ocorrer zero ou mais vezes, e a interrogação ("?") indicando que o elemento é opcional. Caso o qualificador não seja definido, o elemento deve ocorrer uma única vez.

Os elementos permitidos devem ser separados por vírgula (","), indicando que os elementos devem ocorrer na ordem especificada, e/ou por barras verticais ("|") indicando que apenas um dos elementos especificados deve ocorrer. A vírgula tem o significado de "e" e a ordem dos elementos especificados deve ser obedecida. Por exemplo, na figura 2.4 o elemento "*produto*" deve conter como filhos, um elemento "*nome*", seguido por um elemento "*preco*" e depois por um ou mais elementos "*item*", nesta ordem. A barra vertical significa "ou". Se no exemplo anterior as vírgulas usadas para separar os elementos fossem substituídas por barras verticais, o conteúdo do elemento "*produto*" seria composto de um elemento "*nome*", ou de um elemento "*preco*" ou de um ou mais elementos "*item*".

Os atributos são declarados após os elementos da seguinte forma: *<!ATTLIST nome_elemento atributos>.* Cada atributo é composto por três partes: nome, tipo e valor padrão. O tipo pode ser *CDATA*, que define seqüências de caracteres; *ID*, que indica um identificador único para o documento; *IDREF* ou *IDREFS*, que fazem referência a um valor ou

a uma lista de valores de atributos do tipo ID declarados no mesmo documento; *ENTITY*, que faz referência a uma entidade; *NMTOKEN* ou *NMTOKENS* que é uma forma restrita do CDATA e representam um nome ou uma lista de nomes. Um atributo também pode ser do tipo enumeração. Neste caso, o campo tipo na especificação do atributo deve conter os seus possíveis valores, entre parênteses e separados por uma barra vertical.

O valor padrão de um atributo pode ser: *#REQUIRED*, que indica que o atributo deve ser obrigatoriamente especificado; *#IMPLIED*, que indica que o atributo é opcional; *valor*, que indica o valor padrão do atributo quando outro não for especificado; ou *#FIXED valor*, que indica o valor obrigatório do atributo quando este for especificado.

No exemplo da figura 2.4, o elemento "*item*" possui o atributo "*codigo*" do tipo ID e o atributo "*cor*" do tipo CDATA, ambos com valor padrão #REQUIRED. Desta forma, esses atributos devem ser obrigatoriamente especificados todas as vezes que o elemento *item* for especificado. Já o elemento raiz "*catalogo*" possui um único atributo chamado "*mes*" do tipo CDATA e com valor padrão #IMPLIED, indicando que sua especificação é opcional e que seu valor, quando especificado, deve conter uma seqüência de caracteres.

De acordo com a especificação XML, para um objeto de dados ser um documento XML, ele deve estar de acordo com as regras estabelecidas de modo a *ser bem-formado* e, se for o caso, *válido*.

Documento *bem-formado* é aquele que segue as definições léxicas e sintáticas de XML. As condições para que um documento seja *bem-formado* são:

- o documento deve começar com a declaração XML (<?xml version="1.0"?>);
- todos os elementos devem estar contidos dentro de um elemento raiz;
- todos os elementos devem estar devidamente aninhados;
- todos os elementos não vazios devem ter *start-tag* e *end-tag*;
- todas as entidades analisadas que são referenciadas direta ou indiretamente dentro do documento devem ser entidades bem-formadas, ou seja, devem satisfazer estas mesmas regras.

Um documento *válido* é aquele que segue a gramática estabelecida por uma definição de tipo de documento (DTD). Documentos válidos possuem, além da declaração XML, uma declaração de tipo de documento que, conforme visto anteriormente, contém ou referencia as regras estruturais que o documento deve seguir. Esse conjunto de regras, ou gramática, informa ao software como o documento XML deve ser processado, permitindo que o documento seja validado.

Os documentos apresentados nas figuras 2.2 e 2.3 são exemplos de documentos bem-formados e válidos.

Uma aplicação que utilize documentos XML deve processar os documentos e verificar se seu conteúdo está de acordo com as regras de formação de um documento XML em geral (*documento bem-formado*) e, se for o caso, validar sua estrutura e conteúdo frente a gramática correspondente, definida na DTD (*documento válido*).

Como algumas vezes não se deseja validar o documento, mas apenas ler o seu conteúdo, pode-se especificar o atributo "*rmd*" (*required markup declaration*) na declaração XML, apresentada anteriormente. Esse atributo serve de guia para o processador XML e pode assumir três valores: "*internal*", indicando que deve-se validar apenas as declarações internas do documento; "*all*", indicando que deve-se validar tanto as declarações internas quanto as declarações externas ao documento; e "*none*", indicando que nenhuma das declarações do documento deve ser validada.

2.2 O Método OOHDM

O OOHDM é um método integrado para autoria de aplicações hipermídia. Ele provê primitivas de projeto de alto nível e mecanismos de abstração, baseados no paradigma da orientação a objetos, que permitem representar o projeto de aplicações hipermídia complexas que manipulam grande quantidade de informações estruturadas, tais como aplicações para a web, apresentações multimídia, quiosques, etc.

O método OOHDM propõe as seguintes atividades para o processo de desenvolvimento de aplicações hipermídia:

- Levantamento de Requisitos,
- Projeto Conceitual,
- Projeto de Navegação,
- Projeto da Interface Abstrata e
- Implementação.

Essas atividades não seguem o modelo de desenvolvimento em cascata, mas sim, uma mistura de estilos iterativos, incrementais e de prototipação rápida. Em cada passo, um modelo é construído ou enriquecido e ao final do projeto, a aplicação hipermídia é implementada utilizando-se as informações obtidas durante todo o processo.

2.2.1 Levantamento de Requisitos

A atividade de Levantamento de Requisitos apresenta as seguintes fases: identificação de atores e tarefas, especificação de cenários, especificação de use cases, especificação dos diagramas de interação com o usuário, validação dos use cases e diagramas de interação com o usuário.

Na fase de identificação de atores e tarefas, o projetista interage com o domínio da aplicação para identificar atores e tarefas. Esta interação é alcançada através da análise de documentos disponíveis e entrevistas com os usuários. O principal objetivo é perceber e capturar as necessidades dos usuários.

Um exemplo de ator para o domínio de vendas de CDs, apresentado posteriormente como um dos exemplos dessa dissertação, é o *cliente* . Cliente é o usuário que compra CDs através de uma loja virtual. Algumas tarefas relacionadas ao ator cliente são: compra de um CD a partir de seu nome, compra de um CD a partir do nome de uma música e compra de um CD a partir do nome de um artista.

Na fase de especificação de cenários, cada usuário especifica os cenários que descrevem as tarefas que ele deseja realizar no domínio em questão. Cenários são descrições narrativas de como a aplicação pode ser usada pelo usuário. O usuário pode descrever o cenário textualmente ou verbalmente.

A seguir é apresentado um exemplo de cenário especificado por um usuário. Este cenário descreve um usuário comprando um CD baseado no nome de um artista.

Cenário: Comprar um CD a partir do nome de um artista.

Entro com o nome do artista ou as iniciais dele (ex. Caetano ou Ca) e o sistema me retorna um ou mais artista que contém o nome informado. Então, eu seleciono o artista que eu quero e são apresentados os CDs do artista com a capa do CD ao lado. Se um dos CDs apresentados é o CD procurado, eu seleciono o CD e este é anexado à minha lista de compras. Seria desejável que o preço de cada CD também fosse apresentado. Eu posso comprar mais de um CD se eu desejar bastando clicar em mais de um.

Na fase de especificação de use cases, o projetista especifica os use cases a partir dos cenários dos usuários. Um use case é uma forma de usar o sistema [Jacobson 1999]. Use cases tratam apenas a interação entre o usuário e a aplicação ou a informação visível ao usuário. A especificação de use cases requer o agrupamento de todos os cenários que têm uma mesma função. Assim, a descrição de um use case deve incluir a informação apresentada em todos os cenários relacionados.

A seguir é apresentado um exemplo de um use case definido a partir da análise dos cenários que tratam a compra de um CD baseado no nome de um artista.

Use Case: Selecionar um CD baseado no nome do artista.

Cenários: <lista dos cenários analisados>

Descrição:

1. O usuário entra com todo ou parte do nome do artista e, se desejar, o ano ou período do CD procurado.
2. O sistema retorna uma lista de artistas que combinam com a entrada. Se existir somente um artista com aquele nome, é mostrada diretamente o conjunto de CDs (passo 4).
3. O usuário seleciona o artista procurado.
4. O sistema retorna uma lista de CDs do artista. Para cada CD é apresentado o nome, artista, ano, preço, disponibilidade e capa.
5. Se o usuário desejar, ele pode acessar mais informações de um CD (use case **Mostrar CD**).
6. Caso o usuário deseje comprar um ou mais CDs, ele seleciona o(s) CD(s) desejado(s) e inclui na lista de compras para mais tarde efetuar a compra (use case **Comprar**).
7. Se o usuário tiver interesse, ele pode retornar para o passo 5 para comprar outro CD do mesmo artista.

Na fase de especificação dos diagramas de interação do usuário, os diagramas que representam os use cases são especificados. Um diagrama de interação do usuário representa a interação entre o usuário e a aplicação descrita textualmente em um use case. A interação representada descreve a troca de informações entre o usuário e o sistema, sem entrar em detalhes relativos à interface com o usuário.

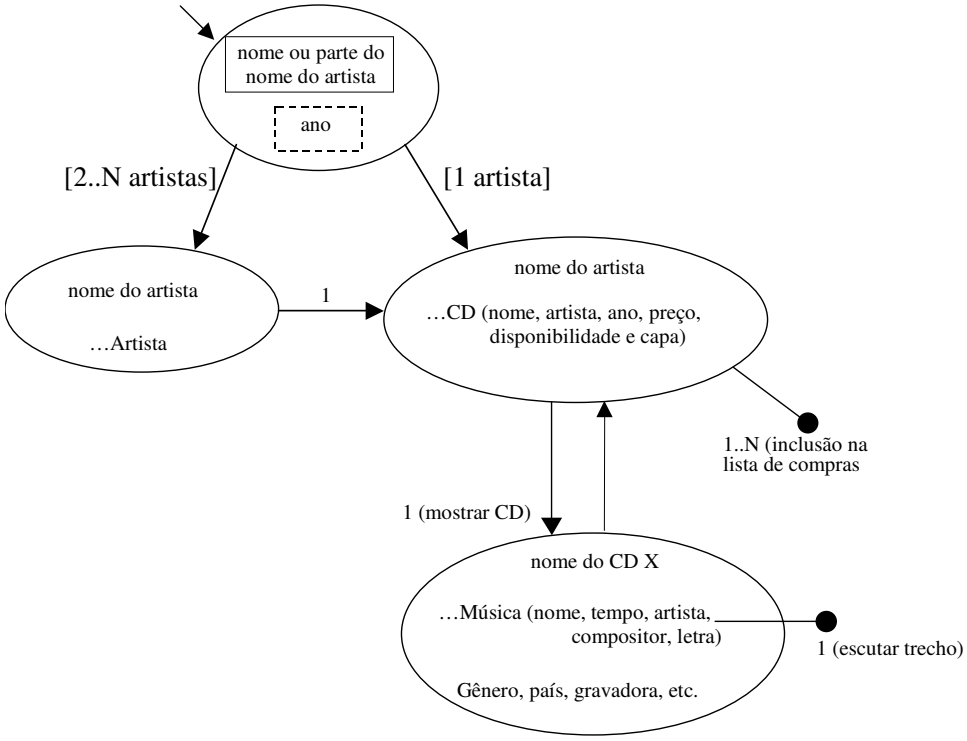


Figura 2.5 – Diagrama de Interação do usuário

A figura 2.5 mostra o diagrama de interação do usuário do use case *Selecionar um CD baseado no nome do artista*.

Na fase de validação dos use cases e dos diagramas de interação do usuário, o projetista interage com cada usuário para validar os uses cases e diagramas de interação do usuário.

2.2.2 Projeto Conceitual

O projeto conceitual é a atividade responsável pela construção de um modelo do domínio da aplicação, utilizando os princípios da modelagem orientada a objetos, com o acréscimo de algumas primitivas, como perspectivas (múltiplos tipos) de atributo e subsistemas (abstrações de um esquema conceitual completo).

O modelo conceitual é definido a partir das descrições dos diagramas de interação do usuário dos use cases simples. Ele é composto por classes de objetos, atributos, relacionamentos, agregação e hierarquia de generalização/ especialização.

O propósito principal dessa atividade é capturar a semântica do domínio, ou seja, engloba todo o universo de informações relevantes para a aplicação em questão, mesmo que apenas um subconjunto dessas informações venha a ser considerado posteriormente na sua implementação.

O produto desta atividade é um esquema conceitual, representado segundo a notação descrita em [Vilain 1999]. Essa notação é similar a UML [UML 1997].

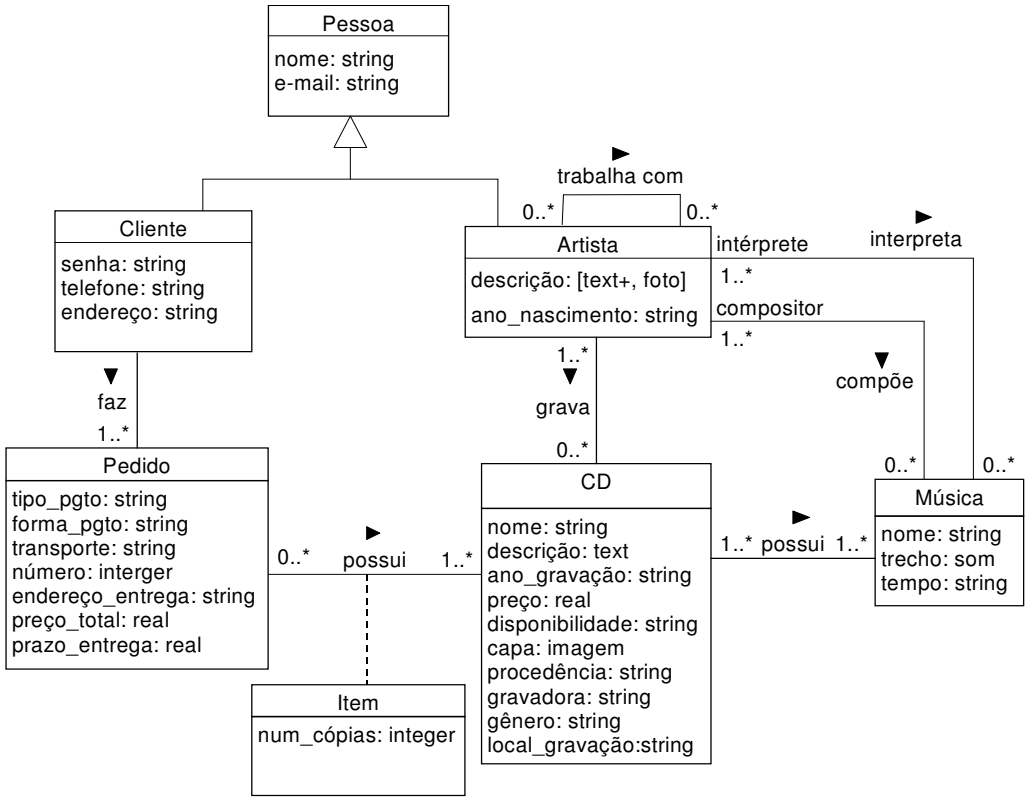


Figura 2.6 – Esquema Conceitual para o domínio de vendas de CDs

A figura 2.6 mostra o esquema conceitual de uma aplicação para o domínio de vendas de CDs. A perspectiva é indicada através da enumeração dos tipos possíveis, com um símbolo + ao lado do tipo *default*. Por exemplo, na classe artista o atributo descrição: [text+, foto] significa que o atributo descrição tem uma representação textual (sempre presente) e pode ter uma representação gráfica, contendo uma foto.

2.2.3 Projeto de Navegação

No método OOHDM, uma aplicação é vista como uma visão navegacional sobre o esquema conceitual, possibilitando que diferentes modelos de navegação sejam construídos sobre o mesmo domínio da aplicação, de acordo com o perfil dos usuários e tarefas que eles irão desempenhar [Moura 1999].

O projeto de navegação é a atividade responsável pela definição da estrutura de navegação de uma aplicação hipermídia, que é composta por: classes navegacionais, elos, âncoras, estruturas de acesso, contextos de navegação e classes em contexto.

Os produtos desta atividade são: o esquema navegacional, o esquema de contextos de navegação e os cartões de especificação de contextos e estruturas de acesso.

Um esquema navegacional define o conjunto de classes navegacionais (ou nós) e elos que fazem parte de uma visão navegacional da aplicação. As classes navegacionais são vistas como uma visão das classes conceituais, uma vez que uma classe navegacional pode não ter todo o conteúdo de informação de uma classe conceitual, ou pode unir conteúdos de mais de uma classe conceitual. A figura 2.7 ilustra o esquema navegacional de uma aplicação para o domínio de vendas de CDs.

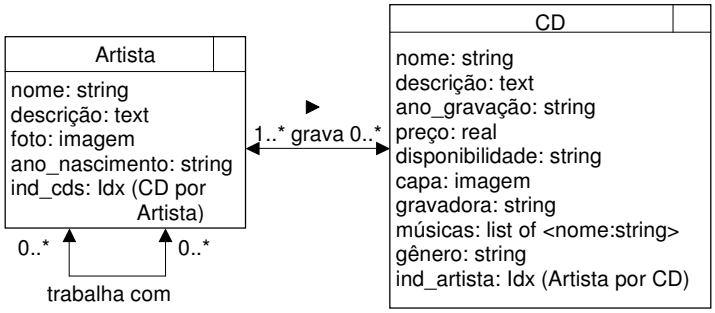


Figura 2.7 – Esquema Navegacional para o domínio de vendas de CDs

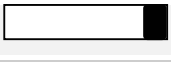
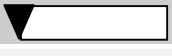
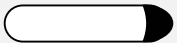
Dependendo da tarefa a ser realizada, o usuário pode navegar por diferentes conjuntos de objetos de classes navegacionais chamados de *contextos de navegação*. Por exemplo, o usuário poderia navegar pelos CDs dos Beatles ou pelos CDs de Rock lançados este ano.

Um contexto de navegação é um conjunto de objetos (nós, elos e outros contextos de navegação) que estão relacionados de acordo com algum aspecto. A definição de um contexto inclui: os elementos pertencentes ao contexto; a especificação da navegação entre esses elementos; o ponto de entrada do contexto; as restrições de acesso; e as estruturas de acesso (índices) associadas ao contexto.

O OOHDM classifica os contextos de navegação da seguinte forma:

- *contexto simples*: inclui todos os elementos de uma classe que satisfazem uma determinada propriedade (*derivado de classe*), ou todos os objetos relacionados a um dado objeto (*derivado de elo*). Por exemplo, o contexto "*CDs gravados no ano 1990*" é um contexto simples derivado de classe, e o contexto "*CDs gravados pelos Beatles*" é um contexto simples derivado de elo.
- *grupo de contexto*: é um conjunto de contextos simples, que podem ser contextos derivados de classe ou derivados de elo. No grupo de contextos derivados de classe, a propriedade definidora de cada contexto é parametrizada. Por exemplo, no contexto "*CDs por ano*" a propriedade ano_gravação pode variar. Já no grupo de contextos derivados de elo, cada contexto do grupo é obtido variando o elemento origem do elo. Por exemplo, "*CDs por artista*" (o artista varia).
- *contexto arbitrário*: neste tipo de contexto, os elementos são escolhidos explicitamente pelo autor do contexto, podendo ser elementos que pertencem a classes navegacionais diferentes. Os roteiros guiados são exemplos deste tipo.
- *contexto dinâmico*: é um contexto cujos elementos são definidos ou alterados em tempo de execução. Podemos utilizar este tipo de contexto para criação, modificação ou exclusão das instâncias de uma classe navegacional. Histórico de navegação e cesta de compras são exemplos deste tipo. Um tipo especial de contexto dinâmico é o *contexto baseado em sessão* (ou temporário), que só existe durante a sessão de navegação.

Representação gráfica para os tipos de contexto:

Contexto Simples ou Grupo de Contextos	
Contexto Dinâmico	
Contexto com Ordenação Múltipla	
Contexto de Criação de Instâncias	

Para todos tipos de contexto descritos acima pode haver uma estrutura de acesso definida. As estruturas de acesso são índices que permitem o acesso ao contexto. Elas são representadas graficamente da seguinte forma:

Índice Simples	
Índice Dinâmico	
Índice com Ordenação Múltipla	

A especificação da navegação entre os elementos do contexto define como será a navegação dentro do contexto, caracterizando a forma como se pode percorrer os elementos de um dado contexto.

Quando um usuário entra num dado contexto de navegação, o tipo de trilhas navegacionais que ele pode seguir dependerá da natureza do contexto e do tipo de especificação que for feita. Por exemplo, podemos fornecer âncoras (e elos de contextos correspondentes), permitindo o acesso ao nó seguinte (ou anterior) no contexto. Em alguns contextos, pode-se oferecer acesso somente ao índice, tendo o usuário que escolher outro item do índice para prosseguir a navegação.

Os tipos de navegação interna ao contexto definidos pelo OOHDM são:

- *navegação seqüencial*: após a entrada no contexto, o usuário pode navegar por todos os elementos do contexto seguindo uma ordem seqüencial e pré-estabelecida de navegação. São definidos os conceitos de primeiro, último, próximo e anterior para os elementos do contexto;
- *navegação circular*: os elementos podem ser vistos como se estivessem dispostos em uma lista circular. Unicamente são definidos os conceitos de próximo e anterior;
- *navegação limitada ao índice*: a navegação é feita apenas do índice para um elemento do contexto e vice-versa. Não existe a possibilidade de navegação entre os outros elementos do contexto, a não ser através do índice de entrada do contexto;
- *combinação da navegação por índice e seqüencial*: os elementos do contexto podem ser acessados indistintamente por índice ou através das âncoras "Anterior" e "Próximo";
- *navegação livre*: todos os elementos do contexto podem ser acessados diretamente a todo momento, não havendo a necessidade de se percorrer nenhum caminho pré-determinado para atingir um dado elemento.

A estrutura navegacional da aplicação é definida no esquema de contextos, que mostra todas as estruturas de acesso e contextos definidos para a aplicação, e as possíveis navegações entre eles.

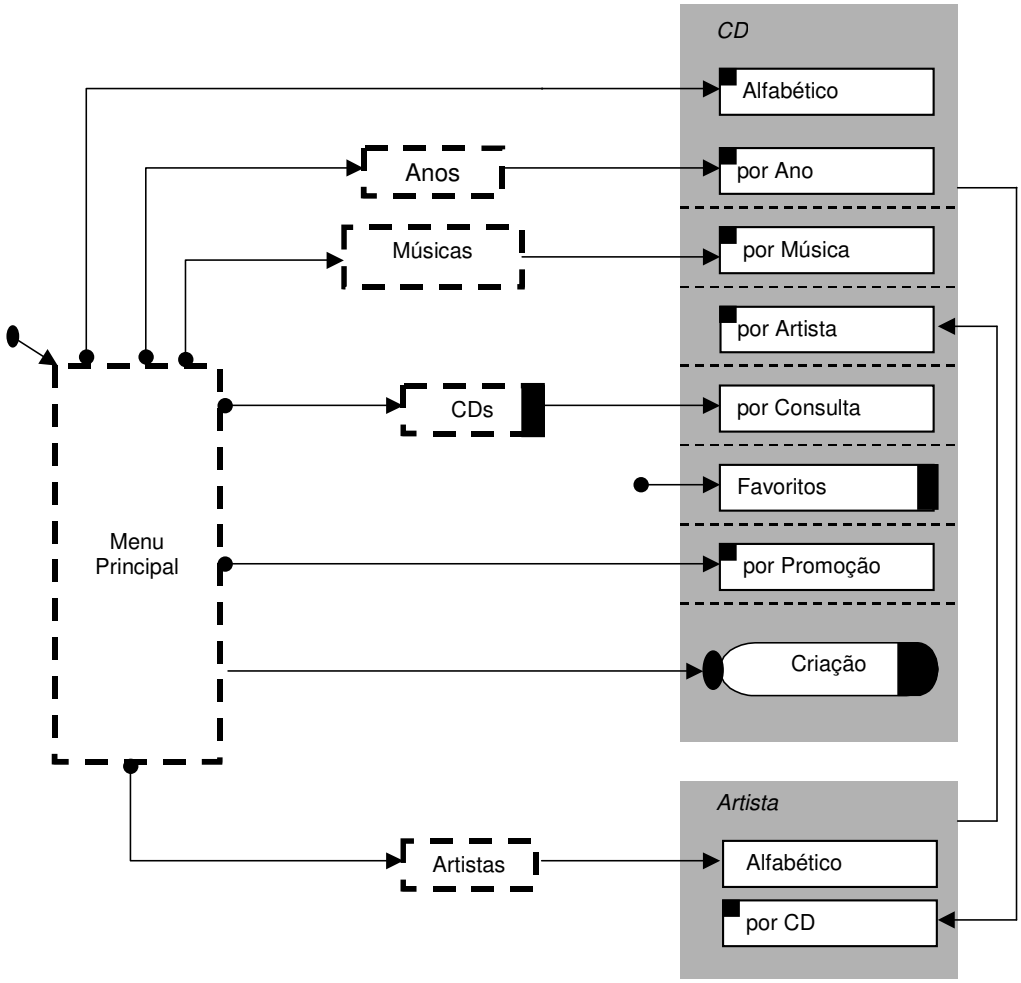


Figura 2.8 – Esquema de Contextos para o domínio de vendas de CDs

A figura 2.8 mostra o esquema de contextos de navegação de uma aplicação para o domínio de vendas de CDs. O pequeno retângulo preto no canto superior esquerdo do contexto "*CDs por Artista*" representa um índice associado a este contexto. A seta com um círculo em uma das extremidades que aparece no lado esquerdo do contexto "*Favoritos*" representa o "design pattern" *landmark* [Rossi 1999], indicando que um contexto ou um índice pode ser acessado em qualquer local da aplicação. Já a pequena elipse associada ao contexto "*Criação*" indica que o acesso a esse contexto é protegido, ou seja, somente usuários com permissão poderão acessá-lo.

Os contextos de navegação e as estruturas de acesso são especificadas utilizando-se os *cartões de especificação*. As propriedades mais relevantes para caracterizar um contexto de navegação são: os elementos que compõem o contexto; a ordenação desses elementos; os pontos de entrada do contexto; e a navegação entre os elementos do contexto. Para uma estrutura de acesso, as propriedades mais importantes a especificar são: os seletores (atributos que são âncoras na estrutura); os demais atributos exibidos na estrutura; a ordenação dos elementos na estrutura; e o destino da estrutura de acesso (pode ser um contexto ou outra estrutura de acesso).

As classes navegacionais podem apresentar características específicas dentro de diferentes contextos de navegação. Por exemplo, se um CD é acessado no contexto "*CDs por Ano*", o nó CD exibe também a foto e um resumo da biografia do artista que gravou o CD. Se este mesmo CD é acessado no contexto "*CDs por Artista*" essa informação não é exibida. Por esta

razão, classes em contexto são definidas para representar as características específicas que um objeto pode apresentar dentro de um contexto particular.

As classes em contexto são classes especiais que decoram os nós, permitindo que um mesmo nó tenha uma aparência diferente e apresente âncoras e funcionalidades distintas quando é mostrado em um contexto específico. Classes em contexto também são utilizadas para especificar os atributos com múltiplas perspectivas, que não possuem uma perspectiva default, definidos no modelo conceitual. Cada perspectiva do atributo deve ser mapeada para um atributo em uma classe em contexto.

2.2.4 Projeto da Interface Abstrata

O projeto da interface abstrata especifica como os objetos de navegação serão percebidos e apresentados. Especifica, também, os objetos de interface que representam as interações.

Esta atividade é desenvolvida antes de iniciar a implementação e de forma independente do ambiente de implementação. Entretanto, deve considerar também algumas características do ambiente para que possa ser implementado.

O OOHDM utiliza Abstract Data View (ADV), diagramas de configuração e ADVCharts para especificar o projeto da interface abstrata.

2.2.5 Implementação

A atividade de implementação é a última atividade proposta pelo método OOHDM e é responsável pela tradução do projeto da aplicação para um ambiente de implementação.

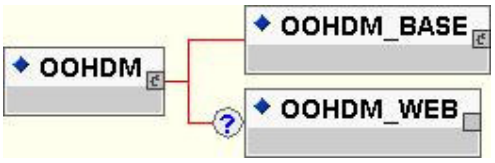
Se a aplicação for implementada na Web, o ambiente OOHDM-XWeb proposto neste trabalho pode ser utilizado. O ambiente OOHDM-XWeb é composto por um programa chamado OOHDM-Translation e pelo sub-ambiente OOHDM-Web 2.0. Ele permite a geração automática da descrição do projeto OOHDM de uma aplicação hipermídia para o ambiente OOHDM-Web 2.0, utilizando o documento XML que contém a especificação declarativa do projeto, definida utilizando-se a linguagem OOHDM-ML.

2.3 A DTD OOHDM-ML

Nesta seção, descrevemos a DTD OOHDM-ML, que contém as regras que devem ser obedecidas na criação de documentos XML para a especificação declarativa do projeto OOHDM de aplicações hipermídia. A DTD OOHDM-ML se encontra listada no apêndice I e em [Medeiros 2001] pode ser encontrada a especificação completa e detalhada da linguagem OOHDM-ML, que contém a descrição de todos os elementos e atributos definidos nesta DTD.

2.3.1 Especificação da Linguagem OOHDM-ML

Um documento XML, criado utilizando-se a linguagem OOHDM-ML, é sempre formado por um elemento raiz denominado *OOHDM*. Este elemento contém todas as marcações que representam a especificação declarativa de uma aplicação hipermídia, projetada de acordo com as primitivas do método OOHDM. Seu conteúdo é formado por um elemento *OOHDM_BASE* e por um elemento opcional *OOHDM_WEB*. A figura 2.9 ilustra o conteúdo do elemento raiz OOHDM e um exemplo de sua definição na DTD.




```
<!ELEMENT OOHDM (OOHDM_BASE, OOHDM_WEB?)>
 <!ATTLIST OOHDM
    name     CDATA #REQUIRED
    version  CDATA #REQUIRED >
<!ELEMENT OOHDM_BASE (conceptual_model, navigational_model, interface_model)>
<!ENTITY % oohdm_web SYSTEM "OOHDM_WEB.dtd">
 %oohdm_web;
```

Figura 2.9 – Conteúdo do elemento raiz OOHDM e sua definição na DTD

O elemento *OOHDM_BASE* contém todos os elementos que descrevem as primitivas do método OOHDM. Ele representa a especificação do projeto OOHDM de uma aplicação hipermídia que é composto basicamente por três modelos: o modelo conceitual, o modelo navegacional e o modelo de interface. Já o elemento opcional *OOHDM_WEB* foi definido para permitir a especificação de informações específicas para o ambiente OOHDM-Web 2.0, necessárias para a implementação da aplicação neste trabalho. Posteriormente, outros elementos poderão ser definidos para a especificação das informações de outros ambientes de implementação, como por exemplo o OOHDM-Java [Pizzol 1998]. Como pode ser observado na figura 2.9, o conteúdo do elemento *OOHDM_WEB* é definido em uma outra DTD, denominada OOHDM_WEB.dtd. Isto permite a separação da definição das marcações para especificação do projeto da aplicação e marcações específicas para as informações do ambiente de implementação. A figura 2.10 ilustra a estrutura de um documento XML especificado em OOHDM-ML.

```
<?xml version="1.0"?>
<!DOCTYPE OOHDM SYSTEM "OOHDM_v1_2.dtd"
[
  <!ENTITY modelo_conceitual   SYSTEM "cds_con_model.xml">
  <!ENTITY modelo_navegacao   SYSTEM "cds_nav_model.xml">
  <!ENTITY mapeamento_oohdmweb SYSTEM "cds_oohdmweb.xml">
]>
<OOHDM name="Venda de CDs" version="1.0">
    <OOHDM_BASE>
        <conceptual_model>
            &modelo_conceitual;
        </conceptual_model>
        <navigational_model>
            &modelo_navegacao;
        </navigational_model>
        <interface_model/>
    </OOHDM_BASE>
    <OOHDM_WEB>
        &mapeamento_oohdmweb;
    </OOHDM_WEB>
</OOHDM>
```

Figura 2.10 – Estrutura de um documento OOHDM-ML

O conteúdo do elemento *OOHDM_BASE* é formado por um elemento *conceptual_model*, um elemento *navigational_model* e um elemento *interface_model* que representam, respectivamente, o modelo conceitual, o projeto de navegação e o projeto da interface abstrata de uma aplicação hipermídia. A figura 2.11 ilustra o conteúdo do elemento OOHDM_BASE.

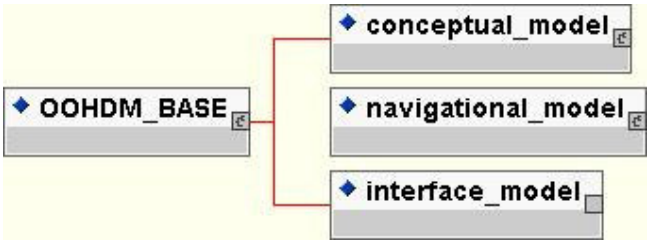


Figura 2.11 – Conteúdo do elemento OOHDM_BASE

O elemento *conceptual_model* contém as marcações que descrevem o modelo conceitual de uma aplicação hipermídia. Seu conteúdo é formado por elementos que especificam as classes conceituais e os relacionamentos que compõem o modelo do domínio da aplicação. A figura 2.12 ilustra o conteúdo do elemento conceptual_model.

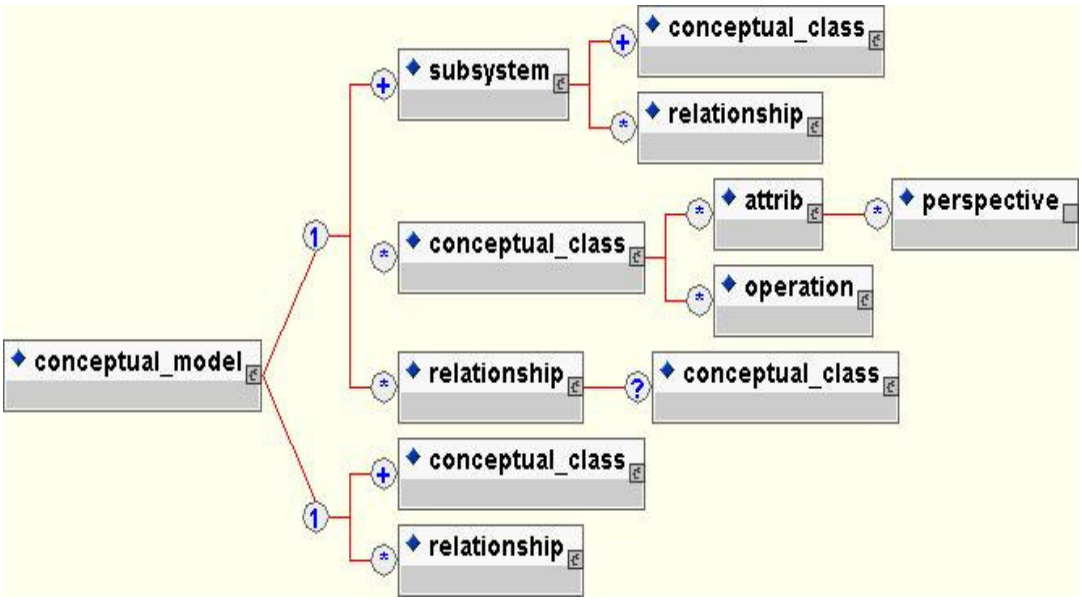


Figura 2.12 – Conteúdo do elemento conceptual_model

O conteúdo do elemento conceptual_model pode ser formado por um ou mais elementos *subsystem*, podendo conter também elementos *conceptual_class* e *relationship*, ou por um ou mais elementos *conceptual_class*, podendo conter vários elementos *relationship*. As duas alternativas apresentadas na figura 2.12 para o conteúdo do elemento conceptual_model mostram que, segundo as primitivas do método OOHDM, as classes e os relacionamentos do modelo conceitual da aplicação podem ou não ser agrupados em subsistemas.

O elemento *conceptual_class* descreve uma classe conceitual, seus atributos e operações. Ele possui um atributo opcional denominado **superclass**, que permite especificar que a classe conceitual herda as características de uma ou mais superclasses em uma hierarquia de generalização/especialização. Os atributos e as operações da classe são especificados, respectivamente, pelos elementos *attrib* e *operation*. O elemento attrib possui um elemento perspective que permite especificar que um atributo possui múltiplas perspectivas.

O elemento *relationship* descreve os relacionamentos existentes entre os objetos das classes conceituais. Seu conteúdo pode ser formado por um elemento opcional *conceptual_class* para especificar que o relacionamento definido apresenta propriedades de classe (atributos e comportamento), formando uma classe de relacionamento no modelo conceitual. Todas as informações sobre o relacionamento são especificadas atribuindo-se valores aos atributos do elemento relationship. Esses atributos especificam o nome do relacionamento, as classes origem e destino do relacionamento e suas cardinalidades, como ilustra a figura 2.13.

O elemento *subsystem* descreve as classes e relacionamentos do modelo conceitual que tratam de um mesmo assunto, ou seja, são abstrações de um sistema conceitual completo.

A figura 2.13 ilustra parte do documento cds_con_model.xml, entidade externa referenciada na figura 2.10, que contém a especificação do conteúdo do elemento conceptual_model para o exemplo do domínio de vendas de CDs.

```
    <!-- ============== Classe Conceitual CD ============== -->

<conceptual_class id="id_cd" name="cd">
    <attrib name="chave_cd" type="string" size="20"/>
    <attrib name="nome_cd" type="string" size="70"/>
    <attrib name="ano_gravacao_cd" type="string" size="4"/>
</conceptual_class>

  <!-- ============== Classe Conceitual Artista ============== -->

<conceptual_class id="id_artista" name="artista">
    <attrib name="chave_artista" type="string" size="20"/>
    <attrib name="nome_artista" type="string" size="50"/>
    <attrib name="descricao_artista" type="text">
       <perspective name="descricao_artista"  type="text" default="yes"/>
       <perspective name="foto_artista" type="image" size="50"/>
    </attrib>
    <attrib name="ano_nasc_artista" type="string" size="4"/>
</conceptual_class>
          ...
<!-- ============ Relacionamento Artista grava CD ============ -->

<relationship id="relacao_grava" name="artista_grava_cd"
                  source_class="id_artista" target_class="id_cd"
                  source_cardinality="n" target_cardinality="n">
</relationship>
          ...
```

Figura 2.13 – Especificação OOHDM-ML do modelo conceitual para vendas de CDs

O elemento *navigational_model*, filho do elemento OOHDM_BASE, contém as marcações que descrevem o modelo de navegação da aplicação hipermídia. Seu conteúdo é formado por elementos que especificam os nós, os elos, os contextos, as classes em contexto e as estruturas de acesso que compõem o projeto de navegação da aplicação. A figura 2.14 ilustra o conteúdo do elemento navigational_model.

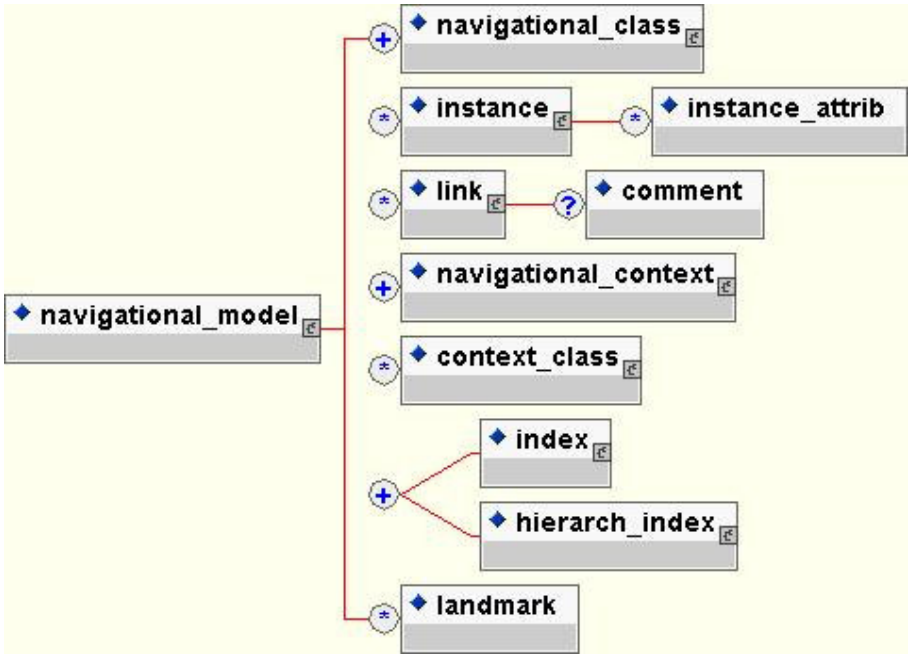


Figura 2.14 – Conteúdo do elemento navigational_model

O conteúdo do elemento navigational_model deve ser formado, obrigatóriamente, por um ou mais elementos *navigational_class*, um ou mais elementos *navigational_context* e um ou mais elementos *index* ou *hierarch_index*. Pode conter, opcionalmente, os elementos *instance*, *link*, *context_class* e *landmark*.

O elemento *navigational_class* descreve os nós (ou classes navegacionais) do modelo de navegação da aplicação hipermídia. Seu conteúdo pode ser formado por elementos *nav_attrib*, *nav_attrib_index*, *perspective_attrib*, *nav_operation* e *anchor*, como ilustra a figura 2.15.

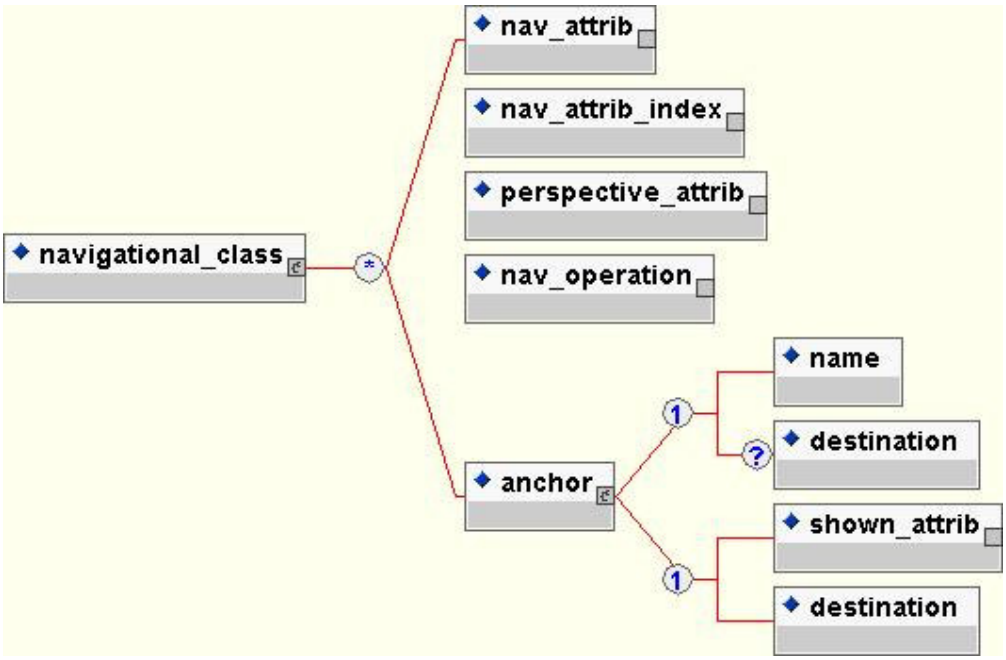


Figura 2.15 – Conteúdo do elemento navigational_class

Os elementos *nav_attrib* e *nav_attrib_index* descrevem atributos de um nó do modelo de navegação. O elemento nav_attrib representa um atributo já definido em uma classe do modelo conceitual que será apresentado no nó. Já o elemento nav_attrib_index representa um atributo do tipo índice e serve para especificar um índice que será apresentado no nó.

O elemento *perspective_attrib* descreve o mapeamento de um atributo de uma classe conceitual que possui múltiplas perspectivas, sendo uma delas default, para vários atributos em um nó, um para cada perspectiva. O elemento *nav_operation* representa uma operação já definida em uma classe conceitual que será apresentada e executada no nó.

O elemento *anchor* descreve uma âncora. Possui o atributo opcional **type** para especificar as âncoras fixas "next", "previous", "first" e "last", que são especificadas nas classes em contexto para implementar a navegação entre os elementos de um contexto. As duas alternativas para o conteúdo do elemento anchor mostram que as âncoras podem possuir um nome fixo, descrito no elemento *name*, ou exibir o valor de um atributo de um nó, descrito no elemento *shown_attrib*. Se o valor do atributo type for especificado, o elemento *anchor* deve receber a primeira alternativa como conteúdo. Neste caso, se o contexto destino da âncora for o contexto corrente, o elemento opcional destination não será especificado. Caso contrário, o atributo target do elemento destination deve conter o valor do atributo identificador que representa o contexto ou o índice destino da âncora.

Quando especificado como conteúdo do elemento navigational_class, o elemento *anchor* representa um atributo do tipo âncora e serve para especificicar uma âncora que será apresentada no nó.

A figura 2.16 ilustra um exemplo de especificação para duas classes navegacionais do domínio de vendas de CDs: **cd** e **artista**.

```
        <!-- ============== Classe Navegacional CD ============== -->
<navigational_class id="no_cd" name="cd">
     <nav_attrib name="chave_cd" conceptual_class="id_cd"/>
     <nav_attrib name="nome" conceptual_class="id_cd"/>
     <nav_attrib name="ano_gravacao" conceptual_class="id_cd"/>
</navigational_class>
      <!-- ============== Classe Navegacional Artista ============== -->
<navigational_class id="no_artista" name="artista">
      <nav_attrib name="chave_artista" conceptual_class="id_artista"/>
      <nav_attrib name="nome" conceptual_class="id_artista"/>
      <perspective_attrib name="descricao" perspective="descricao_artista"
                                    conceptual_attrib="descricao_artista"
                                     conceptual_class="id_artistas"/>
      <perspective_attrib name="foto" perspective="foto_artista"
                                    conceptual_attrib="descricao_artista"
                                     conceptual_class="id_artistas"/>
      <nav_attrib_index name="indice_cds"  index="indice_cds_por_artista_H"/>
</navigational_class>
```

Figura 2.16 – Especificação OOHDM-ML das classes navegacionais cd e artista

A classe navegacional *artista* possui um atributo do tipo índice denominado indice_cds, cujo valor é o índice hierárquico *indice_cds_por_artista_H*, representado na especificação pelo elemento *nav_attrib_index*. Os elementos perspective_attrib especificam o mapeamento do atributo *descricao_artista* da classe conceitual *artista*, que possui uma perspectiva default, para dois atributos da classe navegacional artista: *descricao* e *foto*.

O elemento *instance*, filho do elemento navigational_model, descreve uma instância de um nó. Os atributos do nó instanciado e seus respectivos valores são especificados através do elemento *instance_attrib*.

O elemento *link* descreve todas as propriedades (cardinalidade, papéis, etc.) do elo existente entre classes navegacionais no modelo de navegação. Seu conteúdo pode ser formado por um elemento *comment* que representa um comentário sobre o elo especificado.

O elemento *navigational_context* descreve um contexto de navegação. Possui o atributo **element_class** que especifica as classes navegacionais que fornecem os objetos do contexto, o atributo **type** que especifica se o contexto é estático, dinâmico, temporário ou por consulta, e o atributo **navigation_type** que especifica o tipo de navegação permitida entre os elementos do contexto, entre outros. A figura 2.17 ilustra o conteúdo deste elemento.

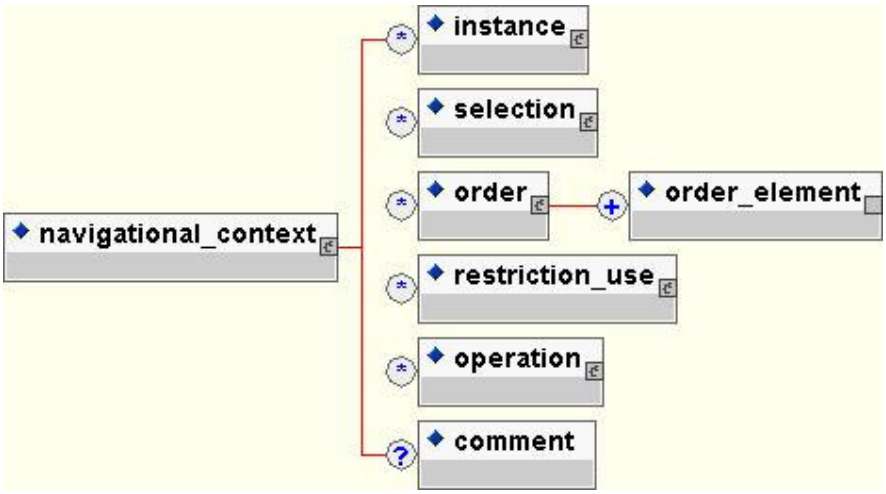


Figura 2.17 – Conteúdo do elemento navigational_context

O conjunto de objetos de um contexto pode ser formado por instâncias dos nós de navegação, descritas pelo elemento *instance*, ou pelo resultado de uma seleção em um ou mais nós, representada pelo elemento *selection*. Os elementos de um contexto podem ser ordenados seguindo um ou mais critérios de ordenação. Esta ordenação é representada pelo elemento *order* e os critérios de ordenação são especificados através do elemento order_element, que possui um atributo chamado **criteria** que pode conter os valores "asc" ou "desc" indicando, respectivamente, os critérios de ordenação ascendente e descendente. Por fim, os elementos *restriction_use*, *operation* e *comment* descrevem, respectivamente, as permissões de acesso para cada classe de usuários do contexto, as operações que manipulam os objetos do contexto e um comentário sobre o contexto de navegação. A figura 2.18 mostra um exemplo de especificação do contexto de navegação *cds_por_genero* para o domínio de vendas de CDs.

```
<!-- ============== Contexto CDs por gênero ============== -->
<navigational_context id="contexto_cds_por_genero" name="cds_por_genero"
                      element_class="no_cd"   type="static"  navigation_type="sequential">
    <selection>
      <equal>
        <attribute name="genero" navigational_class="no_cd"/>
        <nav_parameter>genero_par</nav_parameter>
      </equal>
    </selection>
    <order id="ordenacao_cds_por_genero">
        <order_element name="genero" navigational_class="no_cd" criteria="asc"/>
    </order>
</navigational_context>
```

Figura 2.18 – Especificação OOHDM-ML do contexto cds_por_genero

No exemplo da figura acima, o conjunto de objetos do contexto *cds_por_genero* é formado pelo resultado da seleção representada pelo elemento *selection*, que contém os objetos do nó *cd* (especificado no atributo element_class), cujo atributo genero tem o mesmo valor informado no parâmetro genero_par (genero = genero_par). Os objetos retornados por essa seleção serão ordenados pelo valor do atributo genero em ordem ascendente, conforme especificado nos elementos *order* e *order_element*. O atributo *navigation_type* indica que a navegação entre os objetos do contexto será sequencial.

O elemento *context_class* descreve uma classe em contexto definida para um ou mais contextos de navegação. Possui um atributo chamado **contexts** que especifica os contextos de navegação dos quais a classe em contexto participa. A figura 2.19 ilustra o conteúdo deste elemento.

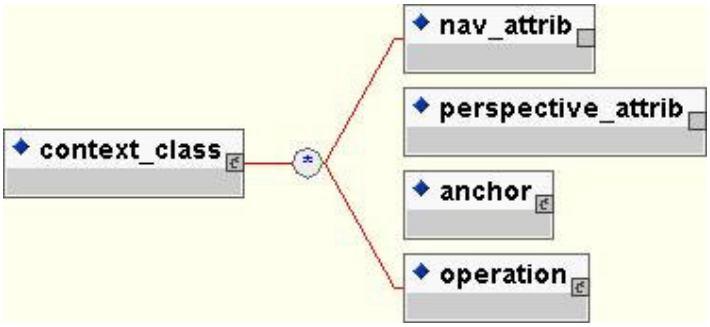


Figura 2.19 – Conteúdo do elemento context_class

As classes em contexto são criadas para definir a aparência e as âncoras de cada nó em cada contexto ao qual ele pertence. Os elementos *nav_attrib* e *anchor* descrevem os atributos e as âncoras de um nó que só serão apresentados em contextos de navegação específicos. Conforme mencionado anteriormente, o elemento *anchor* também descreve as âncoras fixas "next", "previous", "first" e "last", que implementam a navegação interna entre os elementos de um contexto.

O elemento perspective_attrib descreve os atributos com múltiplas perspectivas que não apresentam perspectiva default, definidos no modelo conceitual. Cada uma dessas perspectivas deve ser mapeada para um atributo em uma classe em contexto. Por fim, o elemento operation descreve as operações de um nó que também só serão apresentadas em contextos específicos.

Os elementos *index* e *hierarch_index* descrevem os índices simples e os índices hierárquicos do modelo de navegação, que representam as estruturas de acesso aos contextos. Estes elementos possuem o atributo **element_class,** que especifica quais são as classes navegacionais que compõem os elementos do índice, e o atributo **type** que especifica se o índice é estático, dinâmico ou temporário (derivado de sessão), entre outros. A figura 2.20 ilustra o conteúdo do elemento index.

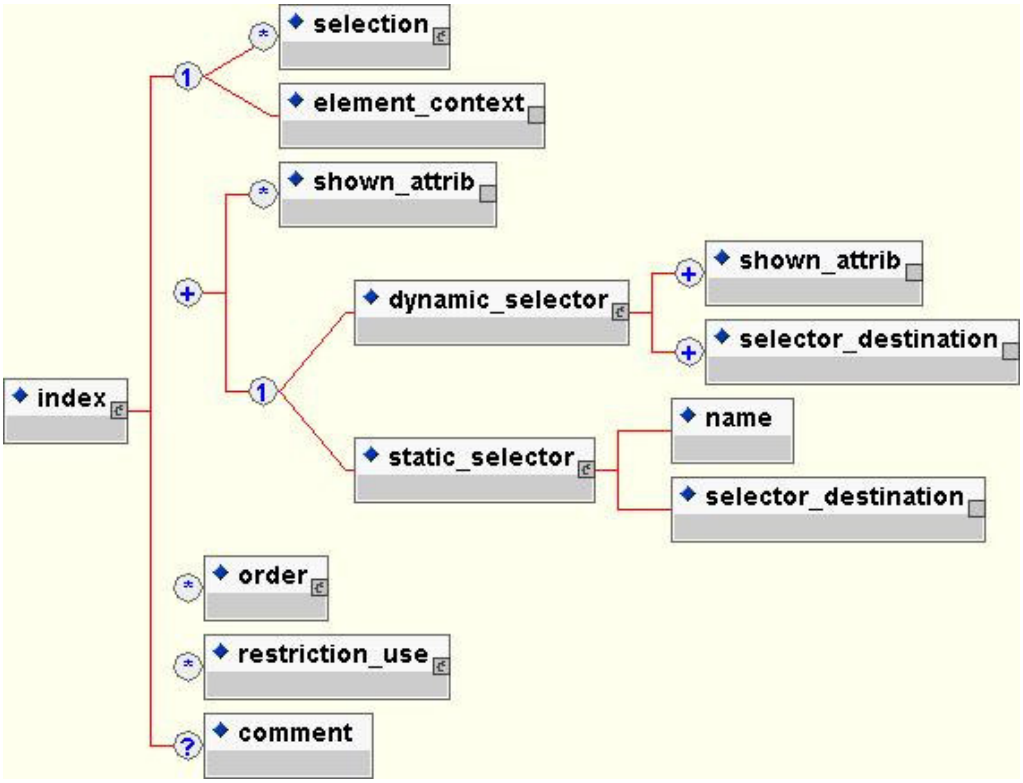


Figura 2.20 – Conteúdo do elemento index

Um índice pode apresentar o conjunto de objetos de um contexto, especificado pelo elemento *element_context* ou os objetos resultantes de uma seleção em um ou mais nós, especificada pelo elemento *selection*. Os atributos de um nó e os atributos seletores, cujos valores serão apresentados no índice, são descritos pelos elementos *shown_attrib*, *dynamic_selector* e *static_selector*, respectivamente.

O elemento *dynamic_selector* descreve os seletores dinâmicos de um índice, ou seja, aqueles seletores formados pelo valor de um atributo do nó. O conteúdo deste elemento é formado por elementos *shown_attrib*, que especificam os nomes dos atributos cujos valores serão apresentados no índice, e elementos *selector_destination* que especificam o destino de um atributo seletor, que pode ser um contexto, um outro índice ou ainda uma instância específica dentro de um contexto.

O elemento *static_selector* descreve os seletores estáticos de um índice, ou seja, aqueles seletores que possui um nome fixo representado pelo elemento *name*. Da mesma forma que o seletor dinâmico, um seletor estático tem um destino especificado pelo elemento *selector_destination*.

Por fim, os elementos *order*, *restriction_use* e *comment* descrevem, respectivamente, o critério de ordenação utilizado para ordenar os objetos apresentados no índice, as permissões de acesso para cada classe de usuários do índice e um comentário sobre o índice.

A figura 2.21 mostra um exemplo de especificação do índice *cds_por_genero_idx* para o domínio de vendas de CDs.

```
<!-- ======= Índice dos elementos do contexto CDs por genero ===== -->
<index  id="indice_cds_por_genero" name="cds_por_genero_idx" type="static">
    <element_context context="contexto_cds_por_genero"/>
    <dynamic_selector>
        <shown_attrib  name="nome" navigational_class="no_cd"/>
        <selector_destination  target="contexto_cds_por_genero"/>
    </dynamic_selector>
</index>
```

Figura 2.21 – Especificação OOHDM-ML do índice cds_por_genero_idx

Neste exemplo, temos a especificação de um índice formado pelos elementos do contexto *cds_por_genero* representado pelo elemento *element_context*. Os atributos seletores desse índice são dinâmicos (*elemento dynamic_selector*), ou seja, serão formados pelos valores do atributo nome da classe navegacional cd (*elemento shown_attrib*) e terão como destino o próprio contexto *cds_por_genero* (*elemento selector_destination*).

Um índice hierárquico possui as mesmas características de um índice simples com uma única diferença: os atributos do nó e os atributos seletores que serão apresentados no índice hierárquico são definidos em níveis de hierarquia especificados pelo elemento *level*, como pode ser visto na figura 2.22, que ilustra o conteúdo do elemento *hierarch_index*.

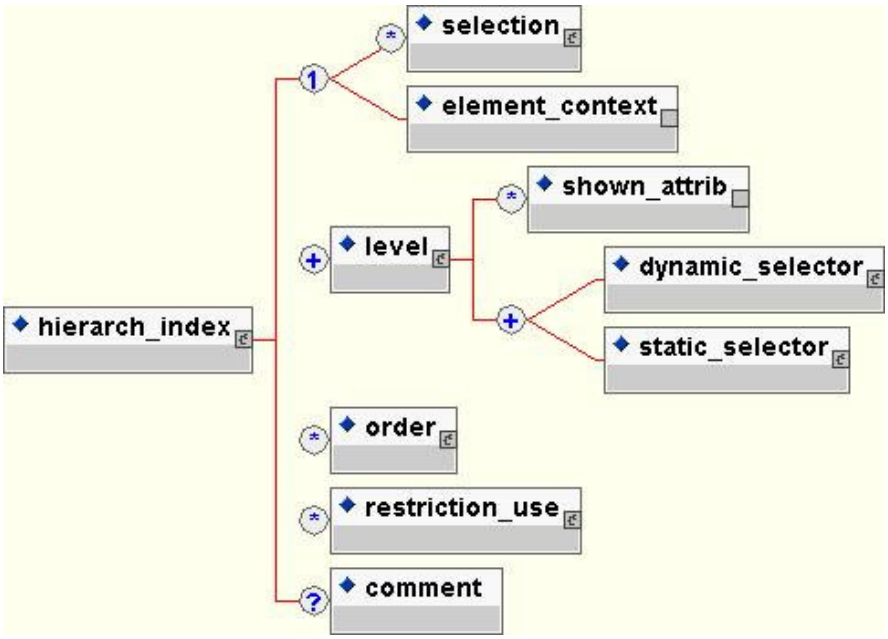


Figura 2.22 – Conteúdo do elemento hierarch_index

O elemento *landmark* descreve o design pattern landmark. Os landmarks representam os pontos acessíveis a partir de qualquer objeto na aplicação. O conteúdo desse elemento é formado por uma sequência de caracteres que especificam o nome que identifica o landmark. O elemento lankmark possui um único atributo chamado **target**. Esse atributo especifica o destino do landmark, que pode ser um índice dos elementos de um contexto ou uma instância de um objeto em um contexto.

A figura 2.23 ilustra parte do documento cds_nav_model.xml, entidade externa referenciada na figura 2.10, que contém o conteúdo do elemento *navigational_model* para o exemplo do domínio de vendas de CDs.

```
            <!-- ============== Classe navegacional CD ============== -->
<navigational_class id="no_cd" name="cd">
    <nav_attrib name="chave_cd" conceptual_class="id_cd"/>
    <nav_attrib name="nome" conceptual_class="id_cd"/>
    <nav_attrib name="ano_gravacao" conceptual_class="id_cd"/>
</navigational_class>
          <!-- ============== Classe navegacional Artista ============== -->
<navigational_class id="no_artista" name="artista">
     <nav_attrib name="chave_artista" conceptual_class="id_artista"/>
     <nav_attrib name="nome" conceptual_class="id_artista"/>
     <perspective_attrib name="descricao" perspective="descricao_artista"
                                    conceptual_attrib="descricao_artista"
                                     conceptual_class="id_artistas"/>
     <perspective_attrib name="foto" perspective="foto_artista"
                                    conceptual_attrib="descricao_artista"
                                     conceptual_class="id_artistas"/>
     <nav_attrib_index name="indice_cds"  index="indice_cds_por_artista_H"/>
</navigational_class>
         ...
              <!-- ============== Elo Artista grava CD ============== -->
<link id="link_grava" name="artista_grava_cd"
       source_class="no_artista" target_class="no_cd"
       source_cardinality="n" target_cardinality="n">
</link>
         ...
        <!-- ============== Contexto CDs por ano de gravação ============== -->
<navigational_context id="contexto_cds_por_ano" name="cds_por_ano"
                          element_class="no_cd"   type="static"  navigation_type="sequential">
     <selection>
        <equal>
           <attribute name="ano_gravacao" navigational_class="no_cd"/>
           <nav_parameter>ano</nav_parameter>
        </equal>
     </selection>
     <order id="ordenacao_cds_por_ano">
          <order_element name="ano_gravacao" navigational_class="no_cd" criteria="asc"/>
     </order>
</navigational_context>
         …
 <!-- ========== Indice dos elementos do contexto CDs por ano de gravação ======== -->
<index  id="indice_cds_por_ano" name="cds_por_ano_idx" type="static">
     <element_context context="contexto_cds_por_ano"/>
     <shown_attrib  name="ano_gravacao" navigational_class="no_cd"/>
     <dynamic_selector>
        <shown_attrib  name="nome" navigational_class="no_cd"/>
        <selector_destination  target="contexto_cds_por_ano"/>
     </dynamic_selector>
</index>
         …
<landmark target="indice_cds_por_ano">CDs por ano de gravacao</landmark>
         ...
```

Figura 2.23 – Especificação OOHDM-ML do modelo de navegação para vendas de CDs

O elemento *interface_model* descreve o modelo de interface abstrata do método OOHDM. Seu conteúdo será definido em versões posteriores da linguagem OOHDM-ML.

O elemento opcional *OOHDM_WEB*, apresentado na figura 2.9, contém as marcações que descrevem o mapeamento das classes e relacionamentos do modelo conceitual, e das classes em contexto do modelo de navegação, para as tabelas de banco de dados que serão usadas pelo ambiente OOHDM-Web 2.0. Este elemento contém, ainda, o elemento *interface_element* que descreve os objetos de interface que serão utilizados na aplicação. A figura 2.24 ilustra o conteúdo do elemento OOHDM_WEB.

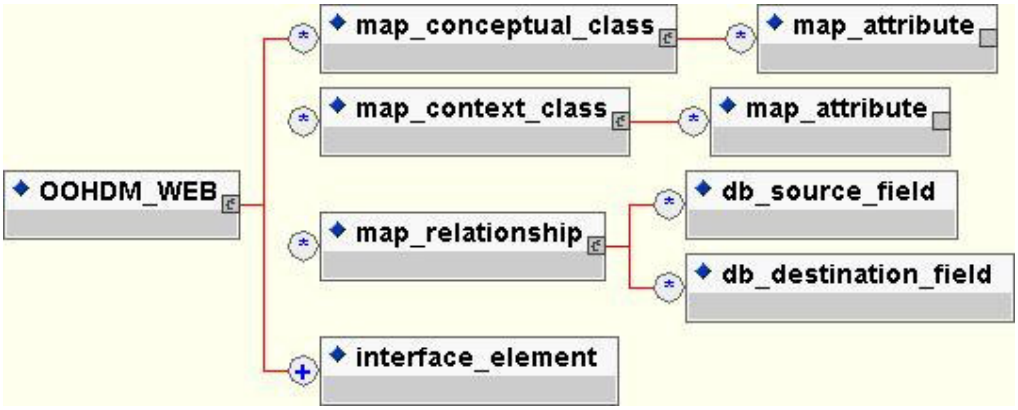


Figura 2.24 – Conteúdo do elemento OOHDM_WEB

No ambiente OOHDM-Web cada classe do modelo conceitual é traduzida para uma tabela de classe no banco de dados da aplicação. Esta tabela pode ser criada com o mesmo nome e atributos definidos para a classe conceitual ou pode ser criada com base no mapeamento da classe para o banco de dados, que pode definir um novo nome para a tabela de classe e seus atributos. Este mapeamento é descrito pelo elemento *map_conceptual_class* que possui o atributo **db_class,** que especifica o nome que será usado para criar a tabela de classe no banco de dados, e o elemento *map_attribute* para descrever o mapeamento dos atributos. O elemento *map_attribute* possui o atributo **db_attribute,** que especifica o nome que será usado para criar o atributo na tabela de classe, o atributo **primary_key**, que especifica que o atributo criado faz parte da chave da tabela, e o atributo **db_type** que especifica um tipo válido para o atributo no banco de dados.

As classes em contexto especificadas no modelo de navegação também são traduzidas para tabelas de classe no banco de dados e podem ser mapeadas. O mapeamento para as classes em contexto é descrito pelo elemento *map_context_class.* Este elemento possui o atributo **db_class_ctx,** que especifica o nome que será usado para criar a tabela de classe no banco de dados, e o elemento *map_attribute* para descrever o mapeamento de seus atributos.

O elemento *map_relationship* descreve o mapeamento dos relacionamentos do modelo conceitual para as tabelas do banco de dados. Este mapeamento deve ser feito da seguinte forma: se a cardinalidade do relacionamento é 1-1, o relacionamento é mapeado em uma das classes participantes do relacionamento; se a cardinalidade é 1-n, o relacionamento é mapeado para um atributo na classe do lado n; e se a cardinalidade for n-n, uma nova tabela deverá ser criada para conter o relacionamento.

O elemento *map_relationship* possui o atributo **db_relationship** que especifica o nome da tabela do banco de dados que contém o relacionameno mapeado. Os atributos chave para esta tabela podem ser mapeados usando os elementos *db_source_field* e *db_destination_field*.

O elemento *db_source_field* especifica o nome do atributo chave da classe origem do relacionamento que será usado na tabela do banco de dados que contém este relacionamento. Este elemento possui um único atributo opcional chamado **map_attribute**,

cujo valor deve ser o identificador de um elemento *map_attribute*. Este atributo indica que a chave da classe origem é composta por mais de um atributo e que o atributo informado foi definido como chave através da especificação do valor "yes" para o atributo primary_key do elemento map_attribute.

O elemento *db_destination_field* especifica o nome do atributo chave da classe destino do relacionamento para a tabela do banco de dados que contém este relacionamento. Também possui o atributo opcional map_attribute para indicar que a chave da classe destino é composta por mais de um atributo.

O elemento *interface_element* descreve o nome de um elemento de interface que será usado por um objeto de navegação da aplicação implementada. Possui um único atributo chamado **target**, cujo conteúdo é uma lista dos identificadores dos objetos de navegação para os quais o elemento de interface está sendo definido. Este atributo indica que um mesmo objeto de interface pode ser usado por mais de um objeto de navegação.

A figura 2.25 ilustra parte do documento cds_oohdmweb.xml, entidade externa referenciada na figura 2.10, que contém o conteúdo do elemento *OOHDM_WEB* para o exemplo do domínio de vendas de CDs.

```
                    <!--  Mapeamento da classe conceitual Artista -->
<map_conceptual_class conceptual_class="id_artista">
  <map_attribute id="map_foto_artista" name="foto_artista" db_attribute="foto"/>
  <map_attribute id="map_descricao_artista" name="descricao_artista"
                 db_attribute="descricao"/>
  <map_attribute id="map_ano_nasc_artista" name="ano_nascimento"
                 db_attribute="ano_nasc_artista"/>
</map_conceptual_class>
        ...
        <!-- Mapeamento do relacionamento Artista grava CD -->
<map_relationship relationship="relacao_grava" db_relationship="artista_grava_cd">
  <db_source_field>chv_artista</db_source_field>
  <db_destination_field>chv_cd</db_destination_field>
</map_relationship>
        ...
    <!-- Mapeamento dos objetos de interface para os objetos de navegacao -->
<interface_element target="no_cd">cd.html</interface_element>
<interface_element target="no_artista">artista.html</interface_element>
        ...
```

Figura 2.25 – Especificação OOHDM–ML do mapeamento para o ambiente OOHDM-Web

Neste documento, o elemento *map_conceptual_class* descreve o mapeamento da classe conceitual *artista*. Como o seu atributo *db_class* não foi especificado, a tabela de classe no banco de dados será criada com o mesmo nome da classe conceitual. Esta tabela terá os mesmos atributos da classe conceitual *artista*, no entanto, os atributos *foto_artista*, *descricao_artista* e *ano_nascimento* serão criados na tabela de classe com os nomes especificados no atributo *db_attribute* dos elementos *map_attribute* que são *foto*, *descricao* e *ano_nasc_artista*. O elemento *map_relationship* descreve o mapeamento do relacionamento *artista_grava_cd*. Como este relacionamento tem cardinalidade n-n, será criada uma nova tabela no banco de dados, que receberá o mesmo nome do relacionamento (*atributo db_relationship*). A chave dessa tabela de relacionamento será formada pelos campos *chv_artista* e *chv_cd* mapeados, respectivamente, nos elementos *db_source_field* e *db_destination_field*. Por fim, foram mapeados os elementos de interface *cd.html* e *artista.html* (*elemento interface_element*) para apresentar os atributos das classes navegacionais *cd* e *artista*, respectivamente.

3 O Ambiente OOHDM-XWeb

O OOHDM-XWeb é um ambiente de desenvolvimento criado para apoiar a implementação de aplicações hipermídia, projetadas de acordo com as primitivas do método OOHDM, a partir de sua especificação OOHDM-ML.

A especificação OOHDM-ML de uma aplicação hipermídia é um documento XML, válido de acordo com a DTD OOHDM-ML, que contém a especificação declarativa do projeto OOHDM da aplicação.

O ambiente OOHDM-XWeb tem como objetivo gerar de forma automática a descrição do projeto OOHDM de uma aplicação hipermídia para o ambiente OOHDM-Web 2.0. A figura abaixo mostra a arquitetura do ambiente OOHDM-XWeb.

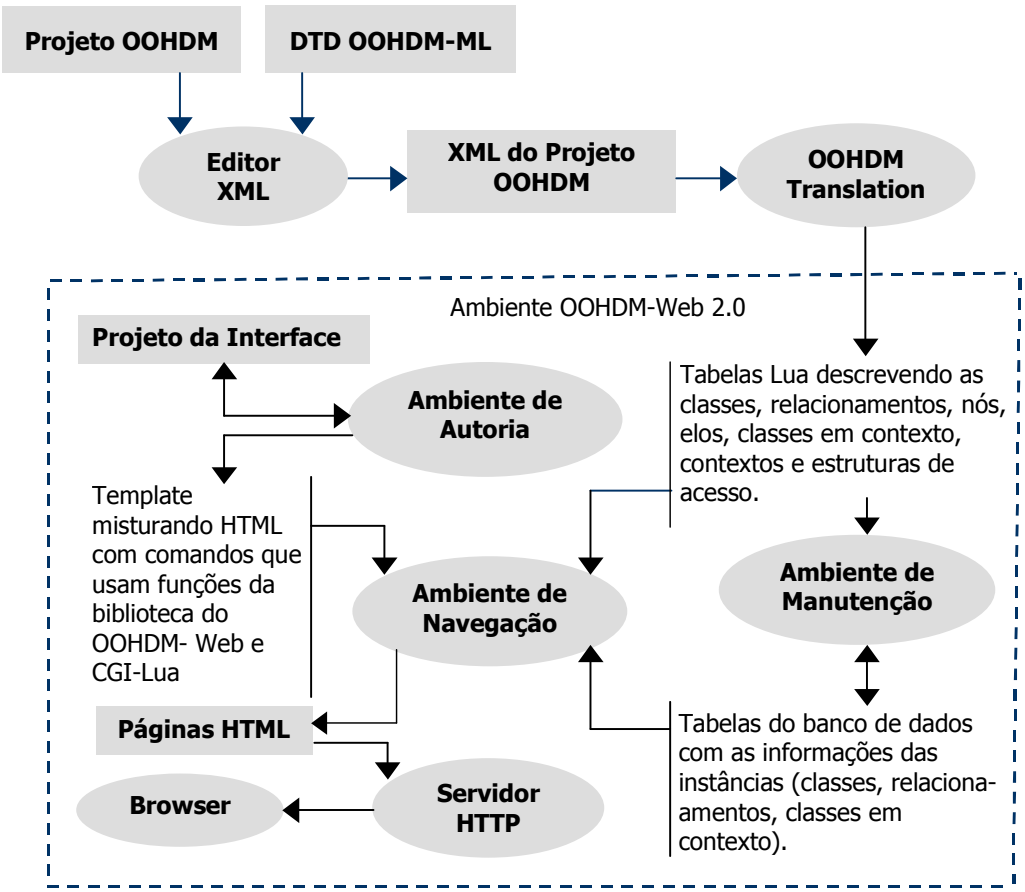


Figura 3.1 - Arquitetura do ambiente OOHDM-XWeb

O ambiente OOHDM-XWeb é composto por um programa chamado OOHDM-Translation e pelo sub-ambiente OOHDM-Web 2.0.

O OOHDM-Translation é o programa responsável pela geração automática da descrição do projeto OOHDM de uma aplicação hipermídia para o ambiente OOHDM-Web 2.0.

O ambiente OOHDM-Web 2.0 é a nova versão do ambiente de desenvolvimento OOHDM-Web, utilizada neste trabalho para a implementação das aplicações hipermídia apresentadas como exemplo.

No ambiente OOHDM-XWeb, o projetista especifica os modelos que compõem o projeto OOHDM da aplicação hipermídia de acordo com a gramática definida na DTD OOHDM-ML. Essa especificação pode ser feita com o auxílio de um editor XML que fornece uma interface amigável e facilita a construção de um documento XML válido. O documento XML criado servirá de entrada para o programa OOHDM-Translation, responsável pela criação das tabelas Lua utilizadas pela biblioteca de funções do ambiente OOHDM-Web 2.0.

Neste capítulo, apresentamos inicialmente um resumo da especificação da linguagem XSLT, utilizada neste trabalho para a criação do programa OOHDM-Translation. Em seguida, apresentamos o ambiente de desenvolvimento OOHDM-Web 2.0 e a especificação do projeto OOHDM-Web que ele utiliza. Por fim, apresentamos o programa OOHDM-Translation.

3.1 A Linguagem XSLT

A linguagem XSLT é uma linguagem declarativa para transformar documentos XML em outros documentos. Ela foi projetada para ser usada como parte da linguagem XSL (Extensible Stylesheet Language) [W3C 2000a] , que é uma linguagem de folha de estilo para XML.

Durante o desenvolvimento da linguagem XSL, tornou-se claro que o processo de converter um documento XML em algum outro formato possuía dois estágios completamente independentes:

- o estágio de transformação estrutural, no qual os dados são selecionados, agrupados e reordenados em uma estrutura que reflete a saída desejada e
- o estágio de formatação, no qual a nova estrutura é traduzida para um formato de apresentação tal como HTML ou PDF.

A transformação estrutural do documento é realizada pela linguagem XSLT e a linguagem XSL inclui um vocabulário XML para especificar a formatação. Assim, a linguagem XSL especifica o estilo de um documento XML usando a linguagem XSLT para descrever como o documento XML é transformado em um outro documento que usa o vocabulário de formatação.

A linguagem XSLT também foi projetada para ser usada de forma independente da linguagem XSL. No entanto, ela não foi criada como uma linguagem de transformação XML de propósito geral. Primeiramente, XSLT está projetada para os tipos de transformação que são necessários quando ela é usada como parte de XSL.

Antes do advento da linguagem XSLT, só era possível processar documentos XML escrevendo uma aplicação customizada (em uma linguagem procedural tal como C++, Visual Basic, ou Java, por exemplo). Esta aplicação definia uma sequência de passos a seguir para produzir a saída desejada. Na realidade, a aplicação não precisava analisar o documento XML cru, mas precisava chamar um parser XML, via uma API (Application Programing Interface) já definida, para obter a informação do documento e fazer alguma coisa com ela. Existem duas APIs principais para isto: a API SAX (Simple API for XML) [Megginson 2000] e o modelo DOM (Document Object Model) [W3C 2000b]. Seja qual for o parser utilizado, este processo tem o mesmo inconveniente: sempre que for necessário tratar um novo tipo de documento XML, será necessário criar um novo programa customizado, descrevendo uma sequência de passos diferente para processar o documento XML.

O projeto da linguagem XSLT foi baseado no reconhecimento de que esses programas são todos muito similares e por isto, deve ser possível descrever o que eles fazem usando uma linguagem declarativa de alto nível ao invés de escrever um programa do zero em C++, Visual Basic ou Java. A transformação requerida pode ser expressa como um conjunto de regras baseadas na definição de qual saída deve ser gerada quando padrões particulares

ocorrem no documento de entrada. XSLT é declarativa no sentido de que o desenvolvedor descreve a transformação desejada, ao invés de fornecer uma sequência de instruções procedurais para obtê-la.

A linguagem XSLT descreve a transformação requerida e conta com um processador XSL para decidir a forma mais eficiente de realizá-la. Ela conta ainda com um parser, que pode ser um parser DOM ou SAX, para converter o documento XML em uma estrutura de árvore. É a estrutura dessa representação em árvore do documento que XSLT manipula, não o próprio documento. A árvore é um tipo de dado abstrato. Não existe uma API ou uma representação de dados definida, apenas um modelo conceitual que define os objetos na árvore, suas propriedades e relacionamentos.

O processo de transformação realizado pela linguagem XSLT é transformar uma árvore origem em uma árvore resultado, como ilustra a figura 3.2.

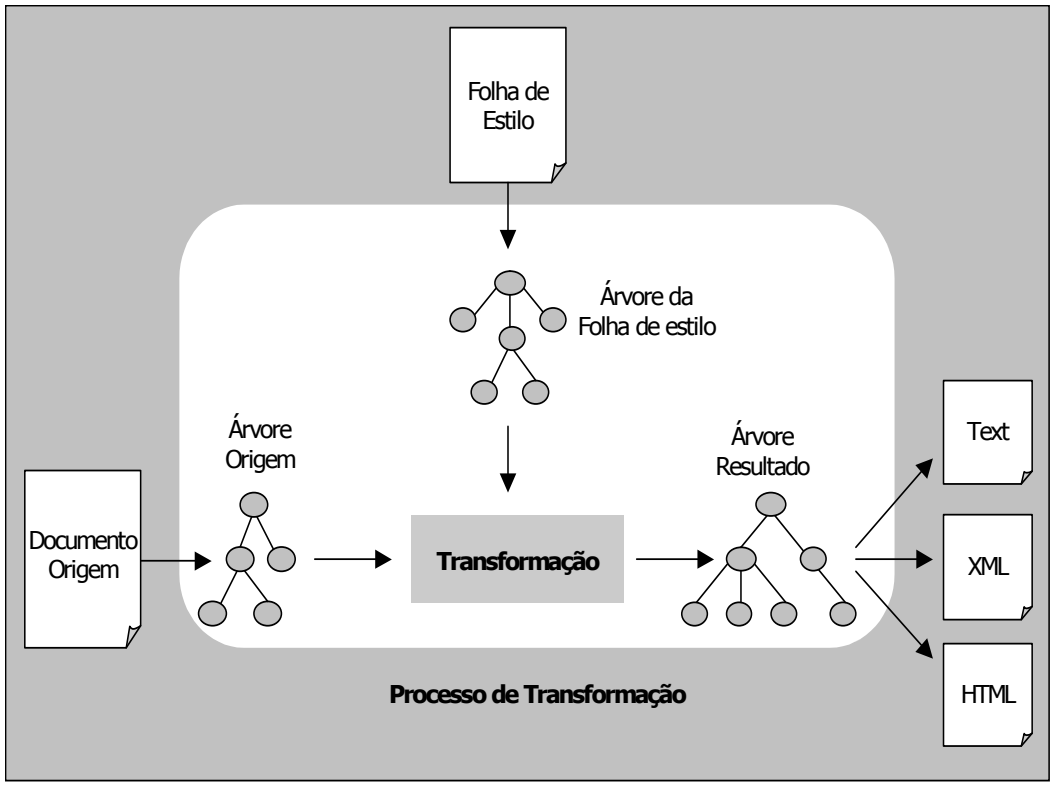


Figura 3.2 – Processo de Transformação XSLT

A figura 3.2 ilustra o processo de transformação realizado pela linguagem XSLT. Nele o documento XML origem é representado como uma árvore (árvore origem) que é transformada em uma árvore resultado sob o controle de uma folha de estilo. A árvore resultado representa o novo documento gerado que pode estar no formato texto, XML ou HTML.

A linguagem XSLT usa a linguagem de expressão definida por XPath (XML Path Language) [W3C 1999] para selecionar os nós da árvore origem que serão processados, especificar condições para diferentes maneiras de processar um nó e gerar o texto que será inserido na árvore resultado. Uma expressão XPath define o caminho de navegação através da árvore do documento. O resultado é sempre um conjunto de nós, que pode ser vazio ou conter apenas um nó, mas ele é sempre tratado como um conjunto. Por exemplo, a expressão *catalogo/produto/item[@cor="branca"]* poderia ser definida para navegar na árvore origem do documento XML apresentado na figura 2.2. Esta expressão seleciona qualquer elemento

item que tem um atributo *cor*, cujo valor é "*branca*", e um elemento pai *produto*, que por sua vez é filho do nó raiz *catalogo*.

A transformação expressa em XSLT é denominada ***folha de estilo*** (stylesheet). Este nome se deve ao tipo mais comum de transformação realizada usando XSLT, que é definir um estilo de apresentação para a informação contida no documento XML.

Uma folha de estilo é representada por um elemento <*xsl:stylesheet*> em um documento XML. Este elemento deve ter um atributo *version* indicando a versão da linguagem XSLT requerida pela folha de estilo. O elemento <*xsl:transform*> é permitido como sinônimo para <*xsl:stylesheet*>.

O elemento <*xsl:stylesheet*> frequentemente conterá declarações de namespace. A especificação XSLT usa o prefixo *xsl:* para referenciar elementos no namespace XSLT que tem a URI http://www.w3.org/1999/XSL/Transform, como ilustra a figura 3.3.

```
<?xml version="1.0"?>
<xsl:stylesheet
      xmlns:xsl="http: //www.w3.org/1999/XSL/Transform"
      version="1.0">
   <xsl:output method="html"/>
   <xsl:template match="/">
      <html>
        <body>
           <h1> Produtos do Catalogo  </h1>
           <table width="640">
             <xsl:apply-templates/>
           </table>
        </body>
      </html>
   </xsl:template>
   <xsl:template match="produto">
      <tr>
        <xsl:apply-templates/>
      </tr>
   </xsl:template>
   <xsl:template match="nome | preco">
      <td> <xsl:value-of select="."/> </td>
   </xsl:template>
</xsl:stylesheet>
```

Figura 3.3 – Folha de estilo para transformar o documento XML da figura 2.2

Todo elemento filho de um elemento <*xsl:stylesheet*> é denominado um elemento *top-level*. O principais elementos *top-level* de uma folha de estilo são:

- **Elemento <xsl:template>**

A estrutura de uma folha de estilo é formada basicamente por um conjunto de regras de template especificadas com o elemento <*xsl:template*>, como ilustra a figura 3.3.

Uma regra de template tem duas partes: um *pattern* que corresponde a um conjunto de nós da árvore origem e um *template* que pode ser instanciado para formar parte da árvore resultado. Isto permite que uma folha de estilo seja aplicável a uma variada classe de documentos que têm estruturas de árvore origem similares.

Um *pattern* especifica um conjunto de condições sobre um nó. Um nó que satisfaz às condições combina com o *pattern* e é processado.

Quando um *template* é instanciado, ele é sempre instanciado com respeito a um nó corrente e uma lista de nós correntes. Um *template* tipicamente contém instruções que selecionam uma lista adicional de nós da árvore origem para processar. O processo de selecionar, comparar e instanciar continua recursivamente até que nenhum novo nó seja selecionado para processamento.

O elemento <*xsl:template*> possui um atributo denominado *match*, cujo valor é um *pattern* que identifica o nó ou os nós da árvore origem para os quais uma regra se aplica. O atributo *match* é obrigatório a menos que o elemento <*xsl:template*> possua um atributo *name*. No exemplo da figura 3.3, o *pattern* "/" encontra o nó raiz *catalogo*; o *pattern* "produto" encontra os elementos *produto*; o *pattern* "nome | preco" encontra elementos *nome* ou elementos *preco*. Um outro *pattern* possível seria "produto/item" que encontraria os elementos *item* cujo pai é um elemento *produto*. O conteúdo do elemento <*xsl:template*> na folha de estilo é uma sequência de elementos e nós texto.

O exemplo apresentado na figura 3.3 contém três regras de template para transformar o documento XML de catálogo da figura 2.2. A primeira regra de template consiste de uma instrução (<*xsl:apply-templates*/>), alguns elementos de resultado literal (por exemplo, o elemento <*html*>) e um texto (*Produtos do Catalogo*).

Quando um template é instanciado, as instruções são executadas de acordo com as regras para cada instrução individual e os elementos de resultado literal e os nós texto são copiados para a árvore resultado.

Assim, quando o primeiro template da folha de estilo exemplo for instanciado, os elementos <*html*>, <*body*>, <*h1*>, <*table*> , etc... e o texto *Produtos do Catalogo* serão copiados da folha de estilo para a árvore resultado, e a instrução <*xsl:apply-templates/*> será executada. Quando escrita assim, sem quaisquer attributos, esta instrução significa "selecione todos os filhos do nó corrente na árvore origem e, para cada um deles, encontre a regra de template correspondente na folha de estilo e a instancie."

Neste exemplo, a instrução <*xsl:apply-templates*/> será chamada para processar todos os nós filhos do nó raiz <*catalogo*>, que é o nó corrente representado pelo *pattern* "/". Todos os filhos do nó raiz são elementos <*produto*>, portanto, eles serão processados pela regra de template cujo *pattern* é ***match="produto"***. Esta regra de template copia um elemento HTML <tr> e executa mais uma vez a instrução <*xsl:apply-templates/*> para processar os filhos do elemento <*produto*> na árvore origem.

Os filhos do elemento <*produto*> podem ser elementos <*nome*>, <*preco*> ou <*item*>. No entanto, apenas os elementos <*nome*> e <*preco*> correspondem ao template cujo *pattern* é *match="nome | preco"*. Portanto, o elemento *item* não será processado por essa folha de estilo. Essa última regra de template copia um elemento HTML <td> que ela preenche executando uma instrução <*xsl:value-of select="."/*>. Esta instrução avalia uma expressão XPath e escreve seu resultado (uma string) como um texto na árvore resultado. A expressão é "." que retorna o valor string do nó corrente, que é o conteúdo textual do elemento <*nome*> ou <*preco*> corrente.

O documento resultado gerado após o processamento dessa folha de estilo é um documento HTML, como ilustra a figura 3.4.

```
<html>
  <body>
    <h1> Produtos do Catalogo  </h1>
    <table width="640">
      <tr>
        <td>Camisa  Polo</td>
        <td>R$18,00</td>
      </tr>
      <tr>
        <td>Camisa  Social</td>
        <td>R$25,00</td>
      </tr>
    </table>
  </body>
</html>
```

Figura 3.4 – Documento html resultado da folha de estilo da figura 3.3

A instrução <*xsl:apply-templates*/> pode possuir um atributo *select* para processar nós selecionados por uma expressão ao invés de processar todos os filhos de um determinado elemento. O valor do atributo *select* é uma expressão, que deve resultar em um conjunto de nós (node-set).

Os elementos <*xsl:template*> e <*xsl:apply-templates*> possuem um atributo opcional denominado *mode*. O uso deste atributo permite que um elemento possa ser processado múltiplas vezes, produzindo um resultado diferente em cada uma delas. Se um elemento <*xsl:apply-templates*> possui um atributo *mode*, então ele aplica apenas as regras de template de elementos <*xsl:template*> que possuem um atributo *mode* com o mesmo valor. Se o elemento <*xsl:apply-templates*> não possui um atributo *mode*, ele aplica apenas as regras de template de elementos <*xsl:template*> que não possuem um atributo *mode*.

Templates podem ser chamados por nomes. Um elemento <*xsl:template*> que possui um atributo *name* especifica um "*template nomeado*" que pode, mas não precisa, ter um atributo *match*. Um elemento <*xsl:call-template*> é usado para chamar um template pelo nome. Ele possui um atributo *name* que identifica o template a ser chamado. Ao contrário do elemento <*xsl:apply-templates*>, o elemento <*xsl:call-template*> não altera o nó corrente ou a lista de nós corrente.

Como mencionado anteriormente, o conteúdo de um template também pode ser formado por nós texto. Nós texto podem ser criados em um template utilizando o elemento <*xsl:text*>. Cada nó texto em um template criará um nó texto com o mesmo valor-string na árvore resultado.

O elemento <*xsl:value-of*> também pode ser usado para criar nó texto em uma árvore resultado. Ele possui um atributo obrigatório que deve receber uma expressão. Esta expressão é avaliada e o objeto resultante é convertido para uma string. A string especifica o valor-string do nó texto criado. Se a string é vazia, nenhum nó texto será criado. O nó texto criado será combinado com quaisquer nós texto adjacentes. Este elemento também pode ser usado para computar texto gerado, por exemplo, extraindo texto de uma árvore origem ou inserindo o valor de uma variável.

- **Elemento <xsl:output>**

O elemento <*xsl:output*> é um elemento top-level usado para controlar o formato de saída da folha de estilo. Ele permite aos autores da folha de estilo especificar como a árvore resultado deve ser gravada no arquivo de saída. Normalmente, um processador XSLT não

precisa ser capaz de escrever a árvore resultado em um arquivo, mas se ele o fizer, deve gravar a árvore resultado conforme especificado pelo elemento <*xsl:output*>.

Este elemento possui um atributo denominado *method* que identifica o método global que deve ser usado para gravar a árvore resultado. Seu valor deve ser *xml*, *html* ou *text*. Por exemplo, a árvore resultado gerada pela folha de estilo apresentada na figura 3.2 será gravada em um arquivo html.

O valor default para o atributo *method* é escolhido da seguinte forma:

- se o nó raiz da árvore resultado tem um elemento filho,
- se o elemento documento da árvore resultado tem uma parte local chamada *html* (com qualquer combinação de letras maiúsculas ou minúsculas) e um namespace URI nulo, e
- quaisquer nós texto precedendo o primeiro elemento filho do nó raiz na árvore resultado contém apenas espaços em branco,

então, o valor default para o atributo *method* é *html*; senão, o valor default é *xml*.

O valor default para o atributo *method* deve ser usado quando não existe elementos <*xsl:output*> ou se nenhum dos elementos <*xsl:output*> especifica um valor para o atributo *method*.

- **Elementos <xsl:variable> e <xsl:param>**

Uma variavél é um nome que pode estar ligado a um valor (o valor da variável). O valor de uma variável pode ser um objeto de qualquer um dos tipos retornados por uma expressão. Os dois elementos que podem ser usados para declaração e atribuição de valor a variáveis são: <*xsl:variable*> e <*xsl:param*>. A diferença é que o valor atribuído à variável declarada pelo elemento <*xsl:param*> é apenas um valor default, ou seja, quando o template ou folha de estilo dentro do qual o elemento <*xsl:param*> ocorre é chamado, parâmetros podem ser passados e usados no lugar dos valores default.

Uma variável tem um escopo limitado ao elemento XSL onde ela está definida. Seu escopo começa em sua declaração e se extende até o final do elemento XSL onde ela foi definida.

O elemento *top-level* <*xsl:variable*> declara uma variável global que é visível em toda a folha de estilo. O elemento *top-level* <*xsl:param*> declara um parâmetro para a folha de estilo. No entanto, XSTL não define o mecanismo pelo qual os parâmetros são passados para a folha de estilo.

Os elementos <*xsl:variable*> e <*xsl:param*> também são permitidos dentro de templates. O elemento <*xsl:variable*> é permitido em qualquer lugar do template onde uma instrução é permitida. Já o elemento <*xsl:param*> é permitido apenas como um filho, declarado no início de um elemento <*xsl:template*>.

Os elementos <*xsl:variable*> e <*xsl:param*> tem um atributo obrigatório denominado *name*, que especifica o nome da variável. O valor da variável pode ser especificado de três maneiras:

- Se o elemento <*xsl:variable*> ou o elemento <*xsl:param*> tem um atributo *select*, então o valor do atributo deve ser uma expressão e o valor da variável é o resultado dessa expressão. Neste caso, o conteúdo do elemento deve ser vazio. No exemplo abaixo, a variável "*var_produto*" recebe a string retornada pela expressão XPath.

```
<xsl:variable name="var_produto" select="./produto/nome"/>
```

- Se o elemento <*xsl:variable*> ou o elemento <*xsl:param*> não tem um atributo *select* e tem um conteúdo não vazio, então o conteúdo do elemento especifica o valor da variável. No exemplo abaixo, a variável "*n*" recebe o valor 2.

```
<xsl:variable name="n">2</xsl:variable>
```

- Se o elemento <*xsl:variable*> ou o elemento <*xsl:param*> tem conteúdo vazio e não tem um atributo *select*, então o valor da variável é uma string vazia. Assim, <xsl:variable name="x"/> é equivalente a <xsl:variable name="x" select="" "'/>.

Os parâmetros são passados para templates usando o elemento <*xsl:with-param*>. O atributo obrigatório *name* especifica o nome do parâmetro e o valor do parâmetro é especificado da mesma forma que é feita para os elementos <*xsl:variable*> e <*xsl:param*>.

- **Elementos <xsl:include> e <xsl:import>**

Os elementos <*xsl:include*> e <*xsl:import*> são usados para combinar folhas de estilo e permitir a criação de uma folha de estilo em módulos. Ambos possuem um atributo *href* cujo valor é a referência URI que identifica a folha de estilo a ser incluída ou importada.

O elemento <*xsl:include*> funciona basicamente como uma macro-substituição: o elemento é substituído pelo conteúdo que ele referencia na folha de estilo principal. A inclusão é realizada no nível da árvore XML. O fato de que as regras de templates ou definições são incluídas não afeta o modo como elas são processadas.

O elemento <*xsl:import*> funciona da mesma forma que o elemento <xsl:include>, a folha de estilo que ele referencia também é macro-substituída, porém existe uma diferença. Com o elemento <xsl:include> o conteúdo que é macro-substituído dentro da folha de estilo tem a mesma precedência que o resto da folha de estilo. É como se existisse uma única folha de estilo. Com o elemento <xsl:import> o conteúdo que é macro-substituído dentro da folha de estilo tem uma precedência mais baixa que o resto da folha de estilo. Por exemplo, suponha que uma folha de estilo "A" importe as folhas de estilo "B" e "C", e que a folha de estilo "B" importa a folha de estilo "D", e a "C" importa a folha de estilo "E". A ordem de precedência para a importação da mais baixa para a mais alta é D, B, E, C, A, ou seja, as regras de template na folha de estilo "A" tem precedência maior do que as regras de template em C, E, etc. Os elementos <xsl:import> devem vir antes que qualquer elemento na folha de estilo principal.

A linguagem XSLT possui algumas instruções específicas para implementar repetição e processamento condicional.

A repetição é implementada pela instrução <*xsl:for-each*> que possui um atributo obrigatório denominado *select*. Esta instrução contém um template que é instanciado para cada nó selecionado pela expressão especificada pelo atributo *select*. O resultado da expressão deve ser um conjunto de nós (node-set). A folha de estilo da figura 3.3 poderia ser modificada para processar os elementos <*produto*> utilizando uma instrução de repetição, como ilustra a figura 3.5.

```
<?xml version="1.0"?>
<xsl:stylesheet
      xmlns:xsl="http: //www.w3.org/1999/XSL/Transform"
      version="1.0">
   <xsl:output method="html"/>
   <xsl:template match="catalogo">
      <html>
        <body>
           <h1> Produtos do Catalogo  </h1>
           <table width="640">
            <xsl:for-each select=”produto”>
              <tr>
              <xsl:for-each select=” nome | preco”>
                 <td> <xsl:value-of select="."/> </td>
              </xsl:for-each>
              </tr>
            </xsl:for-each>
           </table>
        </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

Figura 3.5 – Folha de estilo com instrução de repetição

Neste exemplo, os templates específicos para os elementos <*produto*>, <*nome*> e <*preco*> foram substituídos por duas instruções de repetição <*xsl:for-each*> que, neste caso, estão aninhadas. Estas instruções implementam o seguinte processamento: para cada elemento <*produto*>, filho do elemento raiz <*catalogo*>, processe todos os seus elementos filhos <*nome*> e <*preco*>. Uma vez que a primeira instrução de repetição é encontrada, o nó corrente e a lista de nós correntes passam a ser o resultado da expressão especificada no atributo *select*. Assim, não é necessário especificar o elemento pai na segunda instrução de repetição que processa os elementos <*nome*> e <*preco*>, filhos do nó corrente, que é o elemento <*produto*>.

Existem duas instruções em XSLT que suportam processamento condicional em um template: <*xsl:if*> e <*xsl:choose*>. A instrução <*xsl:if*> fornece uma estrutura condicional simples de *if-then* e a instrução <*xsl:choose*> implementa a seleção de uma escolha quando existem várias possibilidades.

O elemento <*xsl:if*> possui um atributo denominado *test*, que especifica uma expressão, e um template como conteúdo. A expressão é avaliada e o objeto resultado é convertido para um valor booleano. Se o resultado é verdadeiro, o conteúdo template é instanciado. Caso contrário, nada é criado. No seguinte exemplo, os nomes em um grupo de nomes são formatados como uma lista separada por vírgulas.

```
<xsl:template match=”lista_de_nomes/nome”>
      <xsl:apply_templates/>
      <xsl:if test=”not(position()=last())”>, </xsl:if>
</ xsl:template>
```

O elemento <*xsl:choose*> seleciona uma entre um número de alternativas possíveis. Ele consiste de uma sequência de elementos <*xsl:when*> seguidos por um elemento opcional <*xsl:otherwise*>. Cada elemento <*xsl:when*> possui um único atributo test que especifica uma expressão. O conteúdo dos elementos <*xsl:when*> e <*xsl:otherwise*> é um template. Quando um elemento <*xsl:choose*> é processado, cada um dos elementos <*xsl:when*> é

testado, avaliando a expressão e convertendo o objeto resultado em um valor booleano. O conteúdo do primeiro, e apenas do primeiro, elemento *<xsl:when>*, cujo teste é verdadeiro, é instanciado. Se nenhum elemento *<xsl:when>* retorna verdadeiro, o conteúdo do elemento *<xsl:otherwise>* é instanciado. Se este elemento não for especificado, nada é criado.

3.2 O Ambiente OOHDM-Web 2.0

Como mencionado anteriormente, o OOHDM-Web 2.0 é uma nova versão do ambiente OOHDM-Web.

O ambiente OOHDM-Web é um ambiente de desenvolvimento que fornece um maior nível de abstração aos projetistas de aplicações hipermídia baseadas no método OOHDM. Esse ambiente é baseado na linguagem Lua [Ierusalimschy 96] e no ambiente CGILua [Hester 97]. Com o OOHDM-Web, o projetista cria páginas (templates) que são uma mistura de HTML com chamadas a funções de uma biblioteca. Essas funções acessam (através do ODBC) os objetos navegacionais armazenados em um banco de dados relacional.

Em [Moura 1999] foi especificada uma forma de descrever o projeto OOHDM de uma aplicação hipermídia usando estruturas de dados da linguagem Lua. Essa descrição, denominada *projeto de navegação*, fica armazenada em um arquivo e é consultada pelas funções do OOHDM-Web sempre que é necessário obter informações a respeito de classes, relacionamentos, classes em contexto, contextos e índices.

O projeto de navegação, além de possuir estruturas que representam os modelos que compõem o projeto OOHDM da aplicação, também possui estruturas específicas que descrevem o mapeamento desses modelos para uma base de dados relacional. No entanto, esse mapeamento foi criado baseado-se na premissa de que a chave primária de uma tabela do banco de dados é criada como um identificador único, chamado *prim_key*.

Considerando que o documento XML que contém a especificação do projeto OOHDM da aplicação pode descrever esse mapeamento permitindo a criação das tabelas no banco de dados com chaves compostas, através da especificação do elemento OOHDM_WEB, foi necessário alterar a definição de algumas tabelas Lua do projeto de navegação e também as funções da biblioteca do ambiente OOHDM-Web, uma vez que praticamente todas elas utilizam o projeto de navegação para implementar as diversas funcionalidades da aplicação. Todas essas alterações deram origem à nova versão OOHDM-Web 2.0.

O ambiente OOHDM-Web 2.0 (Figura 3.1), assim como o ambiente OOHDM-Web, é composto por três ambientes:

- No primeiro (ambiente de autoria) é feita a descrição do projeto OOHDM da aplicação em um formato que pode ser entendido pelas funções da biblioteca. Portanto essa descrição será feita em tabelas da linguagem Lua. Nessas tabelas são especificadas todas as classes, classes em contexto e relacionamentos do esquema de classes navegacionais. Os contextos e estruturas de acesso representados no esquema de contextos (e especificados nos cartões) também são especificados. A partir dessa descrição, todas as tabelas de classes, classes em contexto e relacionamentos são criadas no banco de dados da aplicação através a interface OOHDM-Project.
- O segundo ambiente (ambiente de manutenção) possui uma interface chamada OOHDM-Data, onde a base de dados da aplicação criada pelo OOHDM-Project (no ambiente de autoria) é alimentada. Através dessa interface também é possível alterar dados ou remover instâncias existentes.
- No terceiro ambiente (ambiente de navegação), é realizada a criação das páginas que compõem a aplicação. Para isso, deve ser usada a biblioteca de funções do

OOHDM-Web 2.0. Através dessas funções é possível montar índices e exibir os elementos de todos os tipos de contexto, incluindo os atributos de classe em contexto. Também existem funções que controlam a navegação entre os elementos de um contexto e funções para a criação e manutenção de instâncias e relacionamentos.

Na próxima seção apresentamos a especificação do *projeto OOHDM-Web* do ambiente OOHDM-Web 2.0, que descreve a nova estrutura das tabelas Lua criada para descrever o projeto OOHDM de uma aplicação hipermídia.

Na seção 3.4 apresentamos um resumo da biblioteca de funções do ambiente OOHDM-Web e as principais alterações que deram origem a versão 2.0.

3.3 Especificação do Projeto OOHDM-Web

O projeto OOHDM-Web é a representação do projeto OOHDM de uma aplicação hipermídia em um formato que pode ser entendido pelas funções da biblioteca do ambiente OOHDM-Web 2.0. Além de possuir estruturas da linguagem Lua para representar todas as classes, relacionamentos, contextos e índices da aplicação, ele também possui estruturas específicas para representar o mapeamento das classes e dos relacionamentos para a base de dados da aplicação. Abaixo é apresentada a relação entre os elementos dos modelos OOHDM e as tabelas onde eles são especificados.

Esquema de Classes	**Tabela Lua** [4]
Classes	con_classes map_classes
Relacionamentos	con_relationships map_relationships
Classes em Contexto	queries map_classes_ctx

Esquema de Contextos	**Tabela Lua**
Contextos	nav_contexts
Índices	nav_indexes
Landmarks	nav_landmarks

3.3.1 Representação Gráfica das Estruturas

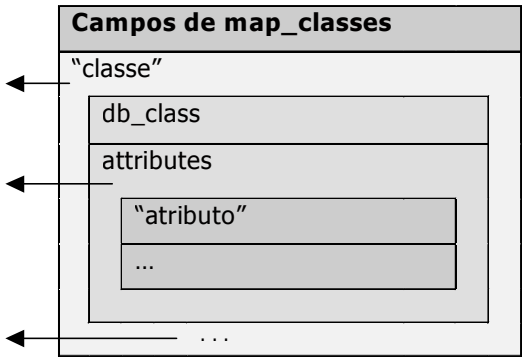
A especificação do projeto OOHDM-Web do ambiente OOHDM-Web 2.0 utiliza a mesma convenção gráfica criada em [Moura 1999] para representar cada tabela Lua do projeto de navegação do ambiente OOHDM-Web, onde tons diferentes de cinza mostram o aninhamento das tabelas. Se um campo possui um retângulo interno (com outro tom), esse campo é uma tabela composta pelos campos representados no retângulo interno. Veja, por exemplo, a estrutura da tabela *map_classes*.

[4] O nome das tabelas e de seus campos estão em inglês. No Apêndice, existe um glossário relacionando todos os nomes em inglês aos correspondentes em português.

Quando o nome do campo está entre aspas, ele deve ser substituído por um valor. Ex: "classe" significa que o nome desse campo é o nome de uma classe conceitual. Esse campo é uma tabela composta pelos campos do retângulo interno, ou seja, *db_class* e *attributes*.

attributes é de fato o nome desse campo, pois não está entre aspas. Esse campo também é uma tabela.

As reticências indicam que o campo representado neste retângulo pode ocorrer várias vezes. Ex: Podem haver várias classes dentro da tabela map_classes. Isto é possível porque o nome do campo é variável (cada classe tem um nome diferente). O mesmo acontece com o campo "atributo". Podem existir vários atributos dentro da tabela attributes.



3.3.2 Tabela Lua con_classes

Essa tabela representa as classes conceituais e seus atributos. É composta por várias tabelas, cada uma representando uma classe conceitual. A tabela *con_classes* terá a seguinte estrutura:

Nome do campo	Significado
"classe"	Nome da classe conceitual.
"atributo"	Nome de cada atributo da classe.
attrib_type	Tipo do atributo. Por exemplo: text, memo, number.
attrib_size	Tamanho do atributo.
prim_key	Indica se o atributo é um atributo chave. Seu valor deve ser "yes" ou "no".
...	
...	

O esquema conceitual para o domínio de vendas de CDs, apresentado no capítulo anterior, possui as classes conceituais *pessoa*, *cliente*, *artista*, *pedido*, *item*, *cd* e *música*. Para representar essas classes a tabela *con_classes* é criada da seguinte forma:

```
con_classes =
  {
   cd = {...},
   artista = {...},
   musica = {...},
   cliente = {...},
   pedido = {...}
  }
```

- A superclasse pessoa no relacionamento de generalização/especialização não precisa ser descrita no projeto OOHDM-Web, pois não será mapeada para uma tabela no banco de dados da aplicação.

Para cada classe são definidos seus atributos. Por exemplo, a classe *artista* possui os atributos *chave_artista*, *nome*, e_mail, *descrição* e *ano_nascimento*.

```
artista =
  {
   chave_artista = {...},
   nome = {...},
   e_mail = {...},
   descrição = {...},
   foto_artista = {...},
   ano_nascimento = {...}
  }
```

- Os atributos *nome* e *e_mail* foram herdados da superclasse pessoa e a perspectiva foto do atributo descrição foi mapeada para um novo atributo da classe chamado foto_artista. O atributo *chave_artista* foi criado para conter a chave primária na tabela de classe que será criada no banco de dados.

Cada atributo será representado por uma tabela que indica o tipo do atributo, seu tamanho e se o atributo é um atributo chave. Por exemplo, o atributo *chave_artista* da classe artista possui as seguintes características:

```
chave_artista =
  {
   attrib_type = "text",
   attrib_size = 70,
   prim_key = "yes"
  }
```

Em *attrib_type* deve ser informado o tipo do atributo. Deve ser um tipo válido para o banco de dados usado, por exemplo Access.

Em *attrib_size* deve ser informado o tamanho máximo que esse atributo pode ter. Caso o atributo não tenha um tamanho definido, este campo pode receber o valor *nil* ou não ser representado na tabela.

Em *prim_key* deve ser informado se o atributo é um atributo chave ou não. Caso o atributo não seja um atributo chave, este campo pode receber o valor "*no*" ou não ser representado na tabela.

Abaixo é mostrada a estrutura da tabela con_*classes* para descrever as classes *cd* e *artista*:

```
con_classes =
  {
   cd =     ------------------------> Nome de uma classe
     {
      chave_cd =   ------------------------> Nome de um atributo
        {
         attrib_type = "text",
         attrib_size = 20,
         prim_key = "yes"
        },
      nome =
        {
         attrib_type = "text",
         attrib_size = 70
```

```
    },
   descricao =
    {
     attrib_type = "memo"
    },
   ano_gravacao =
    {
     attrib_type = "text",
     attrib_size = 4
    },
   preco =
    {
     attrib_type = "integer"
    },
   disponibilidade =
    {
     attrib_type = "text",
     attrib_size = 20
    },
   capa =
    {
     attrib_type = "text",
     attrib_size = 30
    },
   procedencia =
    {
     attrib_type = "text",
     attrib_size = 30
    },
   gravadora =
    {
     attrib_type = "text",
     attrib_size = 70
    },
   genero =
    {
      attrib_type = "string",
      attrib_size = 70
    }
  },
artista =
  {
   chave_artista =
    {
     attrib_type = "text",
     attrib_size = 20,
      prim_key = "yes"
    },
   nome =
    {
     attrib_type = "text",
     attrib_size = 50
    },
   e_mail =
    {
     attrib_type = "text",
     attrib_size = 50
    },
   descricao =
```

```
        {
         attrib_type = "memo"
        },
      foto_artista =
        {
         attrib_type = "text",
         attrib_size = 30
        },
      ano_nascimento =
        {
         attrib_type = "text",
         attrib_size = 4
        }
    }
  }
```

3.3.3 Tabela Lua map_classes

Essa tabela mostra como fazer o mapeamento das classes (representadas na tabela Lua *con_classes*) nas tabelas de classe do banco de dados. Todas as classes existentes na tabela *con_classes* também devem existir na tabela *map_classes*.

Nome do campo	Significado
"classe"	Nome que identifica a classe (deve ser o mesmo nome que a classe recebeu em *con_classes*). É uma tabela que contém o campo db_class e os vários atributos dessa classe.
db_class	Nome da tabela de classe que representa essa classe no BD.
attributes	Tabela Lua com o mapeamento dos atributos Lua para os atributos da tabela de classe.
"atributo"	Nome Lua de cada atributo dessa classe. O valor desse campo deve ser o nome do atributo na tabela de classe do banco de dados.
...	
...	

Para o exemplo mostrado acima, a tabela map_classes seria a seguinte:

```
map_classes =
  {
   cd =
    {
     db_class = "cds"
    },
   artista =
    {
     db_class = "artistas",
     attributes =
       {
        foto_artista = "foto",
        ano_nascimento = "ano_nasc_artista"
       }
    }
```

```
}
```

☞ Caso o nome de um atributo na tabela do banco de dados seja o mesmo nome do atributo na tabela Lua *con_classes*, não é necessário representar esse atributo na tabela de mapeamento *map_classes*. As classes sempre devem ser representadas, ainda que todos os seus atributos tenham o mesmo nome no banco de dados, como por exemplo a classe *cd* no exemplo acima.

Para cada classe representada na tabela *map_classes* será criada uma tabela no banco de dados, com o nome dado em *db_class* e com os campos mapeados em *attributes*. O tipo e tamanho dos campos são obtidos na tabela *con_classes*. No exemplo acima as tabelas de classe **cds** e **artistas** são criadas no banco de dados.

3.3.4 Tabela Lua con_relationships

Essa tabela representa os relacionamentos entre as classes conceituais. É composta por várias tabelas identificadas pelo nome do relacionamento.

Nome do campo	Significado
"nome_relacionamento"	Nome que identifica cada relacionamento.
source_class	Nome da classe origem do relacionamento. Essa classe deve estar definida na tabela Lua con_classes.
destination_class	Nome da classe destino do relacionamento. Essa classe deve estar definida na tabela Lua con_classes.
source_card	Cardinalidade da classe origem do relacionamento. Pode ter o valor "1" ou "n".
dest_card	Cardinalidade da classe destino do relacionamento. Pode ter o valor "1" ou "n".
...	

Por exemplo, a tabela *con_relationships* que contém os relacionamentos entre as classes *cd* e *artista* é a seguinte:

```
con_relationships =
  {
   artista_grava_cd =
     {
      source_class = "artista",
      destination_class = "cd",
      source_card = "n",
      dest_card = "n"
     },
   artista_compoe_com_artista =
     {
      source_class = "artista",
      destination_class = "artista",
      source_card = "n",
      dest_card = "n"
     }
  }
```

Nesse exemplo só existe um relacionamento entre as classes *cd* e *artista* e um auto-relacionamento. Todos os relacionamentos de todas as classes devem estar representados na tabela *con_relationships*.

3.3.5 Tabela Lua map_relationships

A tabela *map_relationships* mostra como fazer o mapeamento dos relacionamentos (representados na tabela Lua *con_relationships*) nas tabelas de relacionamento do banco de dados. Todos os relacionamentos existentes na tabela *con_relationships* também devem existir na tabela *map_relationships*.

Nome do campo	Significado
"nome_relacionamento"	Nome que identifica cada relacionamento.
db_relationship	Nome da tabela do banco de dados que contém o relacionamento.
db_source_field	Tabela Lua com o mapeamento dos atributos chave da classe origem do relacionamento para os campos de db_relationship.
"atributo" ...	Nome Lua de cada atributo chave da classe origem. O valor desse campo deve ser o nome do atributo na tabela de relacionamento do banco de dados.
db_destination_field	Tabela Lua com o mapeamento dos atributos chave da classe destino do relacionamento para os campos de db_relationship.
"atributo" ...	Nome Lua de cada atributo chave da classe destino. O valor desse campo deve ser o nome do atributo na tabela de relacionamento do banco de dados.
...	

Os relacionamentos da tabela *con_relationships* seriam mapeados da seguinte forma:

```
map_relationships =
  {
   artista_grava_cd =
     {
      db_source_field =
        {
         chave_artista = "chv_artista"
        },
      db_destination_field =
        {
         chave_cd = "chv_cd"
       },
      db_relationship = "artista_grava_cd"
     },
   artista_compoe_com_artista =
     {
      db_source_field =
        {
         chave_artista = "chv_artista_orig"
       },
      db_destination_field =
```

```
        {
         chave_artista = "chv_artista_dest"
        },
      db_relationship = "artista_compoe_com_artista"
    }
  }
```

- Caso os atributos da tabela de relacionamento tenham os mesmos nomes dos atributos chave da classe origem ou da classe destino do relacionamento, as tabelas *db_source_field* e *db_destination_field* não precisam ser representadas na tabela map_classes.

Todos os relacionamentos especificados na tabela *con_relationships* devem ser representados na tabela *map_relationships*.

Se a cardinalidade do relacionamento for 1-1, o relacionamento é mapeado para um atributo em uma das classes participantes do relacionamento. Esse atributo deve conter a chave da outra classe. Neste caso, *db_relationship* contém o nome da tabela de classe do banco de dados que recebe o atributo. O nome dado a esse atributo e o nome do atributo chave dessa tabela são especificados em *db_source_field* e *db_destination_field.* Esse atributo será criado apenas na tabela de classe do banco de dados. A tabela de classe de *con_classes* não será alterada.

Por exemplo, poderia existir uma relação entre o artista e seu mais novo cd:

```
con_relationships =
   {
   artista_grava_novo_cd =
     {
      source_class = "artista",
      destination_class = "cd",
      source_card = "1",
      dest_card = "1"
     },
   }
map_relationships =
   {
   artista_grava_novo_cd =
     {
      db_source_field = "chave_artista",
      db_destination_field = "chv_novo_cd",
      db_relationship = "artistas"
     },
   }
```

A tabela de classe **artistas** será usada para conter o relacionamento no banco de dados. Ela conterá os seguintes campos:

#chave_artista	nome	e_mail	descricao	foto	ano_nasc_artista	**chv_novo_cd**

onde *chv_novo_cd* conterá os valores do campo *chave_cd* da tabela de classe **cds**.

Se a cardinalidade for 1-n, o relacionamento é mapeado para um atributo na classe do lado n. Esse atributo deve conter a chave da classe do lado 1. Neste caso, *db_relationship* contém o nome da tabela de classe do lado n (que recebe o atributo). O nome dado a esse atributo e o nome do atributo chave da tabela de classe do lado n são especificados em *db_source_field* e *db_destination_field.*

Por exemplo, suponhamos que o relacionamento entre as classes artista e cd tivesse cardinalidade 1-n:

```
con_relationships =
   {
   artista_grava_cd =
     {
      source_class = "artista",
      destination_class = "cd",
      source_card = "1",
      dest_card = "n"
     }
   }

map_relationships =
   {
   artista_grava_cd =
     {
      db_source_field = "chv_artista",
      db_destination_field = "chave_cd",
      db_relationship = "cds"
     }
   }
```

A tabela de classe **cds** será usada para conter o relacionamento no banco de dados. Ela conterá os seguintes campos:

#chave_cd	nome	descricao	ano_gravacao	disponibilidade	...	**chv_artista**

onde *chv_artista* conterá os valores do campo chave_artista da tabela de classe **artistas**.

Se a cardinalidade do relacionamento for n-n, a tabela que contém o relacionamento não pode ser uma tabela de classe. Deve ser uma nova tabela que conterá somente o relacionamento. Para cada uma dessas tabelas de relacionamento, será criada uma tabela no banco de dados cujo nome será o valor de *db_relationship*. Os campos dessa tabela de relacionamento conterão as chaves dos elementos relacionados. Os nomes desses campos serão os indicados nas tabelas *db_source_field* e *db_destination_field*. Caso estas tabelas não sejam representadas na tabela map_relationships, os nomes dos campos serão os mesmos dos atributos chave da classe origem e da classe destino do relacionamento.

Para o relacionamento artista_grava_cd com cardinalidade n-n, mostrado anteriormente, será criada uma nova tabela chamada **artista_grava_cd**. Essa tabela conterá os seguintes campos:

#chv_artista	#chv_cd

onde *chv_artista* conterá os valores do campo *chave_artista* da tabela de classe **artistas** e *chv_cd* conterá os valores do campo *chave_cd* da tabela de classe **cds**.

Para o auto-relacionamento artista_compoe_com_artista, também será criada uma nova tabela, **artista_compoe_com_artista**:

#chv_artista_orig	#chv_artista_dest

onde os dois campos conterão os valores do campo *chave_artista* da tabela de classe **artistas**, representando o relacionamento entre artistas diferentes.

Todas as tabelas de relacionamento do ambiente OOHDM-Web 2.0 devem ser criadas a partir da tabela Lua *con_relationships*, mas essas tabelas não são preenchidas nesse momento.

3.3.6 Tabela Lua queries

Cada nó do esquema de classes navegacionais deve ser representado por uma *query* que obtém todos os atributos que pertencem ao nó. No exemplo sobre vendas de CDs, temos três classes de navegação: cd, artista e música. Cada classe é representada por uma *query*, definida na tabela Lua *queries*.

A tabela *queries* também representa as classes em contexto definidas no modelo de navegação para uma determinada classe navegacional base. O conteúdo da estrutura da tabela *queries* é o seguinte:

Nome do campo	Significado
"nome_classe"	Nome que identifica a classe navegacional (o nó).
class	Lista das classes conceituais das quais são derivados os atributos da classe navegacional. Seu valor deve conter os nomes de classes especificadas em *con_classes*.
tables	Lista das tabelas que participam da query.
fields	Lista dos campos obtidos pela query. Cada campo deve ter o formato "tabela.campo".
where	Conteúdo da cláusula WHERE da query, se existir. Caso não exista, este campo deve receber nil ou não ser representado na tabela.
class_ctx	Tabela com as classes em contexto dessa classe.
"contexto"	Nome de cada contexto no qual a classe possui classes em contexto. Cada contexto é uma tabela com os atributos da classe em contexto.
"atributo"	Nome de cada atributo da classe em contexto.
attrib_type	Tipo do atributo.
attrib_size	Tamanho do atributo.
. . .	
. . .	
. . .	
. . .	

No exemplo sobre vendas de CDs, o nó que exibe as informações sobre os artistas é representado da seguinte forma:

```
queries =
  {
   artista =
    {
     class = {"artista"},
     tables = {"artistas"},
     fields = {"artistas.nome", "artistas.descricao",
               "artistas.foto", "artistas.ano_nasc_artista"}
    }
  }
```

Repare que os índices não são incluídos em *fields*, somente os atributos. No nó *artista*, todos os atributos pertencem à tabela artistas. A query gerada pela descrição do nó *artista* é a seguinte:

```
SELECT artistas.nome, artistas.descricao, artistas.foto,
       artistas.ano_nasc_artista
FROM artistas
```

O nó que exibe as informações sobre os CDs é representado pela seguinte query:

```
queries =
  {
   arquiteto = {...},
    cd =
      {
       class = {"cd","artista","musica"},
       tables = {"cds", "artistas", "musicas", "artista_grava_cd",
                 "cd_possui_musica"},
       fields = {"cds.nome", "cds.descricao", "cds.ano_gravacao",
                 "cds.preco", "cds.disponibilidade", "cds.capa",
                 "cds.gravadora", "cds.genero", "artistas.nome",
                  "musicas.nome"},
       where = "cds.chave_cd = artista_grava_cd.chv_cd AND
                artistas.chave_artista = artista_grava_cd.chv_artista AND
                cds.chave_cd = cd_possui_musica.chv_musica AND
                musicas.chave_musica = cd_possui_musica.chv_musica"

      }
  }
```

O nó *cd* possui atributos das classes conceituais *artista* e *musica*, que estão relacionadas à classe *cd*. Portanto as tabelas de relacionamento *artista_grava_cd* e *cd_possui_musica* também devem fazer parte da query. A cláusula WHERE contém as condições necessárias para fazer o *join* das tabelas. Essa descrição gera o seguinte comando SQL:

```
SELECT cds.nome, cds.descricao, cds.ano_gravacao, cds.preco,
       cds.disponibilidade, cds.capa, cds.gravadora, cds.genero,
       artistas.nome, musicas.nome
FROM cds, artistas, musicas, artista_grava_cd, cd_possui_musica
WHERE cds.chave_cd = artista_grava_cd.chv_cd
AND   artistas.chave_artista = artista_grava_cd.chv_artista
AND   cds.chave_cd = cd_possui_musica.chv_musica
AND   musicas.chave_musica = cd_possui_musica.chv_musica
```

Essas queries são usadas na criação das páginas correspondentes a cada nó de navegação.

O campo *class_ctx* deve conter uma tabela para cada contexto nos quais essa classe possui classes em contexto. No exemplo, *a* classe cd só possui atributos de classe em contexto no contexto cd*s_por_ano*.

```
class_ctx =
  {
   cds_por_ano = {...}
  }
```

Cada contexto deverá ser representado por uma tabela que conterá os atributos da classe em contexto. Cada atributo será representado por uma tabela que indica o tipo do atributo e seu tamanho.

```
cds_por_ano =
   {
    local_gravacao =
      {
      attrib_type = "text",
      attrib_size = 50
      }
   }
```

A classe cd só possui um atributo de classe em contexto, que é o atributo *local_gravacao*, para o contexto *cds_por_ano*. Isso significa que quando um elemento da classe *cd* for exibido no contexto *cds_por_ano*, o atributo *local_gravacao* também será exibido, além dos demais atributos (*nome*, *descricao*, *ano_gravacao*, *etc*). Abaixo é mostrada a estrutura final da tabela *queries*:

```
queries =
  {
   artistas =
    {
     class = {"artista"},
     tables = {"artistas"},
     fields = {"artistas.nome", "artistas.descricao",
               "artistas.foto", "artistas.ano_nasc_artista"}
    },
   cd =
    {
     class = {"cd","artista","musica"},
     tables = {"cds", "artistas", "musicas", "artista_grava_cd",
               "cd_possui_musica"},
     fields = {"cds.nome", "cds.descricao", "cds.ano_gravacao",
               "cds.preco", "cds.disponibilidade", "cds.capa",
               "cds.gravadora", "cds.genero", "artistas.nome",
               "musicas.nome"},
     where = "cds.chave_cd = artista_grava_cd.chv_cd AND
              artistas.chave_artista = artista_grava_cd.chv_artista AND
              cds.chave_cd = cd_possui_musica.chv_musica AND
              Musicas.chave_musica = cd_possui_musica.chv_musica",

     class_ctx =
     {
      cds_por_ano =
       {
        local_gravacao =
        {
         attrib_type = "text",
         attrib_size = 50
        }
       }
     }
    }
  }
```

3.3.7 Tabela Lua nav_contexts

Todos os contextos representados no esquema de contextos devem ser especificados na tabela *nav_contexts*. Essa tabela é composta por várias outras tabelas, representadas pelo nome de cada contexto.

Nome do campo	Significado
"nome_ctx"	Nome que identifica o contexto de navegação. Cada contexto é uma tabela.
context	Indica se o contexto é estático ou dinâmico e se possui múltipla ordenação. Também indica se é um contexto arbitrário, um grupo de contexto, ou um contexto simples. Os valores que esse campo aceita estão especificados mais adiante.
nav_type	Tipo de navegação usada para percorrer os elementos do contexto. Os valores para esse campo também são pré-definidos e estão especificados mais adiante.
parameters	Lista com os nomes dos parâmetros que devem ser passados para a função que seleciona os elementos do contexto. Os parâmetros são usados para montar a condição da seleção. Caso não haja parâmetros esse campo deve ser = nil.
elements	Especifica os elementos que fazem parte do contexto. É uma tabela com as classes as quais pertencem os elementos do contexto.
"classe"	Nome que identifica cada classe.
order	Lista com os nomes dos atributos pelos os elementos do contexto pertencentes a essa classe são ordenados. Pode possuir o valor nil, para indicar que o contexto não tem ordem pré-definida (o acesso ao contexto é por índice, por exemplo). Essa lista deve ter o formato: nome_atrib = tipo_ord, por exemplo, {nome_obra = "ASC", chave_obra = "DESC"}.
dest_page	Página html onde os elementos do contexto pertencentes a essa classe serão mostrados.
selection	Tabela que especifica como são selecionados os elementos do contexto pertencentes a essa classe. É composta por várias tabelas, uma para cada condição a ser respeitada na seleção dos elementos.
link_name	Tabela lua que indica que os elementos do contexto são derivados de um ou mais relacionamentos (elos). Seu valor deve ser o nome Lua de um ou mais relacionamentos especificados na tabela *con_relationships*.
condition	Condição à qual os elementos devem obedecer. Formato: (atributo operador valor) AND / OR (atrib op val) …

. . .

. . .

. . .

O atributo *context* indica se o contexto é estático, dinâmico ou de sessão (que é sempre um contexto dinâmico); simples, arbitrário, por consulta ou grupo de contexto; com ordenação múltipla ou não. Os valores e significados desse atributo seguem abaixo:

Valor	Significado
E	Estático
D	Dinâmico
T	Sessão (Temporário)
S	Simples
A	Arbitrário
C	Por Consulta
G	Grupo de Contexto
M	Múltiplo Ordenado

O valor do atributo *context* será sempre uma combinação de 2 ou 3 letras. A primeira é E, D ou T. A segunda é S, A, C ou G. A terceira é M, que só existe se o contexto possuir ordenação múltipla. Portanto as combinações possíveis são: ES, ESM, DS, DSM, TS, TSM, EA, EAM, DA, DAM, TA, TAM, DC, DCM, TC, TCM, EG, EGM, DG, DGM, TG, TGM.

O atributo *nav_type* indica o tipo de navegação permitido entre os elementos do contexto. Os valores e significados do atributo seguem abaixo:

Valor	Significado
S	Navegação seqüencial
C	Navegação circular
I	Navegação limitada ao índice
L	Navegação livre
SI	Navegação seqüencial com índice
CI	Navegação circular com índice

Um exemplo bem simples é o contexto derivado de classe "cds em ordem alfabética". A tabela nav_contexts é criada da seguinte maneira:

```
nav_contexts =
  {
   cds_alfa =
    {
     context = "ES",
     nav_type = "C",
     elements =
      {
       cd =
        {
          order = {nome = "asc"},
          dest_page = "cd.html"
        }
      }
    }
  }
```

Neste contexto a tabela selection não precisa ser representada, pois não existe uma condição específica para obter os elementos do contexto (os elementos são todos os cds).

Acrescentando o contexto simples "cds gravados em 1991 ou 1992" temos:

```
nav_contexts =
  {
   cds_alfa = {...},
   cds_91_92 =
    {
     context = "ESM",    ---------> Múltipla Ordenação
     nav_type = "L",
     elements =
      {
       cd =
        {
         order = {ano_gravacao = "desc", nome = "asc"},
         dest_page = "cd.html",
          selection =
           {
            condition = "(ano_gravacao = '1991') OR "..
                        "(ano_gravacao = '1992')"
           }
        }
      }
    }
  }
```

O contexto *cds_91_92* é um contexto derivado de classe, com uma condição para selecionar seus elementos e tem múltipla ordenação. Dois atributos foram especificados no campo *order*. Cada vez que a função *index* do ambiente OOHDM-Web 2.0 for chamada para selecionar os elementos desse contexto, deve ser passado um parâmetro chamado ***ordering***, que indica qual dos atributos especificados em *order* vai ordenar os elementos.

Note que o nome dos atributos, tanto em *order* quanto em *condition*, deve ser sempre o nome especificado em *con_classes*. Os nomes especificados em *map_classes*, para os atributos do banco de dados (caso sejam diferentes dos nomes em *con_classes*) não devem ser usados para identificar atributos aqui na tabela n*av_contexts*.

O nome *ordering* é um nome de parâmetro reservado para contextos com múltipla ordenação e não pode ser usado para identificar outros parâmetros em outros tipos de contexto.

Acrescentando o grupo de contexto derivado de classe "cds por ano de gravação" temos:

```
nav_contexts =
  {
   cds_alfa = {...},
   cds_91_92 = {...},
   cds_por_ano =
    {
     context = "EG",      --------> Grupo de Contexto
     nav_type = "S",
     parameters = {"ano"},
     elements =
      {
       cd =
       {
        order = {ano_gravacao = "asc"},
        dest_page = "cd.html",
         selection =
```

```
          {
           condition = "(ano_gravacao = #ano)"
           }
       }
      }
    }
  }
```

Todo grupo de contexto deve possuir uma condição com um parâmetro. No exemplo acima, a condição *"(ano_gravacao = #ano)"* indica que o atributo pelo qual os elementos do contexto são agrupados é *ano_gravacao*. Quando um valor é passado para o parâmetro *ano*, o resultado é apenas um contexto do grupo de contextos, ou seja apenas os CDs gravados no ano especificado. Se o parâmetro não tiver um valor, o resultado será o grupo, que conterá todas os CDs agrupados pelo ano de gravação.

Note que pode haver mais de um parâmetro: *parameters = {"ano1", "ano2"}, condition = "(ano_gravacao = #ano1) OR (ano_gravacao = #ano2)"*, retornaria a união de dois contextos do grupo de contextos.

Todos os parâmetros devem ser sempre precedidos do caracter #.

Acrescentando o contexto arbitrário "cds e artistas favoritos" temos duas classes formando o contexto:

```
nav_contexts =
  {
   cds_alfa = {...},
   cds_91_92 = {...},
   cds_por_ano = {...},
   favoritos =
    {
     context = "EA",  ----> Contexto Arbitrário
     nav_type = "L",
     elements =
      {
       cd =
        {
         order = {nome = "asc"},
         dest_page = "cd.html",
         selection =
          {
           condition = "((chave_cd = '1') OR (chave_cd = '2') OR "..
                       (chave_cd = '5'))"
          }
        },
        artista =
         {
          order = {nome = "asc"},
          dest_page = "artista.html",
          selection =
           {
            condition = "((chave_artista = 'billy') OR "..
                         (nome = 'Skank'))"
           }
         }
      }
    }
  }
```

Os elementos do contexto arbitrário são enumerados aqui na definição do contexto. O contexto resultante é a união dos elementos selecionados na condição da classe *cd* e dos selecionados na condição da classe *artista*.

Com base no contexto mostrado acima poderíamos criar a versão baseada em sessão do contexto arbitrário "favoritos". A definição dos dois contextos é semelhante, mas no contexto persistente "favoritos", os elementos são enumerados na própria definição, enquanto que no contexto "favoritos_sessao", os elementos são determinados durante a sessão de navegação. Assim, a definição para o contexto "favoritos_sessao" não teria a descrição da tabela *selection* para as classes *cd* e *artista*.

Acrescentando o contexto derivado de elo "cds dos artistas nascidos em 1960":

```
nav_contexts =
  {
   cds_alfa = {...},
   cds_91_92 = {...},
   cds_por_ano = {...},
   favoritos = {...},
   cds_artistas_60 =
    {
     context = "ES",
     nav_type = "I",
     elements =
     {
      cd =
       {
        dest_page = "cd.html",
         selection =
          {
            link_name = {"artista_grava_cd"},
            condition = "(ano_nascimento='1960')"
          }
       }
     }
    }
  }
```

O contexto resultante é formado pelos elementos selecionados na condição realizada sobre o relacionamento entre as classes *cd* e *artista* indicado no campo link_name.

Acrescentando o grupo de contexto derivado de elo "cds por artista":

```
nav_contexts =
  {
   cds_alfa = {...},
   cds_91_92 = {...},
   cds_por_ano = {...},
   favoritos = {...},
   cds_artistas_60 = {...},
   cds_por_artista =
    {
     context = "EG",
     nav_type = "L",
     parameters = {"chave_artista"},
     elements =
     {
      cd =
       {
        order = {nome = "asc"},
         dest_page = "cd.html",
```

```
        selection =
         {
           link_name = {"artista_grava_cd"},
           condition = "(chave_artista = #chave_artista)"
         }
       }
     }
    }
  }
```

Grupos de contextos derivados de elo se comportam de forma semelhante a grupos de contextos derivados de classe, ou seja, a condição indica o atributo pelo qual os elementos são agrupados. Se o parâmetro possuir algum valor, o resultado será um contexto do grupo, senão o resultado será o grupo inteiro.

Acrescentando o grupo de contexto derivado de elo "cds do artista Bryan Adams construídas no ano X":

```
nav_contexts =
  {
   cds_alfa = {...},
   cds_91_92 = {...},
   cds_por_ano = {...},
   favoritos = {...},
   cds_artistas_60 = {...},
   cds_por_artista =       {...},
   cds_bryan =
     {
      context = "EG",
      nav_type = "I",
      parameters = {"ano"},
      elements =
      {
       cds =
         {
         dest_page = "cd.html",
          selection =
           {
             link_name = {"artista_grava_cd"},
             condition = "(ano_gravacao = #ano) AND "..
                         "(chave_artista='bryan')"
           }
         }
      }
     }
  }
```

O contexto resultante é a interseção dos elementos selecionados na condição sobre o atributo da classe *cd* (ano_gravacao = #ano) e dos selecionados na condição sobre o relacionamento artista_grava_cd (chave_artista = 'bryan'). Se o valor do parâmetro ano for nulo, todos os cds do artista Bryan Adams serão selecionados.

Acrescentando o contexto dinâmico por consulta "cds por consulta":

```
nav_contexts =
  {
   cds_alfa = {...},
   cds_91_92 = {...},
   cds_por_ano = {...},
   favoritos = {...},
```

```
  cds_artistas_60 = {...},
  cds_por_artista =       {...},
  cds_bryan = {...},
  cds_por_consulta =
   {
    context = "DC",    ----> Contexto Dinâmico por Consulta
    nav_type = "L",
    parameters = {"sql_statement"},
    elements =
    {
     cd =
      {
       dest_page = "cd.html",
       selection =
         {
          condition = "(#sql_statement)"
                           ----> O sql já está pronto
                           ----> no parâmetro sql_statement.
         }
       }
     }
    }
  }
```

No contexto dinâmico por consulta um comando sql é passado como parâmetro para a função que seleciona os elementos do contexto. Esse comando é executado para fazer a seleção. Esse contexto deve ser representado como uma seleção, com condição igual a "(#sql_statement)". O nome do parâmetro que contém o sql é pré-definido e deve ser sempre ***sql_statement***.

3.3.8 Tabela Lua nav_indexes

Todas as estruturas de acesso (índices) representados no esquema de contextos devem ser especificados na tabela *nav_indexes*, que é formada por várias tabelas identificadas pelo nome de cada estrutura de acesso.

Nome do campo	Significado
"indice"	Nome da estrutura de acesso (índice)[5].
type	Indica se a estrutura de acesso é estática ou dinâmica, simples ou hierárquica, múltipla ordenada ou não e as possíveis combinações. Os valores válidos para esse campo são pré-definidos e são mostrados mais adiante.
parameters	Lista com os parâmetros usados para montar o índice.
elements	Tabela que especifica como são selecionados os elementos que serão mostrados no índice. É composta por várias tabelas, uma para cada condição a ser respeitada na seleção dos elementos.
"elem"	Nome que identifica cada grupo de elementos selecionados a partir da mesma condição.
context	Nome do contexto ao qual pertencem os elementos. Se esse valor for nil, deve ser especificada uma classe em *class*.
class	Nome da classe à qual pertencem os elementos do índice. Deve ser especificada somente se *context* for nil.
condition	Condição à qual os elementos devem obedecer. Formato: (atributo operador valor) AND / OR (atrib op val) …
order	Lista com os nomes dos atributos pelos quais a estrutura de acesso será ordenada. Essa lista deve ter o formato: nome_atrib = tipo_ord, por exemplo, {nome_obra = "ASC", chave_obra = "DESC"}.
attributes	Lista com os nomes dos atributos que serão exibidos na estrutura de acesso, incluindo os seletores.
selectors	É uma tabela com os seletores do índice e seus destinos.
"atributo"	Nome de cada atributo que será um seletor no índice. Para cada atributo é especificado seu destino.
destination	Nome do índice ou contexto destino.
dest_page	Página html onde é exibido o índice destino ou os elementos do contexto destino
anchor	Existe caso o seletor contenha uma âncora estática. A âncora que aparecerá no índice será o conteúdo de âncora, não o atributo.
inst	Lista com os nomes dos atributos que identificam o elemento. Deve conter os nomes dos campos que compõem a chave da tabela do banco de dados, que representa a classe a qual pertence o atributo seletor.

. . .

. . .

. . .

[5] Índices e contextos devem ter nomes únicos, ou seja, não deve existir um índice com o mesmo nome de um contexto.

O atributo *type* indica se o índice é estático, dinâmico ou derivado da sessão (que é um tipo de índice dinâmico); simples ou hierárquico; com ordenação múltipla ou não. Os valores e significados desse atributo são os seguintes:

Valor	Significado
E	Estático
D	Dinâmico
T	Sessão (Temporário)
S	Simples
H	Hierárquico
M	Múltiplo Ordenado

O valor do atributo *type* será sempre uma combinação de 2 ou 3 letras. A primeira é E, D ou T. A segunda é S ou H. A terceira é M, que só existe se o contexto possuir ordenação múltipla. Portanto as combinações possíveis são: ES, ESM, DS, DSM, TS, TSM, EH , EHM, DH, DHM, TH, THM.

A seguir são mostrados alguns exemplos de estruturas de acesso.

A estrutura abaixo representa um índice totalmente estático. É o menu principal da aplicação, onde as âncoras e seus destinos são fixos.

```
nav_indexes =
 {
  menu_princ_idx =
   {
    type = "ES",
    elements =
     {
      cds_alfa =
       {
        attributes = {"sel1"},
        selectors =
         {
          sel1 =    ----> O nome do seletor não importa pois a âncora
                    ----> é fixa e está definida em anchor.
           {
            anchor = "CDs em Ordem Alfabética",
            destination = "cds_alfa_idx",
            dest_page = "cds_alfa.html"
           }
         }
       },
      cds_1997 =
        {
        attributes = {"sel1"},
        selectors =
          {
          sel1 =
            {
            anchor = "CDs gravados em 1997",
            destination = "cds_97_idx",
            dest_page = "cds_1997.html"
            }
          }
```

```
        },
       cds_artista =
        {
        attributes = {"sel1"},
        selectors =
          {
          sel1 =
            {
            anchor = "CDs por Artista",
            destination = "cds_por_artista_idx",
            dest_page = "cds_artista.html"
            }
          }
        },
       cds_ano =
        {
        attributes = {"sel1"},
        selectors =
         {
          sel1 =
           {
            anchor = "CDs por Ano",
            destination = "cds_por_ano_idx",
            dest_page = "cds_ano.html"
           }
         }
        }
      }
    }
 }
```

Neste exemplo, estão representados apenas três itens do menu principal. Cada item é especificado como um elemento. No item *cds_alfa*, por exemplo, a âncora é pré-definida em *anchor*, o destino é o índice *cds_alfa_idx*, e *cds_alfa.html* é o nome da página que contém a chamada da função que monta o índice *cds_alfa_idx*, ou seja, a função *index* do OOHDM-Web 2.0.

Os elementos do índice a seguir são exatamente os mesmos elementos do contexto *cds_alfa*. Não é definida nenhuma condição adicional.

```
nav_indexes =
  {
   menu_princ_idx = {...},
   cds_alfa_idx =
    {
     type = "ES",
     elements =
     {
      group =
       {
        context = "cds_alfa",
        attributes = {"nome","ano_gravacao"},
        selectors =
          {
           nome =
           {
            destination = "cds_alfa",
           dest_page = "cd.html",
           inst = {"chave_cd"}
```

```
          },
        ano_gravacao =
          {
            destination = "cds_por_ano",
           dest_page = "cd.html",
           inst = {"chave_cd"}
           }
         }
       }
     }
    }
  }
```

Note que o campo *order* não está especificado. Portanto a ordem dos elementos do índice será a ordem definida no contexto *obras_alfa*.

Nesse índice, cada item será identificado por dois seletores (*nome* e *ano_gravacao*). O destino dos seletores é o mesmo elemento, que será exibido utilizando o mesmo template (*cd.html*). Porém o contexto onde esse elemento será exibido será diferente, dependendo do seletor escolhido.

Para cada atributo seletor do índice é necessário especificar os campos que compõem a chave da tabela de classe que representa a classe do atributo seletor no banco de dados. Esses campos são informados no campo *inst* da tabela *selectors*.

O próximo exemplo mostra um índice de cds gravados em 1997:

```
nav_indexes =
  {
   menu_princ_idx = {...},
   cds_alfa_idx = {...},
   cds_97_idx =
    {
     type = "ESM",  ---------> Múltipla Ordenação
     parameters = {"ano"},
     elements =
     {
      group_cds =
       {
        class = "cd",
        condition = "(ano_gravacao='1997')",
        order = {nome = "asc", ano_gravacao = "desc"},
        attributes = {"nome", "CDs_1997"},
        selectors =
          {
           nome_cd =
             {
                   destination = "cds_por_ano, #ano",
              dest_page = "cd.html",
                   inst = {"chave_cd"}
             },
           CDs_1997 =
             {
              anchor = "CDs_1997",
              destination = "cds_por_ano",
              dest_page = "cd.html"
             }
          }
       }
     }
    }
```

```
  }
```

Neste exemplo os elementos não são selecionados a partir de um contexto. É dada a classe e uma condição, o que é suficiente para selecionar os elementos. O mesmo resultado seria obtido, usando o contexto *cds_por_ano* e a mesma condição. Essa é apenas uma forma alternativa de representar um índice.

Um dos seletores é o atributo *nome*, mas o outro não é um atributo da classe *cd*, mas uma âncora fixa, que será sempre a mesma ('CDs_1997') para todos os elementos do índice.

O parâmetro *ano* é passado para identificar um contexto no grupo de contextos *obras_por_ano*, que é o contexto destino do seletor *nome*. Mesmo que o ano seja sempre o mesmo, ele deve ser passado para o contexto (definido em *destination*) como um parâmetro.

Note que o parâmetro *ano* poderia ter sido usado na condição (ano_gravacao = #ano). Dessa forma esse índice poderia ser usado para exibir os CDs gravados em qualquer ano.

Esse índice possui ordenação múltipla, ou seja, deve ser passado um parâmetro chamado ***ordering*** para a função *index*, que indica qual dos atributos especificados em *order* vai ordenar os elementos.

O índice *cds_artistas_60_idx* é descrito a seguir:

```
nav_indexes =
  {
   menu_princ_idx = {...},
   cds_alfa_idx = {...},
   cds_97_idx = {...},
   cds_artistas_60_idx =
    {
     type = "ES",
     elements =
     {
      group =
        {
         context = "cds_artistas_60",
         attributes = {"nome", "nome_artista", "ano_gravacao"},
          selectors =
          {
            nome =
            {
                   destination = "cds_artistas_60",
             dest_page = "cd.html",
             inst = {"chave_cd"}
            },
            nome_artista =
             {
              destination = "artistas_alfa",
              dest_page = "artista.html",
              inst = {"chave_artista"}
             }
          }
        }
     }
    }
  }
```

O índice acima tem seus elementos selecionados a partir do contexto derivado de elo *cds_artistas_60*. Os elementos são da classe *cd*, mas atributos da classe *artista* também podem ser exibidos, já que foram obtidos através do elo.

Três atributos de cada elemento serão exibidos no índice, sendo que dois deles são seletores e outro é apenas informação. O seletor *nome_artista* tem como destino um elemento da classe *artista*, no contexto *artista_alfa*. O seletor *nome* tem como destino um elemento da classe *cd*, no contexto *cds_artistas_60*.

Abaixo é descrito o índice cds_bryan_idx:

```
nav_indexes =
  {
   menu_princ_idx = {...},
   cds_alfa_idx = {...},
   cds_97_idx = {...},
   cds_artistas_60_idx = {...},
   cds_bryan_idx =
    {
     type = "ES",
     parameters = {"ano"},
     elements =
     {
      group =
        {
         context = "cds_bryan",
         order = {nome = "asc"},
         attributes = { "nome", "ano_gravacao"},
         selectors =
           {
            nome =
             {
               destination = "cds_bryan, #ano",
               dest_page = "cd.html",
                    inst = {"chave_cd"}
             }
           }
        }
     }
    }
  }
```

Na estrutura mostrada acima, o contexto destino recebe o parâmetro *ano*. O nome do parâmetro passado para o índice deve ser o mesmo nome do parâmetro que á passado para o contexto, ou seja, *ano* deve estar presente na lista de parâmetros do contexto *cds_bryan*.

O tipo de navegação do contexto *cds_bryan* é por índice, portanto não foi definida uma ordem para a navegação pelos elementos desse contexto. No entanto, podemos definir uma ordem para esses elementos aparecerem no índice. Essa ordem será definida pelo valor de order, que é {nome = "asc"}. Se a ordem não for definida nem no contexto nem no índice, os elementos aparecerão em uma ordenação arbitrária.

A seguir é mostrado um índice de um contexto arbitrário:

```
nav_indexes =
  {
   menu_princ_idx = {...},
   cds_alfa_idx = {...},
   cds_97_idx = {...},
   cds_artistas_60_idx = {...},
   cds_bryan_idx = {...},
```

```
  favoritos_idx =
   {
    type = "ES",
    elements =
    {
     group =
       {
        context = "favoritos",
        attributes = {"nome", "nome_artista"},
        selectors =
          {
           nome =
             {
              destination = "favoritos",
              dest_page = "cd.html",
              inst = {"chave_cd"}
             },
           nome_artista =
             {
              destination = "favoritos",
              dest_page = "artista.html",
              inst = {"chave_artista"}
             }
          }
       }
    }
   }
}
```

Os elementos do contexto favoritos que formam esse índice pertencem a duas classes diferentes: cd e artista. Note que na lista de atributos há dois atributos, um de cada classe. Quando um elemento da classe *cd* for exibido, apenas o seletor *nome* será mostrado, pois *nome_artista* não pertence a classe *cd*. Já quando o elemento a ser exibido for da classe *artista*, apenas o seletor *nome_artista* será exibido.

O próximo índice descrito é cds_por_ano_idx.

```
nav_indexes =
  {
   menu_princ_idx = {...},
   cds_alfa_idx = {...},
   cds_97_idx = {...},
   cds_artistas_60_idx = {...},
   cds_bryan_idx = {...},
   favoritos_idx = {...},
   cds_por_ano_idx =
    {
     type = "ES",
     parameters = {"ano"},
     elements =
     {
      group =
        {
         context = "cds_por_ano",
         attributes = {"nome", "ano_gravacao"},
         selectors =
           {
            nome =
              {
               destination = "cds_por_ano, #ano",
```

```
            dest_page = "cd.html",
            inst = {"chave_cd"}
          }
        }
      }
    }
   }
  }
```

O índice *cds_por_ano_idx* tem seus elementos derivados de um grupo de contexto. O destino de cada elemento é um contexto específico do grupo, já que o ano (que é o atributo que define o grupo) é passado por parâmetro.

Abaixo temos um índice do grupo de contexto derivado de elo *cds_por_artista*, representado de forma simples:

```
nav_indexes =
  {
   menu_princ_idx = {...},
   cds_alfa_idx = {...},
   cds_97_idx = {...},
   cds_artistas_60_idx = {...},
   cds_bryan_idx = {...},
   favoritos_idx = {...},
   cds_por_ano_idx = {...},
   cds_por_artista_S_idx =
    {
     type = "ES",
     parameters = {"chave_artista"},
     elements =
     {
      group =
        {
         context = "cds_por_artista",
         order = {nome = "asc"},
         attributes = {"nome", "nome_artista","ano_gravacao"},
         selectors =
           {
           nome_artista =
            {
             destination = "artistas_alfa",
             dest_page = "artista.html",
             inst = {"chave_artista"}
            },
           nome =
            {
             destination = "cds_por_artista, #chave_artista",
             dest_page = "cd.html",
             inst = {"chave_cd"}
            }
           }
        }
     }
    }
  }
```

Esse índice não é hierárquico, portanto mostrará todos os atributos (*nome_artista*, *nome* e *ano_gravacao*) de todos os elementos do grupo na mesma página.

Agora podemos ver o mesmo índice do grupo de contextos *cds_por_artista*, representado de forma hierárquica:

```
nav_indexes =
  {
   menu_princ_idx = {...},
   cds_alfa_idx = {...},
   cds_97_idx = {...},
   cds_artistas_60_idx = {...},
   cds_bryan_idx = {...},
   favoritos_idx = {...},
   cds_por_ano_idx = {...},
   cds_por_artista_S_idx = {...},
   cds_por_artista_H_idx =
    {
     type = "EH",  --------------------> Índice Hierárquico
     parameters = {"chave_artista"},
     elements =
     {
      group =
        {
         context = "cds_por_artista",
         attributes = {{"nome_artista"},
                       {"nome", ano_gravacao"}},
         selectors =
         {
           nome_artista =
           {
             destination = "cds_por_artista_H_idx, #chave_artista",
             dest_page = "cds_artista_idx.html",
              inst = {"chave_artista"}
           },
           nome =
           {
             destination = "cds_por_artista, #chave_artista",
             dest_page = "cd.html",
              inst = {"chave_cd"}
           }
         }
        }
     }
    }
  }
```

Quando o índice é hierárquico, os atributos devem vir representados dentro de tabelas, uma para cada nível da hierarquia. Também deve existir um parâmetro para cada nível.

No primeiro nível o parâmetro *chave_artista* é nil, portanto todos os CDs de todos os artistas serão selecionados para serem exibidos no índice. O atributo a ser mostrado nesse primeiro nível é *nome_artista*, portanto o resultado é um índice dos artistas. Note que o destino do seletor *nome_artista* é um índice, passando *chave_artista* como argumento. O template *cds_artistas_idx.html* chama a função que cria esse índice. Portanto, ao selecionar um artista, o índice *cds_por_artista_H_idx* será criado novamente, dessa vez com a chave do artista selecionado. Isso fará com que seja criado um índice do segundo nível da hierarquia, onde serão exibidos apenas os CDs do artista cuja chave foi passada por parâmetro. Para cada elemento, esse índice vai exibir os atributos *nome* e *ano_gravacao*.

O próximo exemplo mostra um índice cujos elementos são o resultado de uma consulta:

```
nav_indexes =
  {
   menu_princ_idx = {...},
   cds_alfa_idx = {...},
   cds_97_idx = {...},
   cds_artistas_60_idx = {...},
   cds_bryan_idx = {...},
   favoritos_idx = {...},
   cds_por_ano_idx = {...},
   cds_por_artista_S_idx = {...},
   cds_por_artista_H_idx = {...},
   cds_por_consulta_idx =
    {
     type = "DS",
     parameters = {"sql_statement"},
     elements =
     {
      group =
        {
         context = "cds_por_consulta",
         attributes = {"nome"},
         selectors =
          {
           nome =
            {
             destination = "cds_por_consulta, #sql_statement",
             dest_page = "cd.html",
             inst = {"chave_cd"}

            }
          }
        }
     }
    }
 }
```

No índice de um contexto por consulta o parâmetro *sql_statement* deve sempre ser passado. Esse parâmetro é passado para o contexto por consulta definido em *context*, para que seja feita a seleção dos elementos.

3.3.9 Tabela Lua nav_landmarks

Landmarks são elementos que podem ser acessados a partir de qualquer local da aplicação. Para representá-los, foi definida a tabela *nav_landmarks*. Nesta tabela são especificados os *landmarks* representados no esquema de contextos de navegação.

Nome do campo	Significado
"nome_landmark"	Nome que identifica o *landmark*.
anchor	Âncora que representa o *landmark*.
destination	Índice de um contexto ou instância de um objeto de um contexto referenciada pelo *landmark*.
...	

Por exemplo, alguns itens do menu principal poderiam ser especificados como landmarks:

```
nav_landmarks =
  {
    nome =
     {
      anchor = "CDs em Ordem Alfabética",
      destination = "cds_alfa_idx"
     },
    ano =
     {
      anchor = "CDs por Ano",
      destination = "cds_por_ano_idx"
     },
    artista =
     {
      anchor = "CDs por Artista",
      destination = "cds_por_artista_idx"
     }
  }
```

3.4 Biblioteca de Funções do OOHDM-Web

Nesta seção descrevemos as principais funções da biblioteca do ambiente OOHDM-Web e as principais alterações implementadas na nova versão 2.0. A biblioteca de funções do OOHDM-Web está organizada da seguinte forma.

- **Criação da Base de Dados**

A criação da base de dados é o passo intermediário entre a criação do projeto OOHDM e a implementação da aplicação. A função ***Create_Tab_DB*** lê o conteúdo do projeto de navegação e cria no banco de dados todas as tabelas de classe, as tabelas de relacionamento e as tabelas de classe em contexto da aplicação. Ela é chamada na interface OOHDM-Project que dispara o processo de criação da base de dados.

A versão anterior do ambiente OOHDM-Web cria as tabelas no banco de dados considerando que a chave primária é constituída apenas por um identificador único chamado "prim_key". Assim, o atributo da classe conceitual que será um campo chave precisa ser mapeado na tabela Lua *map_classes* do projeto de navegação, para ser criado na tabela de classe do banco de dados com o nome fixo "prim_key".

Na nova versão OOHDM-Web 2.0, é possível especificar que a chave primária de uma tabela será composta por mais de um atributo. Esta especificação é feita na tabela Lua *con_classes* especificando o valor "yes" para o campo *prim_key* na definição dos atributos de uma classe. Os atributos definidos como chave não precisam ser mapeados na tabela Lua *map_classes*, a menos que o projetista queira criá-los com um nome diferente na tabela de classe do banco de dados.

- **Criação de Índices**

Os tipos de página mais comuns em aplicações OOHDM são as páginas que exibem índices e as páginas que exibem os elementos dos contextos. Para a criação dos índices o ambiente OOHDM-Web possui as seguintes funções:

Index	Retorna uma tabela com os elementos do índice. Essa tabela deve ser passada para uma função de exibição, como Horizontal, por exemplo.

Make_Ind_Links	Retorna uma tabela com os elementos do índice.
Vertical	Organiza os elementos de um índice um embaixo do outro. Os atributos de cada elemento são exibidos um ao lado do outro.
Horizontal	Organiza os elementos de um índice um ao lado do outro.

Na versão 2.0, a função ***Make_Ind_Links*** passou a consultar o campo *inst* da tabela Lua nav_indexes para criar variáveis cgi que contém a chave da instância corrente (o seletor selecionado no índice). Estas variáveis são passadas para a página de apresentação do nó para permitir a navegação entre os elementos do contexto, que é sempre realizada com base na instância corrente do nó.

- **Seleção de Elementos de Contextos**

A função ***Get_Ctx_Elements*** seleciona todos os elementos pertencentes a um determinado contexto. Ela é usada por outras funções do OOHDM-Web, como *Index*, mas também pode ser usada separadamente.

A função *Get_Ctx_Elements* é usada pela função *Index* para retornar todos os elementos de um determinado contexto que compõe um índice. Ela retorna também o total de elementos do contexto e uma tabela com o nome das classes utilizadas para selecionar os elementos (classes às quais pertencem os elementos, e classes relacionadas, no caso de contextos derivados de elo). Essa função também está disponível para o desenvolvedor, caso ele queira apenas obter os elementos do contexto sem montar o índice, ou seja necessário realizar algum tipo de tratamento na tabela de elementos (retornada por *Get_Ctx_Elements*), antes de montar o índice.

- **Exibição de Elementos**

A função ***Attrib*** é usada para montar as páginas que exibem informações sobre os elementos dos contextos. Ela retorna o valor de um atributo de uma instância de uma classe navegacional. Essa função busca o valor do atributo na tabela de classe do banco de dados que contém os elementos de uma determinada classe. Seus parâmetros principais são:

ctx	Contexto onde será apresentado o atributo.
class	Classe que contém o atributo.
inst_key	Chave da instância da qual será mostrado o atributo.
attrib	Nome do atributo que será mostrado. O atributo pode ser da classe passada no parâmetro 'class' ou ser um atributo da classe em contexto da classe 'class' no contexto 'ctx'.

Na versão 2.0 do ambiente OOHDM-Web, o parâmetro *inst_key* passou a ser do tipo tabela, onde devem ser informados os nomes e os valores de instância de cada atributo que compõe a chave. Na versão anterior isto não é necessário pois a chave é o campo fixo "prim_key". Assim, basta informar o valor para este campo, como mostra o exemplo abaixo.

Exemplo:

Versão anterior:

```
Attrib {ctx = "cds_alfa", class = "cd", inst_key = "cd0001",
        attrib = "nome"}
```

Versão 2.0:

Attrib {ctx = "cds_alfa", class = "cd",
inst_key = {*chave_cd*="cd0001", *codigo*="REF001"}, `attrib = "nome"}`

- **Navegação entre os Elementos de um Contexto**

O OOHDM define vários tipos de navegação entre os elementos dos contextos. Isso significa que é possível, passar de um elemento a outro, do mesmo ou de outro contexto, sem ter que retornar ao índice.

O OOHDM-Web fornece quatro funções que implementam essa navegação:

Link_Next	Gera um link para o próximo elemento do contexto.
Link_Prev	Gera um link para o elemento anterior do contexto.
Link_First	Gera um link para o primeiro elemento do contexto.
Link_Last	Gera um link para o último elemento do contexto.

Essas funções geram automaticamente links para outros elementos de um determinado contexto, desde que o tipo de navegação definido para o contexto permita esse tipo de acesso. Devem ser utilizadas nas páginas que exibem as informações de um nó.Todas as funções possuem os seguintes parâmetros:

anc_html	Âncora fixa que representa o link para o elemento. É usado quando a âncora tem sempre o mesmo valor. Obrigatório caso anc_attrib seja nil.
anc_attrib	Usado quando a âncora é um atributo da classe. Seu valor variará de acordo com a instância sendo mostrada. Obrigatório caso anc_html seja nil.
ctx	O elemento anterior (próximo, primeiro ou último) é calculado de acordo com os elementos do contexto atual (que foi usado para montar a página da instância atual). Caso ele seja calculado dentro de algum outro contexto, deve-se usar o parâmetro ctx. Não é obrigatório.
inst_key	O elemento anterior (próximo, primeiro ou último) é calculado de acordo com a instância atual (que está sendo exibida). Se o objetivo for descobrir o elemento anterior a outro que não seja o elemento atual, o parâmetro inst_key deve conter o valor da chave do elemento do qual se quer descobrir o anterior. Não é obrigatório.
attrib_key	Nome(s) do campo(s) que compõem a chave primária da tabela do banco de dados que contém os elementos do contexto.

Na versão 2.0 do ambiente OOHDM-Web, o parâmetro *inst_key* passou a ser do tipo tabela, onde deverão ser informados os nomes e os valores de instância de cada atributo que compõe a chave. Um novo parâmetro chamado ***attrib_key*** foi criado para a especificação dos campos que compõem a chave na tabela do banco de dados que contém os elementos do contexto. Esse campo é obrigatório.

Exemplo:

```
Link_Prev {anc_html = '<img src = "'..cgilua.script_vdir..
          'imagens/ant.gif" border=0 alt="Anterior">',
          attrib_key = {'chave_cd'}}
```

- **Contextos Dinâmicos**

Um contexto dinâmico é aquele em que se pode acrescentar e retirar elementos durante a navegação. Existem dois tipos de contextos dinâmicos: os derivados (de classe ou de elo), que são contextos especificados por propriedades ou relações de uma classe, e os explícitos (arbitrários), onde os elementos são inseridos ou retirados explicitamente. As funções fornecidas pelo OOHDM-Web para implementar esses contextos são descritas a seguir, de acordo com o tipo de contexto dinâmico.

Contextos derivados:

Para inserir ou retirar um objeto de um contexto derivado de classe basta alterar a propriedade que define o contexto. Quando um novo objeto é criado em uma classe, ou quando um objeto existente é destruído, ou ainda quando as características de um objeto são modificadas, o contexto derivado dessa classe será automaticamente modificado. As principais funções para criação, exclusão e alteração de objetos de uma classe são:

Create_Object	Cria uma nova instância em uma tabela de classe.
Update_Object	Modifica o valor de um ou mais atributos de um objeto.
Delete_Object	Retira uma instância de uma tabela de classe.
Insert_Cl_Ctx	Cria uma instância de uma classe em contexto de um determinado objeto, na tabela de classes em contexto do banco de dados.
Update_Cl_Ctx	Atualiza os atributos de uma instância de uma classe em contexto de um determinado objeto.
Delete_Cl_Ctx	Apaga uma instância de uma classe em contexto de um determinado objeto.

Para modificar um contexto derivado de elo basta criar ou eliminar um elemento da relação que define o contexto. O contexto passará a refletir esse novo elemento. As funções que permitem a criação ou exclusão de elos são as seguintes:

Create_Relationship	Cria um elo entre dois objetos.
Delete_Relationship	Remove um elo existente entre dois objetos.

Contextos explícitos:

Quando o contexto é arbitrário não existe uma propriedade que o define. Portanto é necessário inserir e retirar elementos do próprio contexto, ou seja, da enumeração. As funções que fazem isso são as seguintes:

Insert_In_Ctx	Representa a ação de inserir um objeto em um contexto dinâmico arbitrário.
Delete_From_Ctx	Retira um elemento do contexto dinâmico arbitrário.

Além das funções descritas acima, existem funções específicas para contextos dinâmicos baseados em sessão (aqueles que só existem durante uma sessão de navegação). São elas:

Insert_In_Ctx_Session	Representa a ação de inserir um objeto em um contexto arbitrário baseado em sessão.
Delete_From_Ctx_Session	Retira um elemento de um contexto arbitrário baseado em sessão.
Session_To_DB	Transforma os dados da sessão em dados permanentes.
Create_Object_Session	Cria um objeto temporário de uma classe.
Update_Object_Session	Modifica o valor de um ou mais atributos de um objeto temporário.
Delete_Object_Session	Retira um objeto temporário de uma tabela de sessão.
Insert_Cl_Ctx_Session	Cria uma classe em contexto temporária para um determinado objeto, na tabela de classes em contexto de sessão.
Update_Cl_Ctx_Session	Atualiza os atributos da classe em contexto.
Delete_Cl_Ctx_Session	Apaga uma classe em contexto temporária.
Create_Relationship_Session	Cria um relacionamento temporário entre dois objetos, na tabela de relacionamento de sessão.
Delete_Relationship_Session	Remove um relacionamento da tabela de relacionamento de sessão.

Todas essas funções foram alteradas para conter o tratamento para chaves compostas e consultar a nova estrutura das tabelas Lua definidas para a versão OOHDM-Web 2.0.

3.5 O OOHDM-Translation

Como mencionado anteriormente, a descrição do projeto OOHDM de uma aplicação hipermídia em estruturas de dados da linguagem Lua não é amigável ao ser humano, devido à sintaxe que deve ser seguida para definir as diversas tabelas Lua que compõem o projeto OOHDM-Web da aplicação. A geração manual dessa descrição é trabalhosa e, muitas vezes, torna-se a principal fonte de erros durante a execução da aplicação na Web.

O OOHDM-Translation foi criado com o objetivo de gerar, de forma automática, a descrição do projeto OOHDM-Web de uma aplicação hipermídia a partir de sua especificação OOHDM-ML.

O OOHDM-Translation é uma folha de estilo composta por templates implementados em XSLT. Estes templates contêm as regras de transformação, que serão aplicadas à árvore de elementos de um documento OOHDM-ML, para gerar a descrição do projeto OOHDM-Web de uma aplicação hipermídia.

Com esta folha de estilo, o desenvolvedor da aplicação poderá gerar automaticamente a descrição de todas as tabelas Lua necessárias à biblioteca de funções do ambiente OOHDM-Web 2.0, informando apenas o nome do documento XML que contém a especificação declarativa do projeto OOHDM da aplicação.

O processo de transformação que o OOHDM-Translation implementa conta com o auxílio de um processador XSLT para aplicar as regras de transformação da folha de estilo à árvore de elementos do documento OOHDM-ML, informado pelo projetista, e produzir a árvore resultado que representa o projeto OOHDM-Web da aplicação. O processador XSLT utilizado neste trabalho é o *Saxon* [Kay 2000]. A figura 3.6 ilustra o processo de transformação do OOHDM-Translation.

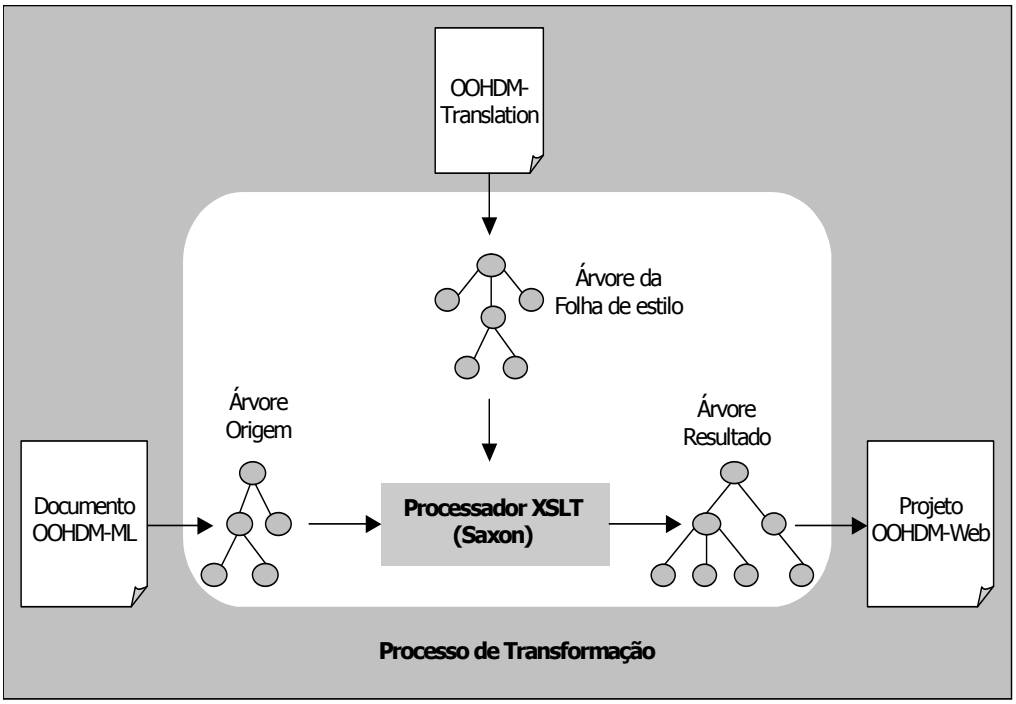


Figura 3.6 – Processo de Transformação do OOHDM-Translation

A folha de estilo OOHDM-Translation é composta por outras folhas de estilo. Cada folha de estilo contém um conjunto de regras de transformação que implementa a definição e a sintaxe que deve ser seguida para descrever cada tabela Lua que compõe o projeto OOHDM-Web da aplicação.

O conjunto de regras de transformação, dentro de cada folha de estilo, também é dividido em templates para tratar um ou mais elementos do documento XML e possibilitar o reuso de código entre as diversas folhas de estilo. As folhas de estilo que compõem o OOHDM-Translation são mostradas na figura abaixo.

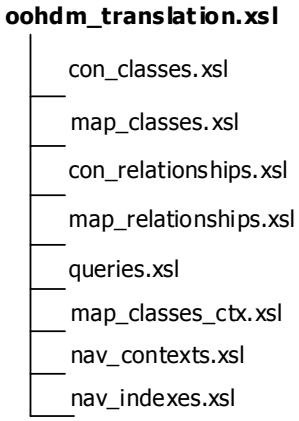


Figura 3.7 – Folhas de estilo que compõem o OOHDM-Translation

A tabela abaixo apresenta a especificação da folha de estilo oohdm_translation.xsl e seu código XSLT correspondente. Esta folha de estilo cria a estrutura básica de cada tabela Lua que compõem o projeto OOHDM-Web. As especificações das demais folhas de estilo, responsáveis pela geração do conteúdo de cada tabela Lua criada, podem ser consultadas no apêndice III. O código fonte de cada folha de estilo pode ser consultado no apêndice IV.

A utilização da folha de estilo OOHDM-Translation e a funcionalidade implementada nas regras de template que a compõem serão abordadas com mais detalhes na seção 4.2 do capítulo 4.

Especificação (pseudo-código)	**Código XSLT**
Para o elemento *conceptual_model* do documento XML associado à folha de estilo faça	<xsl:template match="conceptual_model">
Crie a estrutura Lua con_classes incluindo o formato abaixo: con_classes = {	con_classes = {
Processe recursivamente todos os elementos filhos c*onceptual_class*.	<xsl:apply-templates select="conceptual_class"/>
Complete a estrutura con_classes utilizando o símbolo "}" , alinhando –o com o símbolo "{" inicial.	}
Crie a estrutura Lua map_classes incluindo o formato abaixo: map_classes = {	map_classes = {
Processe recursivamente todos os elementos filhos *conceptual_class* e todos os elementos *map_conceptual_class* relacionados a eles.	<xsl:apply-templates select="conceptual_class" mode="map_classes"/>
Complete a estrutura map_classes utilizando o símbolo "}" , alinhando –o com o símbolo "{" inicial.	}
Se o elemento conceptual_model possui algum elemento filho *relationship* então	<xsl:if test="relationship">
Crie a estrutura Lua con_relationships incluindo o formato abaixo: con_relationships = {	con_relationships = {
Processe recursivamente todos os elementos filhos *relationship*.	<xsl:apply-templates select="relationship"/>
Complete a estrutura con_relationships utilizando o símbolo "}" , alinhando-o com o símbolo "{" inicial.	}
Crie a estrutura Lua map_relationships incluindo o formato abaixo: map_relationships = {	map_relationships = {
Processe todos os elementos *relationship* e todos os elementos *map_relationship* relacionados a eles.	<xsl:apply-templates select="relationship" mode="map_relationships"/>
Complete a estrutura map_relationships utilizando o símbolo "}".	}
fim-Se.	</xsl:if>
Fim-Para.	</xsl:template>

Para o elemento *navigational_model* do documento XML associado à folha de estilo faça	<xsl:template match="navigational_model">
Crie a estrutura Lua queries incluindo o formato abaixo: queries = {	queries = {
Processe recursivamente todos os elementos filhos navigation*al_class*.	<xsl:apply-templates select="navigational_class"/>
Complete a estrutura queries utilizando o símbolo "}" , alinhando –o com o símbolo "{" inicial.	}
Se o elemento navigational_model possui algum elemento filho *context_class* então Crie a estrutura Lua map_classes_ctx incluindo o formato abaixo: map_classes_ctx = {	<xsl:if test="context_class"> map_classes_ctx = {
Processe recursivamente todos os elementos filhos *context_class* e todos elementos *map_context_class* relacionados a eles.	<xsl:apply-templates select="context_class"/>
Complete a estrutura map_classes_ctx utilizando o símbolo "}" , alinhando-o com o símbolo "{" inicial.	}
fim-Se.	</xsl:if>
Crie a estrutura Lua nav_contexts incluindo o formato abaixo: nav_contexts = {	nav_contexts = {
Processe recursivamente todos os elementos *navigational_context*.	<xsl:apply-templates select="navigational_context"/>
Complete a estrutura nav_contexts utilizando o símbolo "}", alinhando –o com o símbolo "{" inicial.	}
Crie a estrutura Lua nav_indexes incluindo o formato abaixo: nav_indexes = {	nav_indexes = {
Processe recursivamente todos os elementos *index ou hierarch_index*.	<xsl:apply-templates select="index|hierarch_index"/>
Complete a estrutura nav_indexes utilizando o símbolo "}", alinhando –o com o símbolo "{" inicial.	}
Fim-Para.	</xsl:template>
Inclua a folha de estilo con_classes.xsl	<xsl:include href="con_classes.xsl"/>
Inclua a folha de estilo map_classes.xsl	<xsl:include href="map_classes.xsl"/>
Inclua a folha de estilo con_relationships.xsl	<xsl:include href="con_relationships.xsl"/>
Inclua a folha de estilo map_relationships.xsl	<xsl:include href="map_relationships.xsl"/>
Inclua a folha de estilo queries.xsl	<xsl:include href="queries.xsl"/>
Inclua a folha de estilo map_classes_ctx.xsl	<xsl:include href="map_classes_ctx.xsl"/>
Inclua a folha de estilo nav_contexts.xsl	<xsl:include href="nav_contexts.xsl"/>
Inclua a folha de estilo nav_indexes.xsl	<xsl:include href="nav_indexes.xsl"/>

Tabela 3.1 – Especificação e código XSLT da folha de estilo OOHDM-Translation

4 Exemplos

Este capítulo apresenta dois exemplos de utilização da linguagem OOHDM-ML para especificar de forma declarativa o projeto OOHDM de aplicações hipermídia. O primeiro exemplo é a especificação de uma aplicação hipermídia sobre arquitetura, cujo principal enfoque é a utilização da gramática definida na DTD OOHDM-ML para criar um documento OOHDM-ML válido a partir das definições do projeto OOHDM da aplicação. O segundo exemplo é a especificação de uma aplicação hipermídia para uma Coleção de CDs, utilizando uma versão simplificada dos modelos apresentados no capítulo 2 para o domínio de vendas de CDs. O enfoque principal deste segundo exemplo é a utilização da folha de estilo OOHDM-Translation para gerar o projeto OOHDM-Web da aplicação, que será usado pelas funções da biblioteca do ambiente OOHDM-Web 2.0.

4.1 Aplicação "Arquitetura Moderna"

Para exemplificar a utilização da linguagem OOHDM-ML para especificar o projeto OOHDM de uma aplicação hipermídia, utilizamos a aplicação "Arquitetura Moderna". Nesta seção, apresentamos o modelo conceitual e o projeto de navegação da aplicação e alguns exemplos da utilização da DTD OOHDM-ML para especificá-los. A especificação completa da aplicação poderá ser consultada no apêndice V.

4.1.1 Modelo Conceitual

O modelo conceitual da aplicação "Arquitetura Moderna" é formado pelas classes *obras*, *arquitetos* e *construtoras*, e pelos relacionamentos entre os objetos dessas classes, como ilustra a figura 4.1 abaixo:

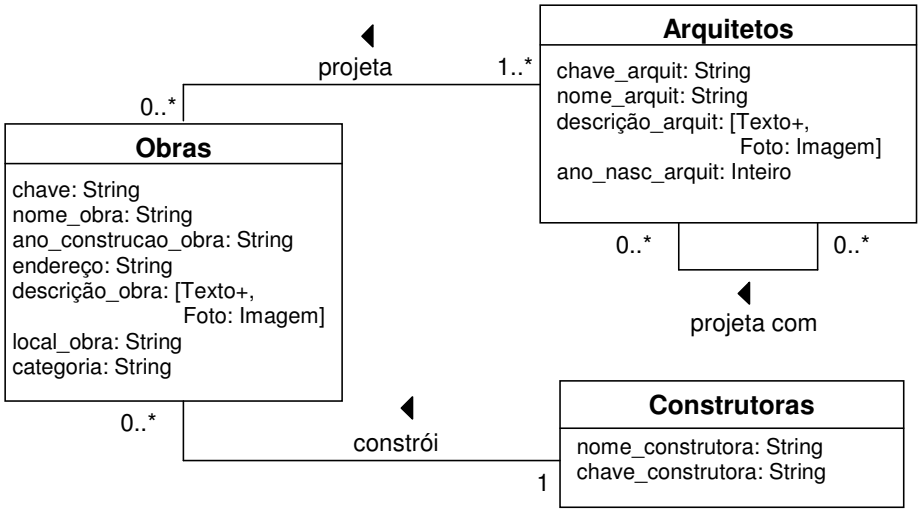


Figura 4.1 – Modelo Conceitual da aplicação "Arquitetura Moderna"

A especificação declarativa do modelo conceitual de uma aplicação hipermídia é feita utilizando-se a estrutura do elemento ***conceptual_model*** da linguagem OOHDM-ML.

Cada classe do modelo conceitual é representada por um elemento ***conceptual_class*** e deve possuir obrigatoriamente um nome e um identificador, que deve ser único em todo o documento OOHDM-ML. Por exemplo, na especificação da aplicação "Arquitetura Moderna" a classe *Obras* possui o identificador "*id_obras*" e o nome "*obras*" informados nos atributos *id* e *name* do elemento ***conceptual_class***, como pode ser observado no trecho de especificação abaixo.

```
<conceptual_class id="id_obras" name="obras">
    <attrib name="chave"    type="string" size="20"/>
    <attrib name="nome_obra" type="string" size="70"/>
    <attrib name="ano_construcao_obra" type="string" size="4"/>
    <attrib name="endereco" type="string"  size="50"/>
    <attrib name="descricao_obra" type="text">
        <perspective name="descricao"  type="text" default="yes"/>
        <perspective name="foto_obra" type="image" size="30"/>
    </attrib>
    <attrib name="local_obra" type="string" size="50"/>
    <attrib name="categoria" type="string" size="50"/>
</conceptual_class>
```

As propriedades[6] de uma classe conceitual são especificadas com o elemento ***attrib***, cujos atributos descrevem suas principais características como nome, tipo e tamanho. No trecho de especificação acima, a propriedade "ano de construção da obra" recebe o nome *ano_construcao_obra*, o tipo *string* e o tamanho *4* especificados, respectivamente, nos atributos *name*, *type* e *size* do elemento ***attrib***. As perspectivas de uma propriedade são especificadas pelo elemento ***perspective***, que possui um atributo chamado *default* para indicar se a perspectiva em questão é uma perspectiva default ou não. Nesta especificação, a propriedade *descrição_obra* possui duas perspectivas: *descricao* do tipo text, que é a perspectiva default, e *foto_obra* do tipo image.

Os relacionamentos entre os objetos das classes conceituais são especificados utilizando-se o elemento ***relationship***. A definição de cada relacionamento é especificada através dos atributos deste elemento. O trecho de especificação abaixo mostra a especificação do relacionamento *arquiteto projeta obra* do modelo conceitual ilustrado na figura 4.1.

```
<relationship  id="relacao_projeta" name="arquit_projeta_obra"
               source_class="id_arquitetos" target_class="id_obras"
               source_cardinality="n" target_cardinality="n">
</relationship>
```

Como na especificação das classes, cada relacionamento também deve possuir um nome e um identificador único em todo o documento OOHDM-ML. Os atributos *source_class* e *target_class* especificam, respectivamente, a classe origem e a classe destino do relacionamento. O valor desses atributos é o identificador de uma classe conceitual, especificado no atributo id do elemento *conceptual_class*. A cardinalidade do papel de cada classe conceitual que participa do relacionamento é especificada pelos atributos *source_cardinality* e *target_cardinality*.

[6] O termo propriedade está sendo usado no lugar de atributo para distinguir atributos de classe (propriedades) dos atributos dos elementos da linguagem OOHDM-ML.

No exemplo acima, as classes que participam do relacionamento "*arquiteto projeta obra*" são "*arquitetos*" e "*obras*", identificadas, respectivamente, pelos valores "*id_arquitetos*" e "*id_obras*" dos atributos *source_class* e *target_class*. A cardinalidade do papel de cada classe nesse relacionamento é "n", como especificado nos atributos *source_cardinality* e *target_cardinality*.

4.1.2 Modelo de Navegação

O modelo de navegação do site "Arquitetura Moderna" é formado pelas classes navegacionais *obras* e *arquitetos*, e pelos elos entre os objetos dessas classes, como mostra a figura 4.2 abaixo:

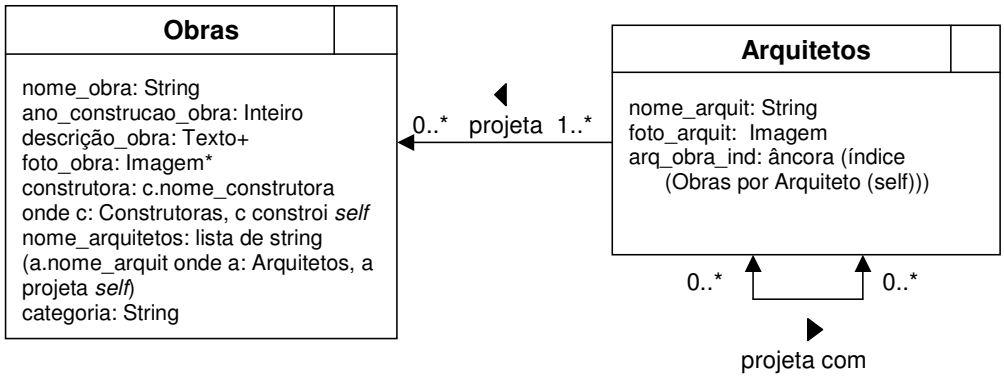


Figura 4.2 – Modelo de Navegação da aplicação "Arquitetura Moderna"

A especificação declarativa do modelo de navegação de uma aplicação hipermídia é feita utilizando-se a estrutura do elemento ***navigational_model*** da linguagem OOHDM-ML.

Cada classe navegacional (nó) do modelo de navegação é representada por um elemento ***navigational_class*** e deve possuir obrigatoriamente um nome e um identificador, que deve ser único em todo o documento OOHDM-ML. Por exemplo, na especificação do site "Arquitetura Moderna", o nó *Obras* possui o identificador "*no_obras*" e o nome "*obras*" informados nos atributos *id* e *name* do elemento ***navigational_class***, como pode ser observado no trecho de especificação abaixo.

```
<navigational_class id="no_obras" name="obras">
    <nav_attrib name="chave" conceptual_class="id_obras"/>
    <nav_attrib name="nome_obra" conceptual_class="id_obras"/>
    <nav_attrib name="ano_construcao_obra" conceptual_class="id_obras"/>
    <nav_attrib name="nome_construtora" conceptual_class="id_construtoras"/>
    <nav_attrib name="nome_arquit"  conceptual_class="id_arquitetos"/>
    <nav_attrib name="categoria" conceptual_class="id_obras"/>
    <perspective_attrib name="foto_obra" perspective="Foto"
                        conceptual_attrib="descricao_obra"
                        conceptual_class="id_obras" optional="yes"/>
    <perspective_attrib name="descricao_obra" perspective="Texto"
                        conceptual_attrib="descricao_obra"
                        conceptual_class="id_obras"/>
</navigational_class>
```

As propriedades de um nó, cujas instâncias serão apresentadas durante a navegação, devem ser propriedades definidas em uma classe conceitual. Essas propriedades são especificadas pelo elemento ***nav_attrib***. Este elemento possui os atributos *name* e *conceptual_class* que especificam o nome da propriedade no nó e a que classe conceitual ela pertence. O nome da propriedade no nó deve ser o mesmo nome definido para a propriedade na classe conceitual. O valor do atributo *conceptual_class* deve ser um identificador de uma classe conceitual especificado pelo atributo id de um elemento *conceptual_class*. Como podemos observar no trecho de especificação acima, um nó pode apresentar propriedades de outras classes conceituais além das propriedades da classe base. Por exemplo, as propriedades *nome_construtora* e *nome_arquit* pertencem respectivamente às classes conceituais *construtoras* e *arquitetos,* identificadas pelos valores "*id_construtoras*" e "*id_arquitetos*" do atributo *conceptual_class*.

Uma propriedade com múltiplas perspectivas de uma classe conceitual, que possui uma perspectiva default, deve ser mapeada no modelo de navegação para várias propriedades de um nó, uma para cada perspectiva. A especificação dessas propriedades no nó é feita utilizando-se o elemento ***perspective_attrib***. Seu atributo *name* especifica o nome da propriedade criada no nó. Os atributos *perspective*, *conceptual_attrib* e *conceptual_class* descrevem, respectivamente, o nome da perspectiva que deu origem a propriedade criada, a qual propriedade conceitual esta perspectiva pertence, e em que classe conceitual ela está definida. Por exemplo, as perspectivas *descrição* e *foto_obra*, especificadas para a propriedade "descrição_obra" da classe conceitual *obras*, foram mapeadas, usando o elemento perspective_attrib, para duas propriedades no nó *obras*: *foto_obra* e *descrição_obra*. A propriedade foto_obra tornou-se opcional, conforme especificado pelo atributo *optional*.

Os elos existentes entre os objetos das classes navegacionais são especificados utilizando-se o elemento ***link***. Como na especificação dos relacionamentos conceituais, as informações sobre um elo são especificadas através dos atributos do elemento ***link***. No exemplo abaixo, os nós que participam do elo "*arquiteto projeta obra*" são "*arquitetos*" e "*obras*", identificados, respectivamente, pelos valores "*no_arquitetos*" e "*no_obras*" dos atributos *source_class* e *target_class*. A cardinalidade do papel de cada nó neste relacionamento é "n", conforme especificado pelos atributos *source_cardinality* e *target_cardinality*.

```
<link id="link_projeta" name="arquit_projeta_obra"
      source_class="no_arquitetos" target_class="no_obras"
      source_cardinality="n" target_cardinality="n">
</link>
```

4.1.3 Esquema de Contextos de Navegação

O esquema de contextos ilustrado na figura 4.3 apresenta os contextos, as estruturas de acesso e os landmarks que representam as possíveis navegações da aplicação "Arquitetura Moderna". Os contextos de navegação são representados pelos retângulos brancos dentro dos retângulos sombreados que representam os nós. As estruturas de acesso são representadas pelos retângulos com linhas tracejadas e os landmarks são representados por setas com um círculo em uma das extremidades.

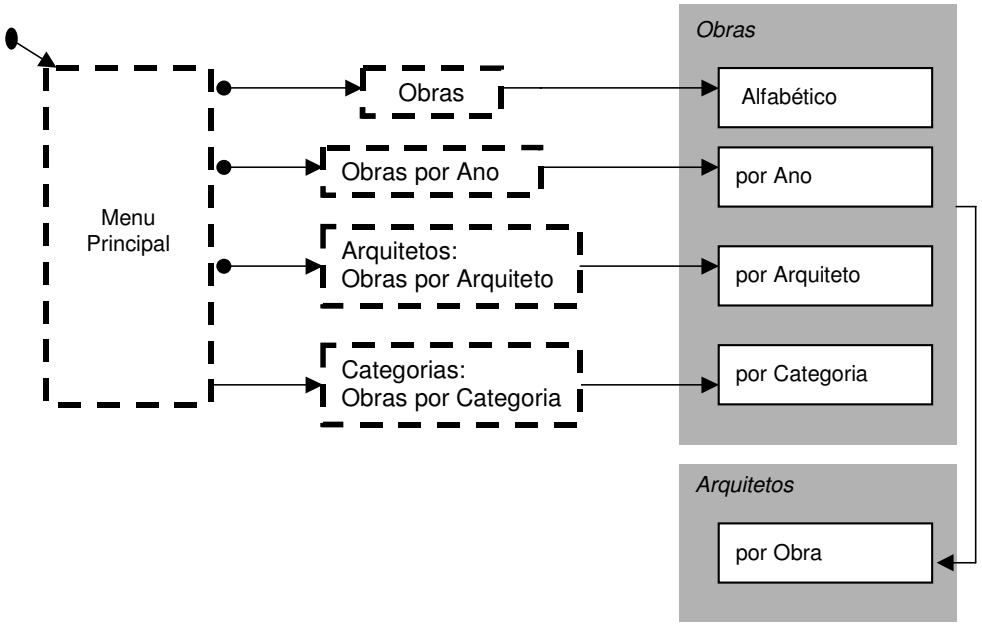


Figura 4.3 – Esquema de Contextos de Navegação do site "Arquitetura Moderna"

4.1.3.1 Especificação dos Contextos de Navegação

Cada contexto representado no esquema de contextos de navegação é especificado utilizando-se o elemento ***navigational_context***.

A especificação OOHDM-ML de um contexto é feita com base no cartão de especificação do contexto. Por exemplo, o grupo de contextos derivado de classe "*Obras por Ano*" é especificado com base no cartão de especificação abaixo.

Contexto: **Obras por Ano**
Tipo**: Estático**
Parâmetro**: ano**
Elementos**: o: Obras ONDE o.ano_construção_obra = <ano>**
Classe em Contexto**:**
Ordenação**: por ano_contrução_obra, ascendente**
Navegação Interna**: seqüencial**
Operações**:**
Usuário**: Visitante** Permissão**: leitura**
Comentários**:**

Este cartão contém as informações que definem o contexto e formam o conteúdo do elemento *navigational_context*, como pode ser observado no trecho de especificação.

```
<navigational_context id="contexto_obras_por_ano" name="obras_por_ano"
                      element_class="no_obras" type="static"
                      navigation_type="sequential">
    <selection>
```

```
            <equal>
                <attribute name="ano_construcao_obra" navigational_class="no_obras"/>
                <nav_parameter>ano</nav_parameter>
            </equal>
        </selection>
        <order  id="ordenacao_por_ano" default="yes">
            <order_element name="ano_construcao_obra" navigational_class="no_obras"
                           criteria="asc"/>
        </order>
        <restriction_use permission=”leitura”>
            <user>Visitante</user>
        </restriction_use>
    </navigational_context>
```

Como os nós e os elos do modelo de navegação, todos os contextos devem possuir um identificador único e um nome, especificados nos atributos *id* e *name* do elemento *navigational_context*. O atributo *element_class* especifica o nó ou os nós de navegação, cujos elementos formam o contexto. Este atributo deve receber como valor o identificador de uma ou mais classes navegacionais especificado no atributo id de um ou mais elementos *navigational_class*. No trecho de especificação acima, o contexto "*Obras por Ano*" é formado pelos objetos do nó "*obras*" identificado pelo valor *no_obras* do atributo *element_class*. O tipo do contexto e a navegação interna entre seus elementos são especificados pelos atributos *type* e *navigation_type*. Nesse exemplo, o contexto é estático e a navegação interna entre seus elementos é seqüencial.

A condição para a seleção dos elementos de um contexto, definida no campo "*Elementos*" do cartão de especificação, é especificada pelo elemento ***selection***. A estrutura desse elemento é formada por outros elementos que especificam as expressões lógicas, relacionais e aritméticas que compõem a condição de seleção. Por exemplo, na especificação do contexto "*Obras por Ano*", o elemento ***equal*** especifica a expressão relacional "*ano_construção_obra = ano*", onde *ano_construção_obra* é uma propriedade da classe navegacional *obras*, especificadas respectivamente pelos atributos *name* e *navigational_class* do elemento ***attribute***, e *ano* é o nome do parâmetro que será informado para o contexto, especificado pelo elemento ***nav_parameter***.

Os critérios de ordenação utilizados para ordenar os elementos do contexto, definidos no campo "*Ordenação*" do cartão de especificação, são especificados utilizando-se os elementos ***order*** e ***order_element***. Na especificação do contexto "*Obras por Ano*", os elementos serão ordenados pelo valor da propriedade "*ano_construcao_obra*" da classe navegacional "*obras*" em ordem ascendente, como especificado pelos atributos *name*, *navigational_class* e *criteria* do elemento *order_element*.

As restrições de acesso aos elementos do contexto são especificadas utilizando-se o elemento ***restriction_use***. Por exemplo, a classe de usuários chamada "*visitante*" só tem permissão de leitura para os elementos do contexto "*Obras por Ano*".

4.1.3.2 Especificação das Estruturas de Acesso

As estruturas de acesso (índices) representadas no esquema de contextos são especificadas utilizando-se os elementos ***index*** (para índices simples) ou ***hierarch_index*** (para índices hierárquicos).

A especificação de uma estrutura de acesso também é feita com base nas informações contidas em seu cartão de especificação. Por exemplo, a estrutura de acesso "*Obras por*

Ano", representada no esquema de contextos da figura 4.3, é especificada com base no cartão de especificação abaixo.

Estrutura de Acesso**: Obras por Ano**	
Tipo**: simples**	
Parâmetros**: Ano**	
Elementos**: o: Obras ONDE o.ano_construção_obra = <ano>**	
Atributos **nome_obra .....**	Destino **Ctx Obras por ano (o)**
Ordenação**: por nome_obra, Ascendente**	
Usuários**: Visitante**	Permissão**: leitura**
Comentários**:**	
Depende de:	Influencia em:**:**

Essa estrutura de acesso é um índice simples. Portanto, sua especificação OOHDM-ML é feita utilizando-se o elemento index, como mostra o trecho de especificação abaixo.

```
<index id="indice_obras_por_ano" name="obras_por_ano_idx" type="static">
    <element_context context="contexto_obras_por_ano"/>
    <shown_attrib name="ano_construcao_obra" navigational_class="no_obras"/>
    <dynamic_selector>
        <shown_attrib name="nome_obra" navigational_class="no_obras"/>
        <selector_destination target="contexto_obras_por_ano"/>
    </dynamic_selector>
    <restriction_use permission="leitura">
        <user>Visitante</user>
    </restriction_use>
</index>
```

Todo índice também deve possuir um nome e um identificador único no documento OOHDM-ML, especificados pelos atributos *name* e *id*. No trecho de especificação acima, o índice "*Obras por Ano*" é formado pelos elementos do contexto "*Obras por Ano*" especificado no atributo *context* do elemento *element_context*. Os atributos comuns e os atributos seletores do índice são especificados pelos elementos ***shown_attrib***, ***dynamic_selector*** e ***static_selector***. Neste exemplo, o índice "*Obras por Ano*" apresenta a propriedade *ano_construção_obra*, especificada no atributo *name* do primeiro elemento *shown_attrib*, e o seletor dinâmico *nome_obra*, especificado pelo elemento *dynamic_selector*. Além do elemento *shown_attrib* que especifica o nome e o classe navegacional do seletor dinâmico, o elemento *dynamic_selector* possui também o elemento ***selector_destination***, cujo atributo *target* recebe como valor o identificador do contexto destino do atributo seletor.

4.1.3.3 Especificação dos Landmarks

A especificação OOHDM-ML de um landmark é feita utilizando-se o elemento *landmark*. Por exemplo, o landmark representado junto ao índice "*Obras por Ano*", representado no esquema de contextos da figura 4.3, é especificado da seguinte forma:

```
<landmark target="indice_obras_por_ano">Obras por Ano</landmark>
```

Neste exemplo, a âncora que representa o landmark receberá o nome "*Obras por Ano*" e terá como destino o índice identificado como "*indice_obras_por_ano*", especificado no atributo target do elemento landmark.

4.2 Aplicação "Coleção de CDs"

Para exemplificar a utilização da folha de estilo OOHDM-Translation, será utilizada uma versão simplificada do projeto OOHDM para o domínio "vendas de CDs", chamada "Coleção de CDs".

4.2.1 Modelo Conceitual

O modelo conceitual dessa aplicação possui apenas duas classes, *cds* e *artistas*, e o relacionamento entre seus objetos denominado "*artista grava CD*", como ilustra a figura abaixo.

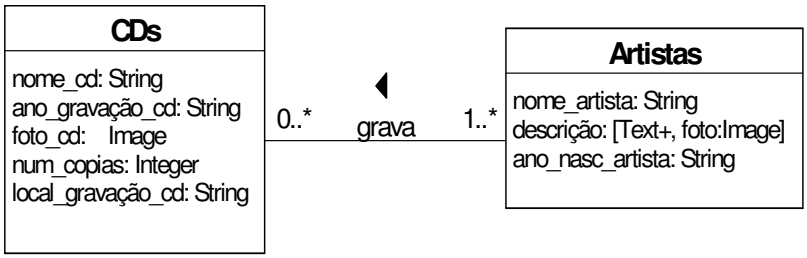


Figura 4.4 – Modelo Conceitual da aplicação "Coleção de CDs"

O modelo conceitual da aplicação e seu mapeamento para as tabelas do banco de dados, são representados no projeto OOHDM-Web pelas tabelas Lua *con_classes*, *map_classes*, *con_relationships* e *map_relationships*. Para gerar essas tabelas, a folha de estilo OOHDM-Translation contém regras de template que serão aplicadas à sub-árvore do elemento ***conceptual_model*** do documento OOHDM-ML. A tabela abaixo apresenta o pseudo-código, que descreve a lógica de transformação que será usada para gerar essas tabelas Lua, e a regra de template que implementa esta lógica.

Pseudo-Código	Regra de Template XSLT

Para o elemento *conceptual_model* do documento XML associado à folha de estilo faça
Crie a estrutura Lua con_classes incluindo o formato abaixo:
con_classes =
{
Processe recursivamente todos os elementos filhos c*onceptual_class*.
Complete a estrutura con_classes utilizando o símbolo "}" , alinhando –o com o símbolo "{" inicial.
Crie a estrutura Lua map_classes incluindo o formato abaixo:
map_classes =
{
Processe recursivamente todos os elementos filhos *conceptual_class* e todos os elementos *map_conceptual_class* relacionados a eles.
Complete a estrutura map_classes utilizando o símbolo "}" , alinhando –o com o símbolo "{" inicial.
Se o elemento conceptual_model possui algum elemento filho *relationship*
então
Crie a estrutura Lua con_relationships incluindo o formato abaixo:
con_relationships =
{
Processe recursivamente todos os elementos filhos *relationship*.
Complete a estrutura con_relationships utilizando o símbolo "}" , alinhando-o com o símbolo "{" inicial.
Crie a estrutura Lua map_relationships incluindo o formato abaixo:
map_relationships =
{
Processe todos os elementos *relationship* e todos os elementos *map_relationship* relacionados a eles.
Complete a estrutura map_relationships utilizando o símbolo "}", alinhando –o com o símbolo "{" inicial.
fim-Se.
Fim-Para.
	<xsl:template match="conceptual_model">
	
	con_classes =
	{
	
	<xsl:apply-templates select="conceptual_class"/>
	
	}
	
	map_classes =
	{
	
	<xsl:apply-templates select="conceptual_class"
	mode="map_classes"/>
	
	}
	
	<xsl:if test="relationship">
	
	con_relationships =
	{
	
	<xsl:apply-templates select="relationship"/>
	
	}
	
	map_relationships =
	{
	
	<xsl:apply-templates select="relationship"
	mode="map_relationships"/>
	
	}
	
	</xsl:if>
	</xsl:template>

Tabela 4.1 – Regra de template para o elemento conceptual_model

Como o documento OOHDM-ML que contém a especificação do projeto da aplicação só possui um único elemento *conceptual_model*, as tabelas Lua que representam o modelo conceitual serão geradas apenas uma vez. A regra de template apresentada na tabela acima

gera apenas a estrutura de cada tabela Lua que descreve as primitivas do modelo conceitual. O conteúdo de cada uma delas será gerado por regras de templates específicas, processadas pelas instruções <*xsl:apply-templates*>. Assim, o resultado inicial dessa primeira regra de template sobre a especificação OOHDM-ML é o seguinte:

Especificação OOHDM-ML	Resultado no Projeto OOHDM-Web

<?xml version="1.0"
encoding="UTF-8"?>
<?xml-stylesheet type="text/xml
href="oohdm_translation.xsl"?>
<!DOCTYPE OOHDM SYSTEM OOHDM_ML.dtd">
<OOHDM>
<OOHDM_BASE>
<**conceptual_model**>
<conceptual_class>
…
</conceptual_class>
…
<relationship>
…
</relationship>
…
</**conceptual_model**>
</OOHDM_BASE>
<OOHDM_WEB>
<map_conceptual_class>
…
</map_conceptual_class>
…
<map_relationship>
…
</map_relationship>
…
</OOHDM_WEB>
</OOHDM>
	con_classes =
	{
	…
	}
	
	map_classes =
	{
	…
	}
	
	con_relationships =
	{
	…
	}
	
	map_relationships =
	{
	…
	}

Tabela 4.2 – Tabelas Lua geradas pela regra de template do elemento conceptual_model

O conteúdo da tabela *con_classes*, ou seja, a descrição de seus atributos é gerada pela regra de template que processa todos os elementos ***conceptual_class*** do documento OOHDM-ML. A tabela abaixo apresenta o pseudo-código e o código XSTL da regra de template que é aplicada à sub-árvore do elemento *conceptual_class* quando a instrução <*xsl:apply-templates select="conceptual_class"/*> é processada.

Pseudo-Código	Regra de Template XSLT

Para todos os elementos *conceptual_class* faça
Inclua na estrutura Lua *con_classes* o valor do atributo **name** seguido dos caracteres "=" e "{" da seguinte forma:
<valor_do_atributo_name> =
{
Se existe um atributo
superclass para o elemento conceptual_class
então
Tratar_Atributo_Superclass
Se existe algum elemento
attrib filho do elemento conceptual_class
então
Inclua uma vírgula.
fim-Se.
fim-Se.
Para cada elemento *attrib*, filho do elemento conceptual_class faça
Tratar_Elementos_Attrib.
Se o elemento attrib que está
sendo tratado é o último elemento filho
então
Inclua na estrutura Lua con_classes o símbolo }
senão
Inclua na estrutura Lua con_classes o símbolo } seguido de vírgula
fim-Se
fim-Para.
 	
Inclua na estrutura Lua con_classes o símbolo }
Se o elemento conceptual_class
não é o último elemento tratado
então
Inclua uma vírgula seguindo o símbolo }
incluído anteriormente
fim-Se
fim-Para.
	<xsl:template match="conceptual_class">
	
	<xsl:value-of select="@name"/>
	<xsl:text> =
	{
	</xsl:text>
	
	<xsl:if test="@superclass">
	<xsl:call-template
	name="Tratar_Atributo_Superclass"/>[7]
	<xsl:if test="attrib">
	<xsl:text>,
	</xsl:text>
	</xsl:if>
	</xsl:if>
	<xsl:for-each select="attrib">
	<xsl:call-template
	name="Tratar_Elementos_Attrib"/>[8]
	<xsl:choose>
	<xsl:when test="position()=last()">
	<xsl:text>
	}
	</xsl:text>
	</xsl:when>
	<xsl:otherwise>
	<xsl:text>
	},
	</xsl:text>
	</xsl:otherwise>
	</xsl:choose>
	</xsl:for-each>
	
	<xsl:text>}</xsl:text>
	<xsl:if test="position()!=last()">
	<xsl:text>,
	</xsl:text>
	</xsl:if>
	</xsl:template>

Tabela 4.3 - Regra de template para o elemento conceptual_class

A regra de template apresentada acima gera a descrição OOHDM-Web das classes conceituais *cds* e *artistas*, e seus atributos, especificadas no documento OOHDM-ML pelos elementos ***conceptual_class***, ***attrib*** e ***perspective_attrib***, como mostra a tabela abaixo.

[7] O código fonte do template *Tratar_Atributo_Superclass* pode ser consultado no Apêndice IV.
[8] O código fonte do template *Tratar_Elementos_Attrib* pode ser consultado no Apêndice IV.

Especificação OOHDM-ML	Resultado no Projeto OOHDM-Web

<conceptual_model>
 	
<**conceptual_class** id="id_cds" name="**cds**">
 	
<attrib name="**chave_cd**" type="string"
size="20"/>
<attrib name="nome_cd" type="string"
size="70"/>
<attrib name="ano_gravacao_cd" type="string"
size="4"/>
<attrib name="foto_cd" type="image"
size="30"/>
<attrib name="num_copias" type="integer"
size="30"/>
<attrib name="local_gravacao_cd" type="string"
size="50"/>
 	
</**conceptual_class**>
 	
<**conceptual_class** id="id_artistas" name="**artistas**">
<attrib name="**chave_artista**" type="string"
size="20"/>
<attrib name="nome_artista" type="string"
size="50"/>
<attrib name="descricao_artista" type="text">
<perspective name="descricao_artista"
type="text" default="yes"/>
<perspective name="foto_artista"
type="image" size="50"/>
</attrib>
<attrib name="ano_nasc_artista" type="string"
size="4"/>
</**conceptual_class**>
</conceptual_model>
	con_classes =
	{
	cds =
	{
	chave_cd =
	{
	attrib_type = "text",
	attrib_size = 20,
	prim_key = "yes"
	},
	nome_cd =
	{
	attrib_type = "text",
	attrib_size = 70
	},
	ano_gravacao_cd =
	{
	attrib_type = "text",
	attrib_size = 4
	},
	…
	},
	artistas =
	{
	chave_artista =
	{
	attrib_type = "text",
	attrib_size = 20,
	prim_key = "yes"
	},
	nome_artista =
	{
	attrib_type = "text",
	attrib_size = 50
	},
	descricao_artista=
	{
	attrib_type = "memo"
	},
	foto_artista=
	{
	attrib_type = "text",
	attrib_size = 50
	}, ...
	}
	}

Tabela 4.4 – Conteúdo da tabela Lua con_classes gerado pela regra de template XSLT

Para cada elemento *attrib* será gerada uma tabela Lua contendo os campos attrib_type, attrib_size e prim_key, cujos valores serão obtidos pela regra de template definida para tratar atributos.

O conteúdo da tabela *map_classes*, que descreve o mapeamento das classes conceituais e suas propriedades para as tabelas de classe do banco de dados, é gerado por uma outra regra de template que também processa todos os elementos *conceptual_class* do documento OOHDM-ML. A diferenciação da regra de template a ser aplicada é feita pelo atributo *mode*

do elemento <*xsl:template*>, como pode ser observado no código da regra de template abaixo.

Regra de Template XSLT [9]

```
<xsl:template match="conceptual_class" mode="map_classes">
  <!-- Variavel que armazena o valor do atributo id do elemento conceptual_class corrente -->
  <xsl:variable name="var_id_conceptual_class" select="@id"/>
  <!-- Variavel que armazena o caminho na árvore origem para encontrar o elemento
       map_conceptual_class correspondente ao elemento conceptual_class corrente -->
  <xsl:variable name="var_map_conceptual_class"
       select="//OOHDM_WEB/map_conceptual_class[@conceptual_class=$var_id_conceptual_class]"/>
  <!-- Variavel que armazena o nome da tabela de classe do banco de dados. Este nome pode
       ser o valor do atributo db_class do elemento map_conceptual_class ou o valor do
       atributo name do elemento conceptual_class corrente, caso nao exista um elemento
       map_conceptual_class especificado. -->

  <xsl:variable name="var_db_class">
    <xsl:choose>
      <xsl:when test="$var_map_conceptual_class/@db_class">
        <xsl:value-of select="$var_map_conceptual_class/@db_class"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="@name"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <!--Inclua na estrutura Lua map_classes o valor do atributo name do elemento
      conceptual_class corrente, seguido pelos caracteres "=" e "{" -->
  <xsl:value-of select="@name"/> =
   {
  <!--Inclua na estrutura Lua map_classes a string 'db_class =' seguida pelo nome da classe
      conceitual na tabela do banco de dados. -->
    db_class = "<xsl:value-of select="$var_db_class"/>"
  <!--Se existe mapeamento da propriedades para os campos das tabelas do banco de dados,
      então inclua a 'string attributes {'e chame a regra de template que gera a descrição desse
      mapeamento -->
    <xsl:if test="$var_map_conceptual_class/map_attribute">,
      attributes =
       {
        <xsl:call-template name="Tratar_Elementos_Map_Attribute">[10]
            <xsl:with-param name="par_map_class" select="$var_map_conceptual_class"/>
        </xsl:call-template>
    }</xsl:if>
   }<xsl:if test="position()!=last()">,
  </xsl:if>
</xsl:template>
```

Tabela 4.5 - Regra de template para o elemento conceptual_class (mode map_classes)

[9] Neste exemplo, o pseudo-código é apresentado junto ao código em forma de comentário por motivos de legibilidade.

[10] O código fonte do template *Tratar_Elementos_Map_Attribute* pode ser consultado no Apêndice IV.

A regra de template apresentada acima processa todos os elementos *conceptual_class* da sub-árvore do elemento *conceptual_model* e também alguns elementos da sub-árvore do elemento *OOHDM_WEB*, que especifica o mapeamento das primitivas do projeto OOHDM para o ambiente OOHDM-Web. Na tabela abaixo exemplificamos apenas o mapeamento da classe conceitual *cds* para a tabela de classe do banco de dados *cds*, onde o atributo *ano_gravacao_cd* é mapeado para o campo "*ano_gravacad*".

Especificação OOHDM-ML	**Resultado no Projeto OOHDM-Web**

<OOHDM>
<OOHDM_BASE>
<conceptual_model>
<**conceptual_class** id="id_cds" name="**cds**">
<attrib name="chave_cd" type="string"
size="20"/>
<attrib name="nome_cd" type="string"
size="70"/>
<attrib name="ano_gravacao_cd" type="string"
size="4"/>
<attrib name="foto_cd" type="image"
size="30"/>
<attrib name="num_copias" type="integer"
size="30"/>
<attrib name="local_gravacao_cd"
type="string" size="50"/>
</**conceptual_class**>
…
</conceptual_model>
</OOHDM_BASE>
<**OOHDM_WEB**>
<map_conceptual_class
conceptual_class="id_cds">
<**map_attribute** id="map_ano_gravacao"
name="ano_gravacao_cd"
db_attribute="**ano_gravacao**"/>
</map_conceptual_class>
…
</**OOHDM_WEB**>
</OOHDM>
	map_classes =
	{
	cds =
	{
	db_class = "**cds**",
	attributes =
	{
	ano_gravacao_cd = "**ano_gravacao**"
	}
	},
	…
	}

Tabela 4.6 - Conteúdo da tabela Lua map_classes gerado pela regra de template XSLT

O conteúdo da tabela *con_relationships* é gerado pela regra de template que processa todos os elementos *relationship* do documento OOHDM-ML. A tabela abaixo apresenta o pseudo-código e o código XSTL da regra de template que é aplicada à sub-árvore do elemento *relationship* quando a instrução <*xsl:apply-templates select="relationship"/*> é processada.

Pseudo-Código	Regra de Template XSLT
Para todos os elementos relationship encontrados no documento XML associado faça Inclua na estrutura Lua con_relationships: ▪ o valor do atributo name do elemento relationship seguido dos caracteres "=" e "{" da seguinte forma: *<valor_do_atributo_name>* = { ▪ a string "*source_class =*" seguida pelo valor do atributo **name** do elemento *conceptual_class*, cujo atributo id tenha o mesmo valor do atributo source_class do elemento relationship. O valor do atributo name deve estar entre aspas (") e seguido de vírgula ▪ a string "*destination_class =*" seguida pelo valor do atributo **name** do elemento *conceptual_class*, cujo atributo id tenha o mesmo valor do atributo target_class do elemento relationship. O valor do atributo name deve estar entre aspas (") e seguido de vírgula. ▪ a string "source_*card =*" seguida pelo valor do atributo *source_cardinality* do elemento relationship. Este valor deve estar entre aspas (") e seguido de vírgula. ▪ a string "dest_*card =*" seguida pelo valor do atributo *target_cardinality* do elemento relationship. Este valor deve estar entre aspas ("). Se o atributo bidirectional estiver especificado no elemento map_relationship, cujo atributo id_relationship tenha o mesmo valor do atributo id do elemento relationship que está sendo tratado então Inclua uma vírgula após o formato dest_card = "*<valor_do_atributo target_cardinality>*" Inclua a string *bidirectional* seguida pelo valor do atributo *bidirectional* do elemento map_relationship Inclua o símbolo } alinhando-o com o símbolo { inserido após o valor do atributo name. fim-Se Se o elemento relationship não é o último elemento encontrado então Inclua uma vírgula seguindo o símbolo } incluído anteriormente fim-Se fim-Para.	<xsl:template match="relationship"> <xsl:value-of select="@name"/> = { source_class = "<xsl:value-of select="id(@source_class)/@name"/>", destination_class = "<xsl:value-of select="id(@target_class)/@name"/>", source_card = "<xsl:value-of select="@source_cardinality"/>", dest_card = "<xsl:value-of select="@target_cardinality"/>" <xsl:if test="$var_atributo_bidirectional">, bidirectional = <xsl:value-of select="$var_atributo_bidirectional"/> </xsl:if> }<xsl:if test="position()!=last()">, </xsl:if> </xsl:template>

Tabela 4.7 - Regra de template para o elemento relationship

A regra de template apresentada acima gera a descrição OOHDM-Web de todos os relacionamentos do modelo conceitual, especificados no documento OOHDM-ML pelos elementos ***relationship***. No modelo conceitual do site "Coleção de CDs" só existe um relacionamento "*artista grava CD*", como mostra a tabela abaixo. Todas as informações sobre o relacionamento são obtidas com base nos valores especificados nos atributos do elemento relationship.

Especificação OOHDM-ML	Resultado no Projeto OOHDM-Web

<conceptual_model>
<**relationship** id="relacao_grava"
name="artista_grava_cd"
source_class="id_artistas"
target_class="id_cds"
source_cardinality="n"
target_cardinality="n">
</**relationship**>
</conceptual_model>
	con_relationships =
	{
	artista_grava_cd =
	{
	source_class = "artistas",
	destination_class = "cds",
	source_card = "n",
	dest_card = "n"
	}
	}

Tabela 4.8 - Conteúdo da tabela Lua relationship gerado pela regra de template XSLT

4.2.2 Projeto de Navegação

O projeto de navegação de uma aplicação hipermídia é representado no projeto OOHDM-Web pelas tabelas Lua *queries*, *map_class_ctx*, *nav_contexts* e *nav_indexes*. Para gerar estas tabelas, a folha de estilo OOHDM-Translation contém regras de template que serão aplicadas à sub-árvore do elemento ***navigational_model*** do documento OOHDM-ML, como mostra a tabela abaixo.

Pseudo-Código	Regra de Template XSLT

Para o elemento *navigational_model* do documento XML associado à folha de estilo faça
Crie a estrutura Lua queries incluindo o formato abaixo:
queries =
{
Processe recursivamente todos os elementos filhos navigation*al_class*.
Complete a estrutura queries utilizando o símbolo "}" , alinhando –o com o símbolo "{" inicial.
Se o elemento navigational_model possui algum elemento filho *context_class* então
Crie a estrutura Lua map_classes_ctx incluindo o formato abaixo:
map_classes_ctx =
{
Processe recursivamente todos os elementos filhos *context_class* e todos elementos *map_context_class* relacionados a eles.
Complete a estrutura map_classes_ctx utilizando o símbolo "}" , alinhando-o com o símbolo "{" inicial.
fim-Se.
Crie a estrutura Lua nav_contexts incluindo o formato abaixo:
nav_contexts =
{
Processe recursivamente todos os elementos *navigational_context*.
Complete a estrutura nav_contexts utilizando o símbolo "}", alinhando –o com o símbolo "{" inicial.
Crie a estrutura Lua nav_indexes incluindo o formato abaixo:
nav_indexes =
{
Processe recursivamente todos os elementos *index ou hierarch_index*.
Complete a estrutura nav_indexes utilizando o símbolo "}", alinhando –o com o símbolo "{" inicial.
Fim-Para.
	<xsl:template match="navigational_model">
	queries =
	{
	<xsl:apply-templates select="navigational_class"/>
	}
	<xsl:if test="context_class">
	map_classes_ctx =
	{
	<xsl:apply-templates select="context_class"/>
	}
	</xsl:if>
	nav_contexts =
	{
	<xsl:apply-templates select="navigational_context"/>
	}
	nav_indexes =
	{
	<xsl:apply-templates select="index|hierarch_index"/>
	}
	</xsl:template>

Tabela 4.9 – Regra de template para o elemento navigational_model

O documento OOHDM-ML que contém a especificação do projeto da aplicação só possui um único elemento *navigational_model*. Assim, as tabelas Lua que representam o projeto de navegação serão criadas apenas uma vez. De forma semelhante ao modelo conceitual, essa

regra de template gera apenas as estruturas das tabelas Lua. O conteúdo de cada tabela é gerado por regras de template específicas processadas pelas instruções *<xsl:apply-templates>*. A tabela abaixo mostra o resultado do processamento da regra de template acima.

Especificação OOHDM-ML	Resultado no Projeto OOHDM-Web

<OOHDM>
<OOHDM_BASE>
<conceptual_model>
…
</conceptual_model>
<**navigational_model**>
<navigational_class>
…
</navigational_class>
…
<link/>
…
<navigational_context>
…
</navigational_context>
…
<context_class>
…
</context_class>
…
<index>
…
</index>
…
<hierarch_index>
…
</hierarch_index>
…
</ **navigational_model**>
</OOHDM_BASE>
</OOHDM>
	queries =
	{
	…
	}
	
	map_classes_ctx =
	{
	…
	}
	
	nav_contexts =
	{
	…
	}
	
	nav_indexes =
	{
	…
	}

Tabela 4.10 – Tabelas Lua geradas pela regra de template do elemento navigational_model

4.2.2.1 Modelo de Navegação

O modelo de navegação da aplicação "Coleção de CDs" é formado pelas classes navegacionais *cds* e *artistas*, e pelos elos entre os objetos dessas classes, como mostra a figura 4.5 abaixo:

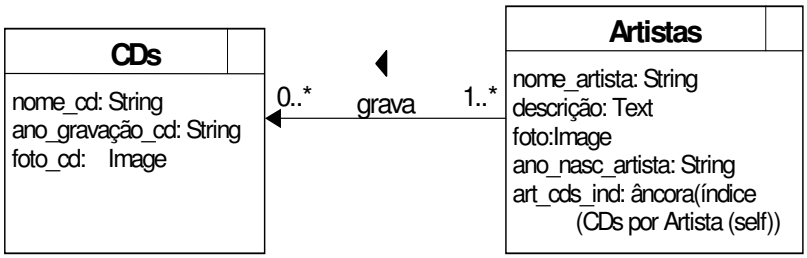


Figura 4.5 – Modelo de Navegação da aplicação "Coleção de CDs"

As classes navegacionais (nós) são representadas no projeto OOHDM-Web pela tabela Lua *queries*. O conteúdo desta tabela é gerado com base na especificação OOHDM-ML dos nós de navegação e dos elos que compõem o modelo de navegação da aplicação, representados pelos elementos *navigational_class* e *link*, respectivamente.

A partir da especificação das classes navegacionais, as regras de template do OOHDM-Translation geram as "queries" que obtêm as instâncias que serão apresentadas em cada nó. Essas "queries" são geradas com base nos elementos ***navigational_class***. Para cada elemento *navigational_class* do documento OOHDM-ML, são identificadas as tabelas de classe que possuem os valores para as propriedades do nó e que farão parte da "query". Estas tabelas são identificadas a partir do valor do atributo *conceptual_class* de cada elemento *nav_attrib* ou *perspective_attrib*, filhos do elemento *navigational_class*, e dos elementos de mapeamento ***map_conceptual_class*** existentes para cada classe conceitual especificada. Quando um nó possui atributos de classes conceituais diferentes, as tabelas de relacionamento entre essas classes também são identificadas. Elas são obtidas a partir do valor do atributo *db_relationship* do elemento *map_relationship*, que representa o mapeamento dos relacionamentos no banco de dados.

A tabela abaixo mostra o conteúdo gerado para a tabela Lua *queries* para a classe navegacional *cds*, obtido a partir da especificação do elemento *navigational_class* no documento OOHDM-ML.[11]

[11] O código da folha de estilo queries.xsl correspondente a essa transformação não foi incluído por questões de legibilidade. No entanto, o mesmo pode ser consultado no Apêndice IV.

Especificação OOHDM-ML	Resultado no Projeto OOHDM-Web

<OOHDM_BASE>
<conceptual_model>
<conceptual_class id="id_cds" name="**cds**">
<attrib name="chave_cd" type="string"
size="20"/>
<attrib name="nome_cd" type="string"
size="70"/>
<attrib name="ano_gravacao_cd"
type="string" size="4"/>
<attrib name="foto_cd" type="image"
size="30"/>
<attrib name="num_copias" type="integer"
size="30"/>
<attrib name="local_gravacao_cd"
type="string" size="50"/>
</conceptual_class>
</conceptual_model>
<navigational_model>
<**navigational_class** id="no_cds" name="cds">
<nav_attrib name="**nome_cd**"
conceptual_class="id_cds"/>
<nav_attrib name="ano_gravacao_cd"
conceptual_class="id_cds"/>
<nav_attrib name="**foto_cd**"
conceptual_class="id_cds"/>
</**navigational_class**>
</navigational_model>
</OOHDM_BASE>
<OOHDM_WEB>
<**map_conceptual_class** conceptual_class="id_cds">
<**map_attribute** id="map_ano_gravacao"
name="ano_gravacao_cd"
db_attribute="***ano_gravacao***"/>
</**map_conceptual_class**>
</OOHDM_WEB>
	queries =
	{
	cds =
	{
	class = {"**cds**"},
	tables = {"cds"},
	fields =
	{
	"cds.**nome_cd**",
	"cds.***ano_gravacao***",
	"cds.**foto_cd**"
	},
	class_ctx =
	{
	…
	}
	}
	…
	}

Tabela 4.11 - Conteúdo da tabela Lua queries gerado pela regra de template XSLT

Neste exemplo, a classe navegacional *cds* apresentará apenas as propriedades *nome_cd*, *ano_gravacao_cd* e *foto_cd*, cujos valores de instância serão obtidos da mesma tabela de classe *cds* que representa a classe conceitual *cds*, indicada no atributo *conceptual_class* de cada elemento ***nav_attrib***, filho do elemento *navigational_class*. O nome da tabela de classe e os nomes de seus campos são obtidos considerando o mapeamento especificado pelo elemento *map_conceptual_class*. Vale notar que, como nenhum valor foi especificado para o atributo *db_class* do elemento *map_conceptual_class*, a tabela de classe é criada no banco de dados com o mesmo nome da classe conceitual que ela representa.

A tabela acima mostra que o conteúdo da tabela Lua *queries* possui um campo chamado *class_ctx*. Este campo deve conter a descrição das classes em contexto que serão utilizadas pela aplicação. O conteúdo deste campo é gerado com base na especificação das classes em contexto representada pelos elementos ***context_class*** e ***map_context_class*** da linguagem OOHDM-ML, como mostra a tabela abaixo.

Especificação OOHDM-ML	Resultado no Projeto OOHDM-Web

<OOHDM_BASE>
<conceptual_model>
<conceptual_class id="id_cds" name="cds">
<attrib name="chave_cd" type="string"
size="20"/>
<attrib name="nome_cd" type="string"
size="70"/>
<attrib name="ano_gravacao_cd"
type="string" size="4"/>
<attrib name="foto_cd" type="image"
size="30"/>
<attrib name="num_copias" type="integer"
size="30"/>
<attrib name="local_gravacao_cd"
type="string" size="50"/>
</conceptual_class>
</conceptual_model>
<navigational_model>
<navigational_class id="no_cds" name="cds">
<nav_attrib name="nome_cd"
conceptual_class="id_cds"/>
<nav_attrib name="ano_gravacao_cd"
conceptual_class="id_cds"/>
<nav_attrib name="foto_cd"
conceptual_class="id_cds"/>
</navigational_class>
<**context_class** id="classe_ctx_cds_por_ano"
base_class="no_cds"
contexts="contexto_cds_por_ano">
<nav_attrib name="***local_gravacao_cd***"
conceptual_class="id_cds"/>
</**context_class**>
</navigational_model>
</OOHDM_BASE>
<OOHDM_WEB>
<**map_context_class**
context_class="classe_ctx_cds_por_ano"
db_class_ctx="**cds_ano**">
<map_attribute id="map_local_gravacao_cd"
name="local_gravacao_cd"
db_attribute="**local_gravacao**"/>
</**map_context_class**>
</OOHDM_WEB>
	queries =
	{
	cds =
	{
	class = {"cds"},
	tables = {"cd"},
	fields =
	{
	"cds.nome_cd",
	"cds.ano_gravacao",
	"cds.foto_cd"
	},
	class_ctx =
	{
	cds_por_ano =
	{
	local_gravacao_cd =
	{
	attrib_type = "text",
	attrib_size = 50
	}
	}
	}
	}
	…
	}
	map_classes_ctx =
	{
	cds =
	{
	db_class_ctx = "**cds_ano**",
	attrib_class_ctx =
	{
	local_gravacao_cd = "**local_gravacao**"
	}
	}
	}

Tabela 4.12 - Conteúdo das tabelas Lua class_ctx e map_classes_ctx

Neste exemplo, a classe em contexto *classe_ctx_cds_por_ano* é criada para o nó *cds*, especificado no atributo *base_class* do elemento *context_class*. Esta classe em contexto especifica que a propriedade *local_gravacao_cd* será mostrada sempre que o nó *cds* for apresentado no contexto "*cds_por_ano*" (atributo *contexts* do elemento *context_class*).

O elemento ***map_context_class*** especifica o mapeamento das classes em contexto para as tabelas de classe do banco de dados, representado no projeto OOHDM-Web pela tabela Lua *map_classes_ctx*, como mostra a tabela acima. Este elemento descreve o nome da tabela de classe e os nomes de seus campos, utilizando o atributo *db_class_ctx* e o elemento

map_attribute, respectivamente. Neste exemplo, o nome considerado para a propriedade *local_gravacao_cd* foi mapeado no atributo *db_attribute* do elemento *map_attribute* para o campo *local_gravacao*.

4.2.2.2 Contextos de Navegação e Estruturas de Acesso

Os contextos de navegação e as estruturas de acesso (índices), representados no esquema de contextos de uma aplicação hipermídia, são representados no projeto OOHDM-Web pelas tabelas Lua *nav_contexts* e *nav_indexes*, respectivamente. A figura 4.6 apresenta o esquema de contextos de navegação para a aplicação "Coleção de CDs".

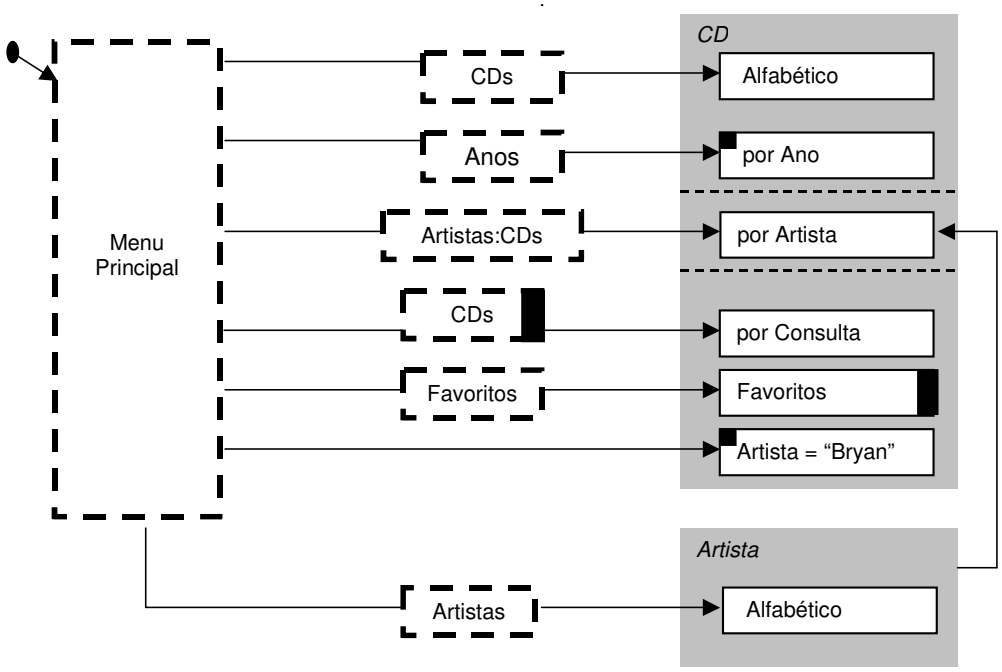


Figura 4.6 – Esquema de Contextos do site "Coleção de CDs"

O conteúdo da tabela *nav_contexts*, ou seja, a descrição dos contextos de navegação é gerada pela regra de template que processa todos os elementos ***navigational_context*** do documento OOHDM-ML. Por exemplo, a tabela abaixo mostra o resultado do processamento da folha de estilo nav_contexts.xsl[12] para gerar a descrição do contexto "*CDs por Artista*" a partir de sua especificação no elemento *navigational_context*.

[12] O código da folha de estilo *nav_contexts.xsl* correspondente a essa regra de template não foi incluído por questões de legibilidade. No entanto, o mesmo pode ser consultado no Apêndice IV.

Especificação OOHDM-ML	Resultado no Projeto OOHDM-Web

<OOHDM_BASE>
<navigational_model>
<**navigational_class** id="no_cds" name="***cds***">
…
</**navigational_class**>
<navigational_class id="no_artistas"
name="artistas">
…
</navigational_class>
<**link** id="link_grava" name="***artista_grava_cd***"
source_class="no_artistas"
target_class="no_cds"
source_cardinality="n"
target_cardinality="n">
</**link**>
<**navigational_context** id="ctx_cds_por_artista"
name="***cds_por_artista***"
element_class="no_cds"
type="static"
navigation_type="sequential">
<***selection***>
<equal>
<attribute name="***chave_artista***"
navigational_class="no_artistas"
link="*link_grava*"/>
<nav_parameter>***chave_artista***
</nav_parameter>
</equal>
</selection>
<order id="ordenacao_cds_por_artista"
default="yes">
<***order_element*** name="***nome_cd***"
navigational_class="no_cds"
criteria="***asc***"/>
</order>
</**navigational_context**>
…
</navigational_model>
</OOHDM_BASE>
<OOHDM_WEB>
…
<**interface_element** target="no_cds">***cd.html***
</**interface_element**>
…
</OOHDM_WEB>
	nav_contexts =
	{
	cds_por_artista =
	{
	context = "EG",
	nav_type = "S",
	parameters = {"***chave_artista***"},
	elements =
	{
	cds =
	{
	order = {***nome_cd*** = "***asc***"},
	dest_page = "***cd.html***",
	selection =
	{
	link_name = {"***artista_grava_cd***"},
	condition = "(***chave_artista*** =
	#***chave_artista***)"
	}
	}
	}
	},
	...
	}

Tabela 4.13- Conteúdo da tabela Lua nav_contexts gerado pela folha de estilo nav_contexts.xsl

O exemplo acima mostra que o conteúdo da tabela Lua nav_contexts é gerado com base na especificação OOHDM-ML dos contextos. Para cada elemento *navigational_context* será gerada uma tabela Lua com o nome do contexto, obtido do atributo name desse elemento. Neste exemplo, foi gerada uma tabela para o contexto "*CDs por Artista*". Os valores para os campos context e nav_type da tabela ***cds_por_artista***, foram obtidos considerando os

valores dos atributos *type e navigational_typ*e do elemento *navigational_context*. O conteúdo da tabela parameters foi obtido do elemento *nav_parameter*, filho do elemento *selection*.

A tabela Lua elements é criada com o nome das classes navegacionais informadas no atributo *element_class*. Para cada valor informado, a regra de template busca o valor do atributo name do elemento ***navigational_class***, cujo valor do atributo id é igual ao valor informado. Neste exemplo, a classe navegacional informada é ***cds***. O critério de ordenação descrito pelo campo order da tabela Lua cds é obtido considerando os valores dos atributos *name* e *criteria* do elemento ***order_element***. Já o campo dest_page descreve o nome da página que será usada para apresentar as informações do nó. Este nome é obtido do elemento ***interface_element***, especificado no mapeamento do projeto OOHDM para o ambiente OOHDM-Web, representado pelo elemento OOHDM_WEB.

Por último, o campo *condition* da tabela Lua selection, que representa a descrição da condição utilizada para recuperar os elementos do contexto, é criado utilizando as informações especificadas no elemento ***selection***. Como o contexto "*CDs por Artista*" é um grupo de contexto derivado de elo, é necessário informar o elo utilizado para obter os elementos do contexto. Neste caso, o elo "*artista_grava_cd*" deve ser informado no campo link_name da tabela selection. Seu valor é obtido do atributo name do elemento ***link***, cujo valor do atributo id é igual ao valor informado no atributo link do elemento *navigational_context*.

A especificação das estruturas de acesso (índices) representadas no esquema de contexto de navegação é realizada utilizando-se o elemento ***index***, para estruturas de acesso simples, e o elemento ***hierarch_index*** para as estruturas de acesso hierárquicas. Estes elementos são processados pelas regras de template da folha de estilo *nav_indexes.xsl*[13] para gerar o conteúdo da tabela Lua nav_indexes do projeto OOHDM-Web. A tabela abaixo mostra a descrição Lua do índice hierárquico "*Artistas:CDs*", gerada a partir de sua especificação OOHDM-ML utilizando o elemento *hierarch_index*.

[13] O código da folha de estilo *nav_indexes.xsl* não foi incluído neste capítulo por questões de legibilidade. No entanto, o mesmo pode ser consultado no Apêndice IV.

Especificação OOHDM-ML	Resultado no Projeto OOHDM-Web

OOHDM_BASE>
<navigational_model>
…
<**hierarch_index** id="indice_cds_por_artista_H"
name="***cds_por_artista_H_idx***"
type="static">
<**element_context** context="*ctx_cds_por_artista*"/>
<level>
<**dynamic_selector**>
<shown_attrib name="***nome_artista***"
navigational_class="no_artistas"/>
<selector_destination
target="indice_cds_por_artista_H"/>
</**dynamic_selector**>
</level>
<level>
<shown_attrib name="***ano_gravacao_cd***"
navigational_class="no_cds"/>
<**dynamic_selector**>
<shown_attrib name="***nome_cd***"
navigational_class="no_cds"/>
<selector_destination
target="ctx_cds_por_artista"/>
</**dynamic_selector**>
</level>
</**hierarch_index**>
…
</navigational_model>
</OOHDM_BASE>
<OOHDM_WEB>
…
<**interface_element** target="no_cds">***cd.html***
</**interface_element**>
<**interface_element**
target="indice_cds_por_artista_H">
cds_artista_idx.html
</**interface_element**>
…
</OOHDM_WEB>
	nav_indexes =
	{
	cds_por_artista_H_idx =
	{
	type = "EH",
	parameters = {"chave_artista"},
	elements =
	{
	group =
	{
	context = "***cds_por_artista***",
	attributes = { {"***nome_artista***"},
	{"***ano_gravacao_cd***",
	"***nome_cd***"} },
	selectors =
	{
	nome_artista =
	{
	destination =
	“cds_por_artista_H_idx,
	#chave_artista",
	dest_page =
	“***cds_artista_idx.html***",
	inst = {"chave_artista"}
	},
	nome_cd =
	{
	destination = "cds_por_artista,
	#chave_artista",
	dest_page = "***cd.html***",
	inst = {"chave_cd"}
	}
	}
	}
	}
	},
	...
	}

Tabela 4.14 - Conteúdo da tabela Lua nav_indexes gerado pela folha de estilo nav_indexes.xsl

No exemplo acima, foi criada uma tabela Lua com o nome do índice hierárquico "*CDs por Artista*" especificado no atributo name do elemento *hierarch_index* (***cds_por_artista_H_idx***). Os valores para a tabela *parameters* e para o campo *context* da tabela elements foram obtidos a partir das informações especificadas no elemento *navigational_context*, cujo atributo id tem o mesmo valor do atributo *target* informado no elemento ***element_context***. O conteúdo da tabela *attributes* será criado com base nos valores do atributo *name* de todos os elementos ***shown_attrib*** especificados para os elementos *level* e *dynamic_selector*.

Os nomes dos atributos seletores do índice, descritos na tabela *Lua selectors*, serão obtidos com base nos elementos ***shown_attrib***, filhos do elemento ***dynamic_selector***. O destino de cada atributo seletor descrito pelo campo destination da tabela Lua *selectors* será obtido através do atributo target do elemento ***selector_destination***. O valor para o campo dest_page de cada atributo seletor será obtido de acordo com o destino do seletor especificado pelo elemento *selector_destination* no documento OOHDM-ML. Se o valor do atributo target deste elemento for o identificador de um índice, o valor do campo dest_page será igual ao valor do elemento ***interface_element***, cujo atributo target tem este identificador como valor . Caso contrário, o valor do campo dest_page será o valor do elemento interface_element, cujo atributo target tem valor igual ao atributo navigational_class do elemento ***shown_attrib*** que especifica o atributo seletor. Por fim, o valor da tabela Lua inst de cada atributo seletor será obtido a partir do valor do atributo navigation_class do elemento *shown_attrib* que especifica cada atributo seletor. Utilizando este valor, a regra de template encontrará na sub-árvore do elemento conceptual_model o nome dos campos que compõe a chave da tabela de classe no banco de dados que serão incluídos na tabela inst.

5 Conclusões

Esta dissertação apresentou a linguagem de marcação OOHDM-ML, criada para permitir a especificação declarativa de projetos OOHDM de aplicações hipermídia. Esta linguagem consiste de uma DTD que contém as regras para a especificação do projeto OOHDM da aplicação, e as regras para a especificação do mapeamento desse projeto para o ambiente de implementação OOHDM-Web.

Foi apresentado também o ambiente OOHDM-XWeb, criado para apoiar a implementação dessas aplicações. Este ambiente é formado pela folha de estilo OOHDM-Translation, responsável pela geração automática do projeto OOHDM-Web da aplicação, e pelo sub-ambiente OOHDM-Web 2.0, cujas funções são utilizadas na criação das páginas template que compõem a aplicação e na criação e manutenção das tabelas do banco de dados utilizado por ela.

Neste capítulo, apresentamos inicialmente as principais contribuições desta dissertação. Em seguida, apresentamos um resumo sobre alguns trabalhos relacionados e, logo após, algumas sugestões para trabalho futuro.

5.1 Contribuições

As principais contribuições desse trabalho são:

- Especificação declarativa de uma aplicação hipermídia independente do ambiente de implementação utilizado para construí-la. A DTD da linguagem OOHDM-ML é composta por dois elementos principais: o elemento *OOHDM_BASE* e o elemento *OOHDM_WEB*, sendo este último opcional. Desta forma, é possível especificar apenas o projeto OOHDM de uma aplicação hipermídia de forma completamente independente do ambiente que será usado para a implementação da aplicação. Isto permite também que outros ambientes possam ser utilizados para a implementação da aplicação a partir de sua especificação.
- Criação de um padrão para especificar o projeto de aplicações hipermídia utilizando a linguagem OOHDM-ML, baseada em XML. A especificação do projeto é feita de acordo com um conjunto de regras definido na DTD da linguagem, que foi estabelecido de acordo com as primitivas do método OOHDM.
- Criação da versão 2.0 do ambiente de desenvolvimento OOHDM-Web. Nesta nova versão foi realizada uma melhoria significativa na criação da chave primária das tabelas no banco de dados. A versão anterior tratava a chave primária de uma tabela como um identificador único e restringia o seu nome para "*prim_key*". A versão 2.0 permite que sejam definidas chaves compostas, ou seja, permite que a chave primária de uma tabela no banco de dados seja composta por mais de um atributo e não faz restrições quanto aos nomes utilizados para os campos que compõem a chave. Em virtude dessa melhoria, o *projeto de navegação* que contém a descrição do projeto OOHDM em tabelas da linguagem Lua e é utilizado pelas funções da biblioteca do ambiente OOHDM-Web, foi reestruturado e recebeu o nome de *projeto OOHDM-Web*. Foi criada uma nova estrutura Lua chamada *map_classes_ctx* para representar o mapeamento das classes em contexto para as tabelas no banco de dados.
- Geração automática da descrição do projeto OOHDM de uma aplicação em tabelas Lua para o ambiente OOHDM-Web 2.0. Esta geração é realizada pela folha de estilo

OOHDM-Translation, desenvolvida utilizando-se a linguagem XSLT. Esta folha de estilo contém as regras de transformação que um processador XSLT aplica à árvore de elementos e atributos de um documento OOHDM-ML para gerar um arquivo lua, contendo a descrição das tabelas que representam o projeto da aplicação. Com a OOHDM-Translation, o projetista gera automaticamente a representação em Lua do projeto, informando apenas o nome do documento XML que contém a especificação do projeto, eliminando grande parte dos erros que ocorrem durante a execução da aplicação na web.

5.2 Trabalhos Relacionados

A seguir são descritos alguns trabalhos relacionados a essa dissertação que integram a especificação de uma aplicação web e sua implementação. Alguns desses trabalhos são descritos com base nas informações contidas em [Fraternali 1999], que são resultado de um estudo realizado sobre ferramentas e abordagens para o desenvolvimento de aplicações web.

5.2.1 HDM-Edit

O HDM-Edit [Garzotto 2000] é um editor de esquemas que permite aos projetistas especificar, com base no modelo HDM2000 [Baresi 2000], todas as diferentes características do projeto de uma aplicação web. Ele promove o reuso de projeto por suportar vários "patterns" de navegação como primitivas de modelagem, e por permitir que os projetistas derivem o esquema de uma aplicação específica como um conjunto de hiper-visões sobre o projeto de uma família de aplicações ("application framework").

O HDM-Edit é um componente de um ambiente maior chamado JWeb [Bochicchio 1999], que suporta todas as fases do desenvolvimento web. O HDM-Edit pode ser usado sozinho para suportar apenas as atividades de projeto e a produção de documentação, ou em conjunto com outros módulos do JWeb.

Embora o formato interno do HDM-Edit para armazenar o projeto de uma aplicação seja proprietário, por razões de eficiência, as especificações de um projeto podem ser exportadas para outras ferramentas usando um arquivo XML. Para isto uma DTD do modelo HDM2000 foi definida.

O HDM-Edit é facilmente integrável a qualquer outro ambiente de desenvolvimento, uma vez que sua comunicação com outras ferramentas é garantida por um protocolo XML usando uma DTD pública.

De forma semelhante a DTD da linguagem de especificação OOHDM-ML que apresentamos nesta dissertação, a DTD do modelo HDM2000 permite descrever um esquema HDM2000 de uma aplicação web em um documento XML. Já a utilização do documento XML criado difere da que utilizamos nesta dissertação. No ambiente OOHDM-XWeb, o documento XML criado é utilizado para a geração do projeto OOHDM-Web que será usado posteriormente pelo ambiente OOHDM-Web 2.0. Neste trabalho relacionado, o documento XML pode ser importado pelo editor HDM-Edit e processado pelos componentes do ambiente JWeb durante a implementação e execução da aplicação.

5.2.2 WebML

WebML (Web Modeling Language) [Ceri 2000] é uma notação para especificar web sites complexos a nível conceitual. A especificação de um site em WebML consiste em descrevê-lo sob múltiplas dimensões:

- seu conteúdo de dados (modelo estrutural),
- as páginas que o compõem (modelo de composição),
- a topologia de links entre páginas (modelo de navegação),
- o layout e os requisitos gráficos para a interface da página (modelo de apresentação),
- e as características de customização para prover conteúdo personalizado (modelo de personalização).

Todos os conceitos de WebML estão associados a uma notação gráfica intuitiva e a uma sintaxe XML textual.

As especificações WebML são independentes da linguagem e da plataforma usadas para desenvolver a aplicação. Mas, elas podem ser usadas efetivamente para produzir a implementação do site em uma arquitetura específica.

Atualmente, a linguagem WebML e seu método de projeto são implementados em um conjunto de ferramentas de projeto web, chamado WebML Tool Suite. O WebML Tool Suite suporta a geração completamente automática de sites a partir de modelos WebML, usando as linguagens de marcação HTML e WML, e as tecnologias ASP e JSP.

Embora a WebML também seja uma linguagem de especificação para projetos de aplicações web baseada em XML, ela difere da linguagem OOHDM-ML apresentada nesta dissertação por abordar aspectos da aplicação mais próximos a implementação, como por exemplo a organização e a composição das páginas que compõem a aplicação. Seu modelo estrutural expressa os dados em termos de entidades e relacionamentos, usando o modelo E/R. Não existe um projeto de navegação separado das páginas, pois seu modelo de navegação expressa apenas como as páginas e o conteúdo do site são ligados para formar o hipertexto.

5.2.3 NCL

NCL (Nested Context Language) [Antonacci 2000] é uma linguagem declarativa para especificação de documentos hipermídia, baseada no modelo conceitual NCM (Nested Context Model) [Soares 2000] é especificada utilizando-se o padrão XML. O NCM é um modelo conceitual hipermídia baseado no conceito de nós, representando os componentes de um documento hipermídia, e elos, representando os relacionamentos entre esses nós.

Através da linguagem NCL é possível especificar documentos hipermídia genéricos contendo as características fundamentais que todo modelo hipermídia contempla, incluindo relacionamentos de sincronização temporal e espacial entre seus componentes.

Um objeto NCM em uma representação NCL, ou seja, um documento NCL pode ser convertido para um objeto NCM em uma representação Java, e vice-versa. Esta conversão é feita de forma automática por conversores NCL no sistema HyperProp [RMS 1998]. HyperProp é um sistema para tratamento de documentos hipermídia baseado no modelo NCM. Esse sistema permite a autoria gráfica de documentos com suporte a controle de versões, assim como a apresentação desses documentos em uma máquina também implementada em Java.

De forma semelhante a linguagem declarativa OOHDM-ML apresentada nesta dissertação, a linguagem NCL possui uma DTD que permite especificar documentos hipermídia baseados no modelo NCM. No entanto, a linguagem NCL contempla aspectos relacionados a sistemas multimídia, como sincronização temporal e espacial, que não são abordados na linguagem OOHDM-ML, uma vez que essa abordagem não faz parte do método OOHDM.

Este trabalho relacionado também apresenta uma forma de converter o documento XML gerado, o documento NCL, em um formato que pode ser entendido pelo sistema HyperProp, como é feito no ambiente OOHDM-XWeb pela folha de estilo OOHDM-Translation. No entanto, esta conversão difere da transformação realizada pelo OOHDM-Translation, uma vez que os conversores importam ou exportam um documento NCL a partir do sistema HyperProp e não utilizam uma linguagem de transformação específica como a que utilizamos neste trabalho.

5.2.4 Araneus

Araneus [Atzeni 1997][Atzeni 1998][Atzeni 1998] é um projeto que define o protótipo e a implementação de um ambiente, chamado Web Base Management System (WBMS), para gerenciar conteúdo Web estruturado e não estruturado de uma maneira integrada.

Na fase de modelagem, o Araneus enfatiza a distinção entre estrutura de dados, navegação e apresentação. Na modelagem da estrutura uma distinção adicional é feita entre banco de dados e estrutura hipertexto: o primeiro é especificado usando o Modelo Entidade-Relacionamento e o último usando uma notação que integra a especificação de estrutura e navegação, chamado Navigational Conceptual Model (NCM).

A modelagem conceitual é seguida por um projeto lógico, usando o modelo relacional para a parte estrutural e o Araneus Data Model (ADM) para a navegação e composição de páginas.

O processo de desenvolvimento segue duas fases: banco de dados e hipertexto. O projeto de banco de dados e a implementação são conduzidos usando o Modelo Entidade-Relacionamento e o seu mapeamento dentro de estruturas relacionais. Posteriormente, o esquema Entidade-Relacionamento é transformado em um esquema NCM. Este passo requer várias atividades de projeto. O próximo passo, o projeto lógico de hipertexto, mapeia o esquema NCM dentro de vários esquemas de páginas escritos em ADM.

Finalmente, a implementação consiste em escrever os esquemas de páginas como templates usando a linguagem Penelope [Atzeni 1997], que especifica como as páginas físicas são construídas a partir dos esquemas de páginas e do conteúdo armazenado no banco de dados, de forma similar aos integradores HTML-SQL comerciais baseados em templates.

Araneus inclui várias ferramentas para suportar a automação das tarefas de design mencionadas acima . Um ponto específico para o projeto Araneus é a integração, dentro do ambiente WBMS, de linguagens e ferramentas para buscar e reestruturar dados HTML de forma que o projetista possa disponibilizar um novo site e fazer a "engenharia–reversa" de um site HTML legado, em um único sistema.

O projeto Araneus não utiliza uma linguagem de especificação para o projeto de aplicações hipermídia como a que apresentamos nesta dissertação. Como vimos, a descrição de uma aplicação web é realizada usando um modelo de dados formal.

5.2.5 AutoWeb

AutoWeb [Fraternali 1998] é um projeto cujo objetivo é aplicar um processo de desenvolvimento orientado por modelos à construção e manutenção de aplicações que lidam

com grande volume de dados. O AutoWeb consiste de três ingredientes: (1) um modelo conceitual chamado HDM-lite, (2) algumas transformações automáticas que mapeiam a descrição feita no modelo conceitual em estruturas de uma base de dados relacional, e a produção de páginas em HTML e Java a partir dos dados armazenados na base de dados, (3) um conjunto de ferramentas CASE que automatizam o projeto, a implementação e a manutenção de uma aplicação web [Moura1999].

O modelo conceitual AutoWeb, chamado HDM-lite, inclui primitivas para a especificação independente da estrutura, navegação e apresentação da aplicação.

O principal foco do AutoWeb é a automação do processo de desenvolvimento. Sendo assim, uma aplicação AutoWeb é construída com o auxílio de várias ferramentas visuais que permitem:

- especificar de forma amigável o modelo conceitual e a apresentação da aplicação,
- traduzir o modelo conceitual para um esquema de banco de dados relacional e alimentar a base de dados automaticamente,
- e construir o conjunto de páginas da aplicação dinamicamente.

O projeto AutoWeb não utiliza uma linguagem de especificação para o projeto de aplicações hipermídia como a que apresentamos nesta dissertação. A especificação da aplicação é realizada utilizando o modelo conceitual HDM-Lite.

5.2.6 Strudel

Strudel [Fernandez 1998] é um projeto que visa experimentar uma nova forma de desenvolver sites Web baseada na especificação declarativa da estrutura e do conteúdo do site.

A idéia principal do Strudel é descrever tanto o esquema como o conteúdo de um site por meio de um conjunto de consultas sobre o modelo de dados semi-estruturado. Assim, o projeto de um site Web requer escrever uma ou mais consultas sobre a representação interna dos dados, usando a linguagem de consulta Strudel (StruQL). Tais consultas permitem que o projetista selecione os dados que serão incluídos no site e os links e coleções de objetos que serão fornecidos para navegação. Desta forma, o Strudel separa a descrição do conteúdo da definição da estrutura e navegação do site. A apresentação é adicionada como uma dimensão separada por meio de templates HTML.

A definição declarativa da estrutura e do conteúdo por meio de consultas abre um caminho para a personalização: diferentes sites ou versões diferentes do mesmo site podem ser construídas em cima do mesmo conteúdo, simplesmente alterando as consultas StruQL de definição do site.

Este trabalho também não utiliza uma linguagem declarativa para especificar o site, como a que apresentamos nesta dissertação. A especificação do site neste trabalho é realizada através da linguagem de *consulta* StruQL, definindo-se uma ou mais consultas sobre a representação interna dos dados.

5.3 Trabalhos Futuros

- **Conversão da DTD OOHDM-ML para XML Schema**

A linguagem de marcação OOHDM-ML foi criada definindo-se uma DTD, que contém o conjunto de regras que devem ser obedecidas para criação de documentos OOHDM-ML válidos. No entanto, DTDs possuem algumas limitações. Elas são escritas em uma sintaxe

diferente (não-XML), não oferecem suporte para namespaces e podem apenas expressar os tipos de dados de atributos em termos de enumeração explícita, e uns poucos formatos de string. DTDs são usadas para especificar a estrutura de um arquivo XML, elas não possuem um mecanismo para definir o conteúdo dos elementos em termos de tipos de dados. Portanto, DTDs não podem ser usadas para descrever números, datas, valores de moeda, etc., ou para definir restrições ou verificações sobre o conteúdo dos elementos, apenas sobre a marcação.

Sendo assim, uma sugestão para trabalho futuro seria fazer a conversão da DTD OOHDM-ML para XML Schema [W3C 2001]. XML Schema é uma linguagem para descrever o conteúdo e a estrutura de documentos XML em XML. Esta linguagem reproduz todas as capacidades da DTD e vence todas as limitações mencionadas, permitindo que tipos de restrições adicionais possam ser especificadas. XML Schemas oferecem tipos de dados mais ricos, como booleanos, números e datas, tipos definidos pelo usuário, agrupamento de atributos e suporte a namespace. XML Schema é mais flexível em especificar o número de instâncias de elementos filhos permitidos. Ela permite especificar o número máximo e mínimo de ocorrências, que pode ser um número não-negativo, não apenas zero, um ou infinito como em DTDs. XML Schema introduz também novos mecanismos de extensibilidade, mecanismos de refinamento, onde os elementos podem herdar as restrições de outros elementos e estender ou modificar essas restrições para propósitos particulares, mecanismos de import e export e mecanismos mais flexíveis e poderosos para garantir a unicidade de qualquer elemento ou tipo de atributo, independentemente de seu tipo.

O trabalho de conversão para XML Schema poderá ser realizado com o auxílio de ferramentas XML já existentes, desde que sejam compatíveis com a versão final da especificação XML Schema. Durante o processo de conversão será necessário analisar as possíveis melhorias que poderão ser realizadas na linguagem OOHDM-ML, considerando todas as vantagens oferecidas por esta recomendação.

- **Análise do padrão XMI (XML Metadata Interchange)**

Uma outra sugestão para trabalho futuro seria realizar uma análise do padrão XMI (XML Metadata Interchange [OMG 2000] para identificar uma possível relação entre ele e a linguagem OOHDM-ML. XMI é um padrão baseado no formato XML para a troca de modelos definidos usando a linguagem UML.

- **Definição do conteúdo para o elemento interface_model**

Um outro trabalho futuro é a definição do conteúdo para o elemento *interface_model* da linguagem OOHDM-ML. Este elemento será usado para especificar o projeto da Interface Abstrata de aplicações hipermídia, permitindo a especificação declarativa da última etapa do projeto OOHDM.

- **Extensão do conteúdo dos elementos conceptual_class e navigational_class**

Um outro trabalho futuro é a definição de duas novas marcações na DTD OOHDM-ML, uma para permitir a especificação de atributos derivados (atributos computados a partir de outros atributos) e outra para permitir a especificação de atributos do tipo lista. A marcação para a especificação de atributos derivados deve fazer parte do conteúdo dos elementos *conceptual_class* e *navigational_class*. Já a marcação para a especificação de atributos do tipo lista deve fazer parte apenas do conteúdo do elemento *navigational_class*.

- **Criação de um ambiente integrado para apoiar o processo de autoria e implementação**

Em relação ao ambiente OOHDM-XWeb, uma sugestão de trabalho futuro é a implementação de uma interface para o processamento da folha de estilo OOHDM-Translation. Esta interface permitirá que o projetista processe a folha de estilo informando apenas o nome do documento OOHDM-ML a ser transformado, sem que ele tenha a necessidade de passar todos os parâmetros necessários para a execução do processador XSLT.

Um outro trabalho futuro seria criar um outro módulo para o ambiente OOHDM-XWeb que permitisse gerar páginas template padrão a partir da especificação da aplicação no documento OOHDM-ML. Este novo módulo poderia ser uma folha de estilo XSLT, que geraria as páginas html da aplicação com base na especificação OOHDM-ML. Estas páginas seriam geradas com um layout básico já contendo chamadas às funções do ambiente OOHDM-Web para apresentação do valor de atributos, definição de índices e navegação entre elementos de um contexto. A geração dessas páginas poderia considerar a passagem de parâmetros cgi, criando as variáveis de cgi automaticamente a partir das definições contidas na especificação OOHDM-ML da aplicação.

Um outro trabalho futuro seria a criação de um ambiente de autoria para projetos de aplicações hipermídia, incluindo a atividade de Levantamento de Requisitos, para tornar mais amigável o processo de autoria da aplicação. Este ambiente seria capaz de importar um documento OOHDM-ML contendo a especificação do projeto e, a partir dele gerar a representação gráfica dos esquemas conceitual, navegacional e de contextos. Este ambiente poderia também exportar um documento OOHDM-ML, criado a partir das definições do projeto.

Um outro trabalho futuro interessante seria criar um ambiente integrado para apoiar todas as etapas do processo de desenvolvimento de uma aplicação hipermídia. Este ambiente seria composto por duas ferramentas principais, uma para apoiar a atividade de autoria da aplicação e outra para apoiar a implementação da aplicação. Com a ferramenta de autoria, o projetista poderia entrar com as definições das classes conceituais, seus atributos e relacionamentos e obter a representação gráfica do esquema conceitual da aplicação. Da mesma forma ele poderia entrar com a definição dos nós e elos e obter o esquema navegacional da aplicação. Os contextos e estruturas de acesso seriam especificados em espécies de cartões de especificação através de uma interface amigável e a representação do esquema de contextos também seria gerada automaticamente. A ferramenta de autoria seria capaz de gerar automaticamente, a partir dessas definições, a especificação de todo o projeto OOHDM em documentos XML, usando a linguagem OOHDM-ML. Neste ambiente integrado, após definir o projeto OOHDM da aplicação, o projetista teria a opção de escolher em que ambiente ele deseja implementá-la, por exemplo ele poderia optar em usar o ambiente OOHDM-Web ou o ambiente OOHDM-Java [Pizzol 1998]. Estes ambientes seriam capazes de consultar os documentos XML gerados e utilizar a especificação definida para construir a aplicação.

- **Estudo sobre a utilização de outras linguagens de transformação**

Nesta dissertação utilizamos a linguagem XSLT para implementar a folha de estilo OOHDM-Translation. A linguagem XSLT foi adotada pelo fato de estarmos trabalhando com a transformação de um documento XML. No entanto, a linguagem XSLT possui algumas limitações. Uma dessas limitações está relacionada à capacidade de formatação da saída. Em XSLT a formatação da saída segue a formatação utilizada no código fonte da folha de estilo. Muitas vezes, essa limitação faz com que o desenvolvedor tenha que decidir entre sacrificar a formatação do documento resultado da transformação ou a identação do código fonte na folha de estilo. Este problema certamente será resolvido quando a linguagem XSL tornar-se uma recomendação do W3C. Uma outra limitação está relacionada à funcionalidade de algumas instruções, como por exemplo o tratamento de variáveis como constantes e a ausência de comandos de repetição. Essa limitação faz com o desenvolvedor use funções de

extensão de um processador XSLT específico, limitando a execução da folha de estilo a esse processador. Sendo assim, um outro trabalho futuro interessante é realizar um estudo sobre a possibilidade de utilizar outras linguagens de transformação para gerar o projeto OOHDM-Web a partir de um documento OOHDM-ML.

- **Análise e especificação de mapeamentos para outros ambientes de implementação**

Nesta dissertação definimos um conjunto de marcações (elementos e atributos) para permitir a especificação do mapeamento do projeto OOHDM da aplicação para o ambiente de implementação OOHDM-WEB 2.0. Um outro trabalho futuro é definir outros conjuntos de marcações para permitir a especificação de mapeamentos do projeto OOHDM para outros ambientes de implementação, como por exemplo o ambiente OOHDM-Java.

Apêndice I

Listagem da DTD OOHDM-ML

```
<!-- ============================================================ -->
<!-- =               OOHDM Document Type Definition             = -->
<!-- =  Description: XML Document Type Definition (DTD) for the = -->
<!-- =               Object-Oriented Hypermedia Design Method   = -->
<!-- =  Department of Computer Science, PUC-Rio                 = -->
<!-- =  Authors: Daniel Schwabe, Adriana Pereira de Medeiros    = -->
<!-- =           and Mark Douglas de Azevedo Jacyntho           = -->
<!-- =  Version: 1.2                                            = -->
<!-- =  Date: January/2001                                      = -->
<!-- ============================================================ -->

<!-- ======================== General entities ================== -->

<!-- global attributes -->

<!ENTITY % global_class_attrib  "
     id            ID     #REQUIRED
     name          CDATA  #REQUIRED
     property      CDATA  #IMPLIED
     superclass    IDREFS #IMPLIED
     discriminator CDATA  #IMPLIED
     type          (concrete|abstract) 'concrete'
     subclass_restriction (disjunct_complete|disjunct_incomplete|
                           superpose_complete|superpose_incomplete)
                           'disjunct_incomplete'">

<!ENTITY % global_attrib  "
     name          CDATA #REQUIRED
     default_value CDATA #IMPLIED
     property      CDATA #IMPLIED
     derived       (yes|no) 'no'
     visibility    (public|private) 'public'" >

<!ENTITY % global_relationship_attrib  "
     id                 ID    #REQUIRED
     name               CDATA #REQUIRED
     source_class       IDREF #REQUIRED
     target_class       IDREF #REQUIRED
     source_cardinality CDATA #REQUIRED
     target_cardinality CDATA #REQUIRED
     source_role        CDATA #IMPLIED
     target_role        CDATA #IMPLIED
     type               (association|aggregate|composition)
                        'association'" >

<!ENTITY % global_ref_nav  "
     name               CDATA #REQUIRED
     navigational_class IDREF #REQUIRED ">
<!ENTITY % global_type  "
     type (string|text|integer|real|
           boolean|date|image|audio|video) #REQUIRED ">

<!-- ====================== OOHDM Element ====================== -->

<!ELEMENT OOHDM (OOHDM_BASE,OOHDM_WEB?)>
```

```
<!ATTLIST OOHDM
     name     CDATA #REQUIRED
     version  CDATA #REQUIRED >

<!-- ================== OOHDM_BASE Element ====================== -->

<!ELEMENT OOHDM_BASE (conceptual_model,navigational_model,
                      interface_model)>

<!--================== conceptual_model Element ================ -->

<!ELEMENT conceptual_model ((subsystem+,conceptual_class*,
                             relationship*)|
                            (conceptual_class+,relationship*))>

<!-- ==================== subsystem Element ===================== -->

<!ELEMENT subsystem (conceptual_class+, relationship*)>
<!ATTLIST subsystem
     name           ID     #REQUIRED
     imported_class IDREFS #IMPLIED >

<!-- ================== conceptual_class Element ================ -->

<!ELEMENT conceptual_class (attrib*,operation*)>
<!ATTLIST conceptual_class
     %global_class_attrib; >

<!-- ====================== attrib Element ======================= -->

<!ELEMENT attrib (perspective*)>
<!ATTLIST attrib
     %global_attrib;
     %global_type;
     size                  CDATA #IMPLIED
     multi_occurrence (yes|no) "no">

<!-- ====================== perspective Element ================== -->

<!ELEMENT perspective EMPTY>
<!ATTLIST perspective
     name          CDATA #REQUIRED
     default_value CDATA #IMPLIED
     size          CDATA #IMPLIED
     default (yes|no) "no"
     %global_type; >

<!-- ====================== operation Element ==================== -->

<!ELEMENT operation (parameter*,return_type?)>
<!ATTLIST operation
     name          CDATA #REQUIRED
     property      CDATA #IMPLIED
     visibility    (public|private) "public">

<!-- ====================== parameter Element ==================== -->

<!ELEMENT parameter (primitive_type|defined_type)>
<!ATTLIST parameter
     name          CDATA #REQUIRED
     default_value CDATA #IMPLIED >

<!-- ==================== return_type Element ==================== -->

<!ELEMENT return_type (primitive_type|defined_type)>

<!-- ================ primitive_type Element ==================== -->

<!ELEMENT primitive_type EMPTY>
```

```
<!ATTLIST primitive_type
     %global_type; >

<!-- ================== defined_type Element ================== -->

<!ELEMENT defined_type EMPTY>
<!ATTLIST defined_type
     class   IDREF #REQUIRED >

<!-- ================== relationship Element ================== -->

<!ELEMENT relationship (conceptual_class?)>
<!ATTLIST relationship
     %global_relationship_attrib;>

<!-- ============== navigational_model Element ================= -->

<!ELEMENT navigational_model (navigational_class+,instance*,link*,
                              navigational_context+,context_class*,
                              (index|hierarch_index)+,landmark*)>

<!-- ================ navigational_class Element ============== -->

<!ELEMENT navigational_class (nav_attrib|nav_attrib_index|
                              perspective_attrib|nav_operation|
                              anchor)*>
<!ATTLIST navigational_class
     %global_class_attrib; >

<!-- ================== nav_attrib Element ====================== -->

<!ELEMENT nav_attrib EMPTY>
<!ATTLIST nav_attrib
     name             CDATA #REQUIRED
     conceptual_class IDREF #REQUIRED >

<!-- =============== nav_attrib_index Element ================== -->

<!ELEMENT nav_attrib_index EMPTY>
<!ATTLIST nav_attrib_index
     name  CDATA #REQUIRED
     index IDREF #REQUIRED >

<!-- ================ perspective_attrib Element ============== -->

<!ELEMENT perspective_attrib EMPTY>
<!ATTLIST perspective_attrib
     %global_attrib;
      optional         (yes|no) "no"
      perspective       CDATA  #REQUIRED
      conceptual_attrib CDATA  #REQUIRED
      conceptual_class  IDREF  #REQUIRED >

<!-- ==================== nav_operation Element ================ -->

<!ELEMENT nav_operation EMPTY>
<!ATTLIST nav_operation
     name             CDATA #REQUIRED
     conceptual_class IDREF #REQUIRED >

<!-- ===================== anchor Element ====================== -->

<!ELEMENT anchor ((name,destination?)|(shown_attrib,destination))>
<!ATTLIST anchor
     type  (next|previous|first|last) #IMPLIED >

<!-- ======================= name Element ====================== -->

<!ELEMENT name (#PCDATA)>
```

```
<!-- ================== shown_attrib Element ==================== -->

<!ELEMENT shown_attrib EMPTY>
<!ATTLIST shown_attrib
     %global_ref_nav;>

<!-- ==================== destination Element ================== -->

<!ELEMENT destination (#PCDATA)>
<!ATTLIST destination
     instance  IDREF #IMPLIED
     target    IDREF #IMPLIED >

<!-- ==================== instance Element ====================== -->

<!ELEMENT instance (instance_attrib*)>
<!ATTLIST instance
     name               ID    #REQUIRED
     navigational_class IDREF #REQUIRED >

<!-- ==================== instance_attrib Element =============== -->

<!ELEMENT instance_attrib (#PCDATA)>
<!ATTLIST instance_attrib
     name    CDATA  #REQUIRED >

<!-- ======================== link Element ====================== -->

<!ELEMENT link (comment?)>
<!ATTLIST link
     %global_relationship_attrib;
     persistent_source   (yes|no) "no"
     dependence          CDATA #IMPLIED
     influence           CDATA #IMPLIED >

<!-- ======================== comment Element =================== -->

<!ELEMENT comment (#PCDATA)>

<!-- ============= navigational_context Element ================ -->

<!ELEMENT navigational_context (instance*,selection*,order*,
                                restriction_use*,operation*,
                                comment?)>
<!ATTLIST navigational_context
     id              ID     #REQUIRED
     name            CDATA  #REQUIRED
     element_class   IDREFS #REQUIRED
     type            (static|dynamic|by_query|temporary) #REQUIRED
     navigation_type (sequential|circular|index|free|
                      sequential_index|circular_index) #REQUIRED
     polymorphic (yes|no) "no"
     persistent   (yes|no) "no"
     context_classes IDREFS  #IMPLIED>

<!-- ================ selection Element ======================== -->

<!ELEMENT selection ((AND|OR)|(greater|greater_equal|lesser|
                               lesser_equal|equal|different))>
<!ATTLIST selection
     selected_element_class IDREF #IMPLIED >

<!-- ======================== AND Element ====================== -->

<!ELEMENT AND (((AND|OR),(greater|greater_equal|lesser|
                          lesser_equal|equal|different|AND|OR)) |
               ((greater|greater_equal|lesser|
                 lesser_equal|equal|different),
```

```
                (greater|greater_equal|lesser|
                 lesser_equal|equal|different)))>

<!-- ======================= OR Element ========================= -->

<!ELEMENT OR (((AND|OR),(greater|greater_equal|lesser|
                 lesser_equal|equal|different|AND|OR)) |
               ((greater|greater_equal|lesser|
                 lesser_equal|equal|different),
                (greater|greater_equal|lesser|
                 lesser_equal|equal|different)))>

<!-- ======================= greater Element ==================== -->

<!ELEMENT greater ((attribute|plus|minus|mult|div),
                   (attribute|value|nav_parameter|plus|minus|
                    mult|div))>

<!-- ================ greater_equal Element ===================== -->

<!ELEMENT greater_equal ((attribute|plus|minus|mult|div),
                         (attribute|value|nav_parameter|plus|minus|
                          mult|div))>

<!-- ==================== lesser Element ======================== -->

<!ELEMENT lesser ((attribute|plus|minus|mult|div),
                  (attribute|value|nav_parameter|plus|minus|
                    mult|div))>

<!-- ================= lesser_equal Element ===================== -->

<!ELEMENT lesser_equal ((attribute|plus|minus|mult|div),
                        (attribute|value|nav_parameter|plus|minus|
                         mult|div))>

<!-- ======================= equal Element ====================== -->

<!ELEMENT equal ((attribute|plus|minus|mult|div),
                 (attribute|value|nav_parameter|plus|minus|
                  mult|div))>

<!-- ================== different Element ======================= -->

<!ELEMENT different ((attribute|plus|minus|mult|div),
                     (attribute|value|nav_parameter|plus|minus|
                      mult|div))>

<!-- ======================= plus Element ======================= -->

<!ELEMENT plus (attribute,(attribute|value|nav_parameter))>

<!-- ====================== minus Element ======================= -->

<!ELEMENT minus (attribute,(attribute|value|nav_parameter))>

<!-- ====================== mult Element ======================== -->

<!ELEMENT mult (attribute,(attribute|value|nav_parameter))>

<!-- ====================== div Element ========================= -->

<!ELEMENT div (attribute,(attribute|value|nav_parameter))>

<!-- ================== attribute Element ======================= -->

<!ELEMENT attribute EMPTY>
<!ATTLIST attribute
     link   IDREF           #IMPLIED
     role   (source|target) #IMPLIED
     %global_ref_nav;>
```

```
<!-- ======================= value Element ======================= -->

<!ELEMENT value (#PCDATA)>

<!-- ================== nav_parameter Element ================== -->

<!ELEMENT nav_parameter (#PCDATA)>

<!-- ========================= order Element ==================== -->

<!ELEMENT order (order_element+)>
<!ATTLIST order
      id      ID  #REQUIRED
      default (yes|no) "no" >

<!-- ================== order_element Element ================== -->

<!ELEMENT order_element EMPTY>
<!ATTLIST order_element
     %global_ref_nav;
     criteria (asc|desc) "asc" >

<!-- ================ restriction_use Element ================== -->

<!ELEMENT restriction_use (user+)>
<!ATTLIST restriction_use
     permission CDATA  #REQUIRED>

<!-- ================== user Element ============================-->

<!ELEMENT user (#PCDATA)>

<!-- ================== context_class Element ================== -->

<!ELEMENT context_class (nav_attrib|perspective_attrib|anchor|
                         operation)*>
<!ATTLIST context_class
     id         ID     #REQUIRED
     name       CDATA  #IMPLIED
     base_class IDREF  #REQUIRED
     contexts   IDREFS #REQUIRED >

<!-- ================ hierarch_index Element ================== -->

<!ELEMENT hierarch_index ((selection*|element_context),level+,
                           order*,restriction_use*,comment?)>
<!ATTLIST hierarch_index
     id             ID              #REQUIRED
     name           CDATA           #REQUIRED
     element_class  IDREFS          #IMPLIED
     dependence     CDATA           #IMPLIED
     influence      CDATA           #IMPLIED
     nested_index   IDREFS          #IMPLIED
     type           (static|dynamic|temporary) "static" >

<!-- ============== element_context Element ================== -->

<!ELEMENT element_context EMPTY>
<!ATTLIST element_context
     context   IDREF #REQUIRED>

<!-- ======================= level Element ====================== -->

<!ELEMENT level (shown_attrib*,(dynamic_selector|static_selector)+)>

<!-- ==================== index Element ======================== -->

<!ELEMENT index ((selection*|element_context),
                (shown_attrib*,(dynamic_selector|static_selector))+,
                 order*,restriction_use*,comment?)>
```

```
<!ATTLIST index
     id              ID               #REQUIRED
     name            CDATA            #REQUIRED
     element_class   IDREFS           #IMPLIED
     dependence      CDATA            #IMPLIED
     influence       CDATA            #IMPLIED
     nested_index    IDREFS           #IMPLIED
     type            (static|dynamic|temporary) "static" >

<!-- ================ dynamic_selector Element ================ -->

<!ELEMENT dynamic_selector (shown_attrib+,selector_destination+)>

<!-- ============== static_selector Element ================== -->

<!ELEMENT static_selector (name,selector_destination)>
<!ATTLIST static_selector
     id   ID  #REQUIRED>

<!-- ================ selector_destination Element ============ -->

<!ELEMENT selector_destination EMPTY>
<!ATTLIST selector_destination
     target     IDREF #REQUIRED
     order      IDREF #IMPLIED
     instance   IDREF #IMPLIED >

<!-- ================== landmark Element ====================== -->

<!ELEMENT landmark (#PCDATA)>
<!ATTLIST landmark
target IDREF  #REQUIRED >

<!-- ================= interface_model Element ================ -->

<!ELEMENT interface_model EMPTY>

<!-- ==================== OOHDM_WEB Element =================== -->

<!ENTITY % oohdm_web SYSTEM "OOHDM_WEB.dtd">
%oohdm_web;
```

Listagem da DTD OOHDM_WEB

```
<!-- ============================================================ -->
<!-- =            OOHDM-Web Document Type Definition            = -->
<!-- =  Description: XML Document Type Definition (DTD) for     = -->
<!-- =               the OOHDM-Web environment                   = -->
<!-- =  Department of Computer Science, PUC-Rio                  = -->
<!-- =  Authors: Adriana Pereira de Medeiros and Daniel Schwabe = -->
<!-- =  Version: 1.0                                             = -->
<!-- =  Date: November/2000                                      = -->
<!-- ============================================================ -->

<!-- ======================= General entities ================= -->

<!-- As entidades parametro convert_access e convert_oracle contem o
     criterio de conversao de um tipo de um atributo de uma classe
     conceitual, especificado pelo usuario, para o tipo valido em um
     banco de dados especifico (ACCESS ou ORACLE). O simbolo "_"
     (underscore) separando dois tipos representa que o primeiro tipo
     sera convertido para o segundo (no caso de convert_access o tipo
     string especificado para um atributo sera convertido para o tipo
     text do banco ACCESS). Ja o simbolo "-"(hifen) apenas separa as
     diversas conversoes. -->
<!ENTITY % convert_access
           "string_text-image_text-audio_text-video_text-
            boolean_yesno-text_memo-real_integer-">
<!ENTITY % convert_oracle
           "string_varchar-image_varchar-audio_varchart-
           video_varchar-text_varchar2-real_number-integer_number-">
<!-- Esta "conditional section" determina a enumeracao que sera
     considerada para o atributo db_type do elemento map_attribute,
     de acordo com o banco de dados considerado no documento XML.
     As entidades parametro convert_access e convert_oracle sao
     utilizadas apenas para que o usuario nao seja obrigado a mapear
     um atributo apenas para converte-lo para um tipo valido no banco
     de dados. Assim, quando nao for especificado valor para o
     atributo db_type o valor default (covert_access ou
     convert_oracle) sera usado para realizar a conversao
     automaticamente. -->
<![%MS_ACCESS;[
   <!ENTITY % type "db_type (text|memo|byte|short|integer|single|
                             number|date|yesno|%convert_access;)
                             '%convert_access;' ">
]]>
<![%ORACLE; [
   <!ENTITY % type "db_type (char|varchar|varchar2|number|date|
                             boolean|%convert_oracle;)
                             '%convert_oracle;' ">
]]>

<!-- ==================== OOHDM_WEB Element ================== -->

<!ELEMENT OOHDM_WEB (map_conceptual_class*, map_context_class*,
                     map_relationship*,interface_element+)>

<!-- =============== map_conceptual_class Element ============== -->

<!ELEMENT map_conceptual_class (map_attribute*)>
<!ATTLIST map_conceptual_class
```

```
     conceptual_class IDREF #REQUIRED
     db_class         CDATA #IMPLIED >

<!-- ================== map_attribute Element ================== -->

<!ELEMENT map_attribute EMPTY>
<!ATTLIST map_attribute
       id           ID          #REQUIRED
       name         CDATA       #REQUIRED
       db_attribute CDATA       #REQUIRED
       primary_key  (yes|no)    "no"
       %type;>

<!-- ================ map_context_class Element ================== -->

<!ELEMENT map_context_class (map_attribute*)>
<!ATTLIST map_context_class
     context_class  IDREF #REQUIRED
     db_class_ctx   CDATA #REQUIRED >

<!-- ================= map_relationship Element ================= -->

<!ELEMENT map_relationship (db_source_field*,db_destination_field*)>
<!ATTLIST map_relationship
     relationship    IDREF   #REQUIRED
     db_relationship CDATA   #REQUIRED>

<!-- ================ db_source_field Element =================== -->

<!ELEMENT db_source_field (#PCDATA)>
<!ATTLIST db_source_field
     map_attribute    IDREF   #IMPLIED>

<!-- ================ db_destination_field Element ============== -->

<!ELEMENT db_destination_field (#PCDATA)>
<!ATTLIST db_destination_field
     map_attribute    IDREF   #IMPLIED>

<!-- ================ interface_element Element ================ -->

<!ELEMENT interface_element (#PCDATA)>
<!ATTLIST interface_element
     target IDREFS #REQUIRED>
```

Apêndice II

Glossário

Como os nomes das tabelas Lua e de seus campos no projeto OOHDM-Web foram definidos em inglês, foi criado este glossário que relaciona cada nome a seu correspondente em português, com uma breve descrição sobre o nome relacionado.

Inglês	Português	Descrição
anchor	âncora	Âncora de um índice ou de um landmark. Aparece em nav_indexes.
attrib_class_ctx	atrib_classe_ctx	Tabela com o mapeamento dos atributos de classe em contexto para o BD. Aparece em queries.
attrib_size	tam_atrib	Tamanho do atributo. Aparece em con_classes.
attrib_type	tipo_atrib	Tipo do atributo. Aparece em con_classes.
attributes	Atributos	Em map_classes, é a tabela com o mapeamento dos atributos de classe para o BD. Em nav_indexes, é uma lista de atributos que são exibidos no índice.
Class	Classe	Nome de uma classe. Aparece em nav_indexes.
Class_ctx	classe_ctx	Tabela com atributos de classe em contexto. Aparece em queries.
condition	Condicao	Condição de seleção de elementos. Aparece em nav_contexts e nav_indexes.
context	Contexto	Em nav_indexes é o nome de um contexto. Em nav_contexts é o tipo de um contexto.
Con_classes	con_classes	Tabela que descreve as classes.
Con_relationships	con_relacionamentos	Tabela que descreve os relacionamentos.
db_class	nome_classe_bd	Nome da tabela de classe no BD. Aparece em map_classes.
db_class_ctx	nome_classe_ctx_bd	Nome da tabela de classe em contexto no BD. Aparece em queries.
db_destination_field	campo_destino	Tabela que especifica os nomes dos campos destino no BD. Aparece em map_relationships.
db_relationship	tab_relacao	Tabela do BD que contém um relacionamento. Aparece em map_relationships.
db_source_field	campo_origem	Tabela que especifica os nomes dos campos origem no BD. Aparece em map_relationships.
dest_card	card_destino	Cardinalidade da classe destino. Aparece em con_relationships.

Dest_page	pag_destino	Nome de uma página HTML, destino de um índice ou de um contexto. Aparece em nav_contexts e nav_indexes.
destination	destino	Nome do índice ou contexto destino. Aparece em nav_indexes.
destination_class	classe_destino	Nome da classe destino. Aparece em con_relationships.
elements	elementos	Tabela que especifica os elementos de um índice ou contexto. Aparece em nav_contexts e nav_indexes.
Inst	instancia	Atributo que identifica um elemento de um contexto. Aparece em nav_indexes.
Link_name	nome_elo	Tabela que especifica o relacionamentos em um contexto derivado de elo. Aparece em nav_contexts.
Map_classes	map_classes	Tabela de mapeamento das classes.
Map_relationships	map_relacoes	Tabela de mapeamento dos relacionamentos.
Nav_contexts	nav_contextos	Tabela que descreve os contextos.
Nav_indexes	nav_indices	Tabela que descreve os índices.
Nav_landmarks	nav_landmarks	tabela que descreve os landmarks.
Nav_type	tipo_nav	Tipo de navegação. Aparece em nav_contexts.
Order	ordem	Atributos que ordenam os elementos de um contexto ou índice. Aparece em nav_contexts e nav_indexes.
Parameters	parametros	Lista de parâmetros. Aparece em nav_contexts e nav_indexes.
Prim_key	chave	Indica que o atributo é um atributo chave. Aparece em con_classes.
Selection	selecao	Tabela que especifica a seleção de elementos de um contexto. Aparece em nav_contexts.
Selectors	seletores	Seletores de um índice. Aparece em nav_indexes.
source_card	card_origem	Cardinalidade da classe origem. Aparece em con_relationships.
source_class	classe_origem	Nome da classe origem. Aparece em con_relationships.
Type	tipo	Tipo de índice. Aparece em nav_indexes.

Apêndice III

Especificação da folha de estilo oohdm_translation.xsl

A folha de estilo *oohdm_translation.xsl* é responsável pela criação da estrutura de cada tabela Lua que compõem o projeto OOHDM-Web. Todas as tabelas Lua são criadas com base nas sub-árvores dos elementos *conceptual_model* e *navigational_model* do documento OOHDM-ML. A especificação desta folha de estilo é a seguinte:

Para o elemento **conceptual_model** do documento XML associado à folha de estilo faça

Crie a estrutura Lua con_classes incluindo o formato abaixo:

con_classes =

{

Processe recursivamente todos os elementos filhos *conceptual_class*.

Complete a estrutura con_classes utilizando o símbolo "}", alinhando–o com o símbolo "{" inicial.

Crie a estrutura Lua map_classes incluindo o formato abaixo:

map_classes =

{

Processe recursivamente todos os elementos filhos *conceptual_class* e todos os elementos *map_conceptual_class* relacionados a eles.

Complete a estrutura map_classes utilizando o símbolo "}", alinhando–o com o símbolo "{" inicial.

Se o elemento *conceptual_model* possui algum elemento filho *relationship*

então

Crie a estrutura Lua con_relationships incluindo o formato abaixo:

con_relationships =

{

Processe recursivamente todos os elementos filhos *relationship*.

Complete a estrutura con_relationships utilizando o símbolo }", alinhando-o com o símbolo "{" inicial.

Crie a estrutura Lua map_relationships incluindo o formato abaixo:

map_relationships =

{

Processe todos os elementos *relationship* e todos os elementos *map_relationship* relacionados a eles.

Complete a estrutura map_relationships utilizando o símbolo "}", alinhando–o com o símbolo "{" inicial.

fim-Se

fim-Para.

Para o elemento **navigational_model** do documento XML associado à folha de estilo faça

Crie a estrutura Lua queries incluindo o formato abaixo:

queries =

{

Processe recursivamente todos os elementos filhos *navigational_class*.

Complete a estrutura queries utilizando o símbolo "}", alinhando–o com o símbolo "{" inicial.

Se o elemento *navigational_model* possui algum elemento filho *context_class*

então

Crie a estrutura Lua map_classes_c*tx* incluindo o formato abaixo:

map_classes_ctx =

{

Processe recursivamente todos os elementos filhos *context_class* e todos os elementos *map_context_class* relacionados a eles.

Complete a estrutura map_classes_ctx utilizando o símbolo "}", alinhando–o com o símbolo "{" inicial.

fim-Se.

Crie a estrutura Lua nav_contexts incluindo o formato abaixo:

nav_contexts =

{

Processe recursivamente todos os elementos filhos *navigational_context.*

Complete a estrutura nav_contexts utilizando o símbolo "}", alinhando–o com o símbolo "{" inicial.

Crie a estrutura Lua nav_indexes incluindo o formato abaixo:

nav_indexes =

{

Processe recursivamente todos os elementos filhos *index*.

Processe recursivamente todos os elementos filhos **hierarch_index**.

Complete a estrutura *nav_indexes* utilizando o símbolo "}", alinhando–o com o símbolo "{" inicial.

fim-Para.

Inclua as regras da folha de estilo con_classes.xsl

Inclua as regras da folha de estilo map_classes.xsl

Inclua as regras da folha de estilo con_relationships.xsl

Inclua as regras da folha de estilo map_relationships.xsl

Inclua as regras da folha de estilo queries.xsl

Inclua as regras da folha de estilo map_classes_ctx.xsl

Inclua as regras da folha de estilo nav_contexts.xsl

Inclua as regras da folha de estilo nav_indexes.xsl

Especificação da folha de estilo con_classes.xsl

A folha de estilo *con_classes.xsl* é responsável pela descrição do conteúdo da tabela Lua con_classes, que representa as classes do modelo conceitual de uma aplicação hipermídia. Estas classes são especificadas no documento OOHDM-ML utilizando-se o elemento *conceptual_class*. A especificação desta folha de estilo é a seguinte:

Para todos os elementos *conceptual_class* encontrados no documento XML associado a folha de estilos faça

Inclua na estrutura Lua con_classes o valor do atributo name seguido dos caracteres "=" e "{" da seguinte forma:

<*valor_do_atributo_name*> =

{

Se existe um atributo superclass para o elemento *conceptual_class*

então

Chame Tratar_Atributo_Superclass

Se existe algum elemento *attrib* filho do elemento *conceptual_class*
então
Inclua uma vírgula.
fim-Se.
fim-Se.
Para cada elemento *attrib*, filho do elemento *conceptual_class* faça
Chame Tratar_Elementos_Attrib.
Se o elemento attrib que está sendo tratado é o último elemento filho
então
Inclua na estrutura Lua con_classes o símbolo "}" .
senão
Inclua na estrutura Lua con_classes o símbolo "}" seguido de vírgula.
fim-Se
fim-Para.
Inclua na estrutura Lua con_classes o símbolo "}".
Se o elemento *conceptual_class* não é o último elemento tratado
então
Inclua uma vírgula seguindo o símbolo "}" incluído anteriormente.
fim-Se
fim-Para.

Tratar_Atributo_Superclass

Para cada valor (id) informado no atributo superclass faça
Se o elemento *conceptual_class* identificado pelo valor em questão também possui um atributo superclass
então
Chame Tratar_Atributo_Superclass, recursivamente.
Se existe algum elemento *attrib* filho do elemento *conceptual_class* identificado pelo valor em questão
então
Inclua uma vírgula.
fim-Se.
fim-Se.
Para cada elemento *attrib*, filho do elemento *conceptual_class* identificado pelo valor informado em superclass faça
Chame Tratar_Elementos_Attrib.
Se o elemento attrib que está sendo tratado é o último elemento filho
então
Inclua na estrutura Lua con_classes o símbolo "} ".
senão
Inclua na estrutura Lua con_classes o símbolo } seguido de vírgula
fim-Se
fim-Para.
fim-Para.

Tratar_Elementos_Attrib

Se o elemento *attrib* possui algum elemento filho *perspective*
então

Para cada elemento *perspective* filho do elemento *attrib* faça

Inclua na estrutura Lua con_classes o valor do atributo name do elemento *perspective* seguido dos caracteres "=" e "{" da seguinte forma:

<valor_do_atributo_name> =

{

Chame Tratar_Atributos, informando os valores dos atributos type e size do elemento *perspective*, e o valor do atributo id do elemento *conceptual_class* que está sendo tratado.

Se o elemento perspective não é o último elemento tratado

então

Inclua na estrutura Lua con_classes o símbolo } seguido de vírgula

fim-Se

fim-Para

senão

Inclua na estrutura Lua con_classes o valor do atributo name do próprio elemento *attrib* seguido dos caracteres "=" e "{" da seguinte forma:

<valor_do_atributo_name> =

{

Chame Tratar_Atributos, informando os valores dos atributos type e size do elemento *attrib*, e o valor do atributo id do elemento *conceptual_class* que está sendo tratado.

fim-Se.

Tratar_Atributos

Se existe um elemento *map_attribute*, filho do elemento *map_conceptual_class*, cujo atributo *conceptual_class* tem o mesmo valor do atributo id do elemento *conceptual_class* informado, e seu atributo name tem o mesmo valor do atributo name do elemento *attrib* (ou *perspective*, ou *nav_attrib* ou *perspective_attrib*) que está sendo tratado

então

Se o atributo db_type do elemento *map_attribute* não está especificado

então

Chame Converter_Tipo_BD, informando os valores dos atributos type e size que estão sendo tratados e o valor do atributo primary_key do elemento map_attribute

senão

Chame Incluir_Estrutura_Type_Size_PrimKey, informando os valores dos atributos db_type e size que estão sendo tratados e o valor do atributo primary_key do elemento *map_attribute*

fim-Se

senão

Converter_Tipo_BD, informando os valores dos atributos type e size que estão sendo tratados

fim-Se

Converter_Tipo_BD

Se o valor default para o atributo db_type definido na DTD OOHDM_Web contém o valor do atributo type que está sendo tratado

então

Recupera o tipo válido para o banco de dados correspondente ao valor do atributo type

senão

Considere o próprio valor do atributo type

fim-Se.

Chame Incluir_Estrutura_Type_Size_PrimKey, informando os valores dos atributos type e size que estão sendo tratados e o valor do atributo primary_key

Incluir_Estrutura_Type_Size_PrimKey

Inclua na estrutura Lua con_classes a string *attrib_type =* seguida pelo valor do atributo type em questão entre aspas (“)

Se existe valor para o atributo size do elemento referente ao atributo da classe em questão e o valor do seu atributo type é **diferente** de “integer”

então

Inclua uma vírgula logo após o valor do atributo type

Inclua a string *attrib_size =* seguida do valor do atributo size como mostrado abaixo:

attrib_type = “*<valor _type>*”,

attrib_size = *<valor_size>*

fim-Se.

Se o valor do atributo primary_key é “yes” **ou** o elemento referente ao atributo da classe em questão é o primeiro elemento filho (do elemento *conceptual_class* ou *context_class*)

então

Inclua uma vírgula logo após o valor do atributo incluído anteriormente.

Inclua a string prim*_key =* seguida do valor “yes”, como mostrado abaixo:

attrib_type = “*<valor _type>*”,

attrib_size = *<valor_size>,*

prim_key = “yes”

fim-Se.

Especificação da folha de estilo map_classes.xsl

A folha de estilo *map_classes.xsl* é responsável pela descrição do conteúdo da tabela Lua map_classes, que representa o mapeamento das classes conceituais (representadas na tabela lua con_classes) nas tabelas de classe do banco de dados. Este mapeamento é especificado no documento OOHDM-ML através do elemento *map_conceptual_class*. A especificação desta folha de estilo é a seguinte:

Para todos os elementos *conceptual_class* encontrados no documento XML associado a folha de estilos faça

Inclua na estrutura Lua map_classes o valor do atributo name seguido dos caracteres “=” e “{“ da seguinte forma:

<valor_do_atributo_name> =

{

Se existe um elemento *map_conceptual_class* especificado no documento XML, cujo atributo *conceptual_class* tenha o mesmo valor do atributo id do elemento *conceptual_class* que está sendo tratado

então

Se existe um valor especificado para o atributo db_class do elemento *map_conceptual_class* especificado

então

Inclua na estrutura Lua map_classes a string *db_class =* seguida pelo valor do atributo db_class, entre aspas (“) seguido por vírgula, da seguinte forma:

db_class = “*<valor_do_atributo_db_class >”,*

senão

Inclua a string *db_class =* seguida pelo valor do atributo name do elemento *conceptual_class* que está sendo tratado da seguinte forma:

db_class = "*<valor_do_atributo_name_conceptual_class >",*

fim-Se.

Se o elemento *map_conceptual_class* possui algum elemento filho *map_attribute*

então

Inclua na estrutura Lua map_classes a string *attributes* = seguida do caracter "{" da seguinte forma:

attributes =

{

Chame Tratar_Elementos_Map_Attribute

fim-Se.

senão (não há mapeamento para a classe)

Inclua a string *db_class =* seguida pelo valor do atributo name do elemento conceptual_class que está sendo tratado da seguinte forma:

db_class = "*<valor_do_atributo_name_conceptual_class >",*

fim-Se.

Inclua na estrutura Lua map_classes o símbolo }

Se o elemento conceptual_class não é o último elemento tratado

então

Inclua uma vírgula seguindo o símbolo } incluído anteriormente

fim-Se

fim-Para.

Tratar_Elementos_Map_Attribute:

Para cada elemento *map_attribute*, filho do elemento *map_conceptual_class* em questão faça

Inclua o valor do atributo name, seguido pelo símbolo "=" e pelo valor do atributo db_attribute entre aspas ("), da seguinte forma:

valor_do_atributo_name> = "*<valor_do_atributo_db_attribute>*"

Se o elemento *map_attribute* não é o último elemento tratado

então

Inclua uma vírgula após o valor do atributo db_attribute.

fim-Se

fim-Para

Inclua na estrutura Lua map_classes o símbolo "}" .

Especificação da folha de estilo con_relationships.xsl

A folha de estilo *con_relationships.xsl* é responsável pela descrição do conteúdo da tabela Lua con_relationships, que representa os relacionamentos entre os objetos das classes conceituais. Estes relacionamentos são especificados no documento OOHDM-ML através do elemento *relationship*. A especificação desta folha de estilo é a seguinte:

Para todos os elementos *relationship* encontrados no documento XML associado faça

Inclua na estrutura Lua con_relationships o valor do atributo name do elemento *relationship* seguido dos caracteres "=" e "{" da seguinte forma:

<valor_do_atributo_name> =

{

Inclua a string “source_class =” seguida pelo valor do atributo name do elemento *conceptual_class*, cujo atributo id tenha o mesmo valor do atributo source_class do elemento *relationship*. O valor do atributo name deve estar entre aspas (“) e seguido de vírgula, como mostrado abaixo.

source_class = “<valor_do_atributo_name> “,

Inclua a string “destination_class =” seguida pelo valor do atributo name do elemento *conceptual_class*, cujo atributo id tenha o mesmo valor do atributo target_class do elemento *relationship*. O valor do atributo name deve estar entre aspas (“) e seguido de vírgula, como mostrado abaixo.

destination_class = “<valor_do_atributo_name> “,

Inclua a string “source_card =” seguida pelo valor do atributo source_cardinality do elemento *relationship*. Este valor deve estar entre aspas (“) e seguido de vírgula, como mostrado abaixo.

source_card = “<valor_do_atributo_source_cardinality> “,

Inclua a string “dest_card =” seguida pelo valor do atributo target_cardinality do elemento *relationship*. Este valor deve estar entre aspas (“) como mostrado abaixo.

dest_card = “<valor_do_atributo_target_cardinality> “

Inclua o símbolo “} “ alinhando-o com o símbolo “{“ inserido após o valor do atributo name.

Se o elemento relationship não é o último elemento encontrado

então

Inclua uma vírgula seguindo o símbolo “}“ incluído anteriormente.

fim-Se

fim-Para.

Especificação da folha de estilo map_relationships.xsl

A folha de estilo *map_relationships.xsl* é responsável pela descrição do conteúdo da tabela Lua map_relationships, que representa o mapeamento dos relacionamentos (representadas na tabela lua con_relationships) nas tabelas do banco de dados. Este mapeamento é especificado no documento OOHDM-ML através do elemento *map_relationship*. A especificação desta folha de estilo é a seguinte:

Para todos os elementos *relationship* encontrados no documento XML associado faça

Inclua na estrutura Lua map_relationships o valor do atributo name do elemento *relationship* seguido dos caracteres “=” e “{“ da seguinte forma:

<valor_do_atributo_name> =

{

Se existe algum elemento *db_source_field*, filho do elemento *map_relationship*, cujo atributo relationship tem o mesmo valor do atributo id do elemento relationship que está sendo tratado

então

Inclua na estrutura Lua map_relationships a string ‘db_source_field = {‘

Para cada elemento *db_source_field* filho do elemento *map_relationship* faça

Se existe valor especificado para o atributo map_attribute

então

Inclua o valor do atributo name do elemento *map_attribute*, cujo atributo id tem o mesmo valor do atributo map_attribute encontrado acima, o símbolo “=” e o valor existente entre as tags do elemento *db_source_field* em questão.

senão

Inclua o valor do atributo name do primeiro elemento filho *attrib* do elemento *conceptual_class*, cujo atributo id tem o mesmo valor do atributo source_class do elemento *relationship* que está sendo tratado, o símbolo “=” e o valor existente entre as tags do elemento *db_source_field* em questão, entre aspas (“).

fim-Se.

Se o elemento que está sendo tratado não é o último

então

Inclua uma vírgula.

senão

Inclua o símbolo "}" seguido de vírgula.

fim-Se.

fim-Para.

fim-Se.

Se existe algum elemento *db_destination_field*, filho do elemento *map_relationship*, cujo atributo relationship tem o mesmo valor do atributo id do elemento relationship que está sendo tratado

então

Inclua na estrutura Lua map_relationships a string 'db_destination_field = {'

Para cada elemento *db_destination_field* filho do elemento *map_relationship* faça

Se existe valor especificado para o atributo map_attribute desse elemento *db_destination_field*

então

Inclua o valor do atributo name do elemento map_attribute, cujo atributo id tem o mesmo valor do atributo map_attribute encontrado acima, o símbolo "=" e o valor existente entre as tags do elemento db_destination_field em questão.

senão

Inclua o valor do atributo name do primeiro elemento filho *attrib* do elemento *conceptual_class*, cujo atributo id tem o mesmo valor do atributo target_class do elemento *relationship* que está sendo tratado, o símbolo "=" e o valor existente entre as tags do elemento *db_destination_field* em questão, entre aspas (").

fim-Se.

Se o elemento que está sendo tratado não é o último

então

Inclua uma vírgula.

senão

Inclua o símbolo "}" seguido de vírgula.

fim-Se.

fim-Para.

fim-Se.

Inclua na estrutura Lua map_relationships a string db_relationship = seguida pelo valor do atributo db_relationship do elemento *map_relationship*, cujo atributo relationship tenha o mesmo valor do atributo id do elemento *relationship* que está sendo tratado. O valor do atributo db_relationship deve estar entre aspas ("), como mostrado abaixo:

db_relationship = "<valor_do_atributo_db_relationship>"

Inclua na estrutura map_relationship o símbolo "}"

Se o elemento relationship não é o último elemento que está sendo tratado

então

Inclua uma vírgula seguindo o símbolo "}" incluído anteriormente

fim-Se

fim-Para.

Especificação da folha de estilo queries.xsl

A folha de estilo *queries.*xsl é responsável pela descrição do conteúdo da tabela Lua *queries*, que representa a query que obtém todos os atributos que pertence a um nó do modelo de navegação, e as classes em contexto relacionadas a ele. O nó de navegação e as classes em contexto são especificados no documento OOHDM-ML através dos elementos *navigational_class* e *context_class*. A especificação desta folha de estilo é a seguinte:

Para todos os elementos *navigational_class* encontrados no documento XML associado a folha de estilos faça

Inclua na estrutura Lua queries o valor do atributo name do elemento navigational_class seguido dos caracteres "=" e "{" da seguinte forma:

<valor_do_atributo_name> =

{

Inclua em seguida a string "*class* = "

Se os valores do atributo conceptual_class de **todos** os elementos *nav_attrib* e *perspective_attrib* filhos do elemento *navigational_class* forem **iguais** (todos os atributos e perspectivas do nó tem como origem uma única classe conceitual)

então

Inclua, logo após a string *class* = , o valor do atributo name do elemento *conceptual_class*, cujo atributo id tenha o mesmo valor do atributo conceptual_class do primeiro elemento *nav_attrib*, ou *perspective_attrib*, filho do elemento *navigational_class* que está sendo tratado. Este valor deve estar entre aspas (“), delimitado por chaves e seguido de uma vírgula, como mostrado abaixo:

class = {"*<valor_do_atributo_name_conceptual_class>*"},

Chame Tratar_Tabela_Unica, informando o valor do atributo conceptual_class.

senão (os atributos e perspectivas do nó <u>não</u> têm como origem uma única classe)

Inclua, logo após a string *class* =, todos os valores **distintos** do atributo name do elemento *conceptual_class*, cujo atributo id tenha o mesmo valor do atributo *conceptual_class* dos elementos *nav_attrib*, e *perspective_attrib*, filhos do elemento *navigational_class* que está sendo tratado. Estes valores deve estar entre aspas (“), delimitado por chaves e seguido de uma vírgula, como mostrado abaixo:

class = {"*<valor_atributo_name_conceptual_class1">,*
"<valor_atributo_name_conceptual_classr2>",...},

Chame Tratar_Tabelas_Distintas.

Inclua na estrutura queries a string '*where* = " '.

Para todos os elementos *relationship* no documento XML, cujos atributos source_class e target_class possuem valores iguais aos valores **distintos** do atributo conceptual_class dos elementos nav_attrib e perspective_attrib faça

Se os atributos source_cardinality e target_cardinality têm como valor o caracter "**n**" (relacionamento "n" pra "n")

então

Inclua, após a string 'where = " ', os seguintes formatos:

<valor_target_class_bd> ❶ **.** <valor_atributo_chave>❷ **=**
<valor_db_relationship>❸ **.** <valor_db_destination_field>❸ AND
<valor_source_class_bd> ❶ **.** <valor_atributo_chave>❷ **=**
<valor_db_relationship>❸ **.** <valor_db_source_field>❸

senão (relacionamentos "1" pra "1" e "1" pra "n")

Se <valor_db_relationship>❸ = valor do atributo target_class

então

Inclua, após a string 'where = " ', o seguinte formato:

<valor_source_class_bd> ❶ **.** <valor_atributo_chave>❷ **=**

<valor_db_relationship>❸ **.** <valor_db_source_field>❸

senão

<valor_target_class_bd> ❶ **.** <valor_atributo_chave>❷ **=**

<valor_db_relationship>❸ **.** <valor_db_destination_field>❸

fim-Se.

fim-Se.

Se o elemento relationship não é o último que está sendo tratado

então

Inclua a string 'AND' após o último valor incluído após a string 'where = "

fim-Se.

fim-Para

fim-Se.

Se existe um elemento *context_class* no documento XML, cujo valor do atributo base_class seja igual ao valor do atributo id do elemento *navigational_class* que está sendo tratado

então

Inclua uma vírgula após o último valor incluído na estrutura queries.

Chame Tratar_Elementos_Context_Class, informando o valor do atributo id do elemento context_class.

fim-Se

Inclua o símbolo "}" .

Se o elemento navigational_class não é o último elemento que está sendo tratado

então

Inclua uma vírgula seguindo o símbolo "}" incluído anteriormente

fim-Se

fim-Para.

Legenda:

❶ <*valor_target_class_bd*> e <*valor_source_class_bd*> devem ser encontrados da seguinte forma:

Se existe algum elemento *map_conceptual_class* cujo atributo conceptual_class tem o mesmo valor do atributo target_class (ou source_class) que está sendo tratado **e** existe um valor especificado para o seu atributo db_class

então

O nome da classe conceitual é igual ao valor do atributo db_class do elemento map_conceptual_class considerado.

senão

O nome da classe conceitual é igual ao valor do atributo name do elemento conceptual_class, cujo atributo id tem o mesmo valor do atributo target_class (ou source_class)

fim-Se.

❷ <valor_atributo_chave> deve ser encontrado da seguinte forma:

Se existe algum elemento map_attribute, filho do elemento *map_conceptual_class* cujo atributo conceptual_class tem o mesmo valor do atributo target_class (ou source_class) que está sendo tratado, com o valor do atributo primary_key igual a "yes"

então

O valor do atributo chave é igual ao valor do atributo db_attribute do elemento map_attribute considerado.

senão

O valor do atributo chave é igual ao valor do atributo name do primeiro elemento attrib filho do elemento conceptual_class, cujo atributo id tem o mesmo valor do atributo target_class (ou source_class)

fim-Se.

❸ <valor_db_relationship>, <valor_db_source_field> e <valor_db_destination_field> correspondem, respectivamente, ao valor do atributo db_relationship e o valor existente entre as tags dos elementos db_source_field e db_destination_field, filhos do elemento map_relationship, cujo atributo relationship tem o mesmo valor do atributo id do elemento relationship que está sendo tratado.

Tratar_Tabela_Unica:

Inclua na estrutura queries a string “tables = ”.

Se existe um elemento map_conceptual_class, cujo atributo conceptual_class tenha o mesmo valor do atributo conceptual_class do primeiro elemento nav_attrib ou perspective_attrib, filho do elemento navigational_class que está sendo tratado

então

Se existe um valor especificado para o atributo db_class do elemento map_conceptual_class encontrado acima

então

Inclua, após a string “tables = “, o valor do atributo db_class do elemento map_conceptual_class entre aspas (“), delimitado por chaves e seguido de uma vírgula, como mostrado abaixo:

tables = {“<valor_do_atributo_db_class>”},

senão

Inclua, após a string “tables = “, o valor do atributo name do elemento conceptual_class, cujo atributo id tenha o mesmo valor do parametro par_conceptual_class.

tables = {“<valor_do_atributo_name_conceptual_class>”},

fim-Se.

Inclua na estrutura queries a string “fields = { ”

Para cada elemento nav_attrib e perspective_attrib, filhos do elemento navigational_class que está sendo tratado faça

Se existe um elemento map_attribute, filho do elemento map_conceptual_class encontrado acima, e seu atributo name tem o mesmo valor do atributo name do elemento nav_attrib ou perspective_attrib em questão

então

Se o valor de seu atributo primary_key é igual a “no”

então

Inclua, após a string “fields = “, o valor do atributo db_attribute do elemento map_attribute encontrado acima, caso não exista valor especificado para o atributo primary_key. Este valor deve suceder o valor especificado após a string “tables =”, entre aspas (“), delimitado por chaves e seguido de uma vírgula, como mostrado abaixo:

“<valor_em_tables>.<valor_atributo_db_attribute>”

fim-Se.

senão

Se o elemento nav_attrib ou perspective_attrib não é o primeiro elemento filho do elemento navigational_class

então

Inclua, após a string “fields = “, o valor do atributo name do elemento nav_attrib ou perspective_attrib que está sendo tratado, caso este não seja o primeiro elemento filho do elemento navigational_class. Este valor deve suceder o valor especificado após a string “tables=”, entre aspas (“), delimitado por chaves e seguido de uma vírgula, como mostrado abaixo:

“<valor_em_tables>.<valor_atributo_name>”,

fim-Se.

fim-Se.

Se o elemento nav_attrib ou perspective_attrib não é o último elemento tratado

então

Inclua uma vírgula, após o valor especificado acima.

fim-Se.

fim-Para.

Inclua o símbolo “}”.

Senão (não existe map_conceptual_class)

Inclua, após a string “tables = “, o valor do atributo name do elemento conceptual_class, cujo atributo id tenha o mesmo valor do atributo conceptual_class do primeiro elemento nav_attrib ou perspective_attrib, entre aspas (“), delimitado por chaves e seguido de uma vírgula, como mostrado abaixo:

tables = {“<valor_do_atributo_name_conceptual_class>”},

Inclua a string “fields = “ seguida pelos valores do atributo name de todos os elementos nav_attrib ou perspective_attrib filhos do elemento navigational_class que está sendo tratado. Este valor deve suceder o valor especificado após a string “tables =”, entre aspas (“), delimitado por chaves e seguido de uma vírgula, como mostrado abaixo:

fields = {“<valor_em_tables>.<valor_atributo_name>”,
“<valor_em_tables>.<valor_atributo_name>” }

fim-Se.

Tratar_Tabelas_Distintas:

Inclua na estrutura queries a string “*tables* = ”

Para todos os valores distintos do atributo conceptual_class dos elementos *nav_attrib* e *perspective_attrib* faça

Se existe um elemento, cujo atributo conceptual_class tem valor igual ao encontrado *map_conceptual_class* acima

então

Se existe um valor especificado para o atributo db_class do elemento map_conceptual_class encontrado acima

então

Inclua, após a string “tables = “, o valor do atributo db_class do elemento map_conceptual_class entre aspas (“), delimitado por chaves e seguido de uma vírgula, como mostrado abaixo:

tables = {“*<valor_db_class1>*”,“*<valor_db_class2>*”,...},

senão

Inclua, após a string “tables = “, o valor do atributo name do elemento conceptual_class, cujo atributo id tenha o mesmo valor do parametro par_conceptual_class.

tables = {“*<valor_do_atributo_name_conceptual_class>*”},

fim-Se.

senão

Inclua, após a string “tables = “, o valor do atributo name do elemento conceptual_class, cujo atributo id tem valor igual ao valor encontrado acima, entre aspas (“), delimitado por chaves e seguido de uma vírgula, como mostrado abaixo:

tables = {“*<valor_name1>*”,“*<valor_name2>*”,…},

fim-Se.

fim-Para.

Para todos os elementos *relationship* no documento XML, cujos atributos source_class e target_class possuem valores iguais aos valores distintos do atributo conceptual_class dos elementos nav_attrib e perspective_attrib, e seus atributos source_cardinality e target_cardinality têm o mesmo valor igual a “n” faça

Acrescente à string “tables = “, após o último valor especificado, o valor do atributo db_relationship do elemento map_relationship, cujo atributo relationship tem o mesmo valor do atributo id do elemento relationship que está sendo tratado, como mostrado abaixo:

tables = {“*<valor_table1>*”,“*<valor_table2>*”,“*<valor_table3>*”,
“*<valor_atributo_db_relationship>*”},

fim-Para.

Inclua na estrutura queries a string “fields = { ”

Para todos os elementos nav_attrib e perspective_attrib faça

Se existe um elemento map_attribute, filho de um elemento map_conceptual_class cujo atributo conceptual_class tem o mesmo valor do atributo conceptual_class do elemento nav_attrib ou perspective_attrib que está sendo tratado, e seu atributo name tem o mesmo valor do atributo name do elemento que está sendo tratado

então

Se existe um valor especificado para o atributo db_class do elemento map_conceptual_class em questão

então

Inclua, após a string “fields = {“, os valores dos atributos db_class e db_attribute separados por ponto (.) e entre aspas (“), conforme mostrado abaixo:

fields = {“*<valor_db_class>*.*<valor_db_attribute>*”

senão

Inclua, após a string “fields = {“, o valor do atributo name do elemento conceptual_class, cujo atributo id tem o mesmo valor do atributo conceptual_class do elemento map_conceptual_class, e o valor do atributo db_attribute separados por ponto (.) e entre aspas (“), conforme mostrado abaixo:

fields = {“*<valor_name_conceptual_class>*.*<valor_db_attribute>*”

fim-Se.

senão

Se existe um valor especificado para o atributo db_class do elemento map_conceptual_class em questão

então

Inclua, após a string “fields = {“, o valor do atributo db_class do elemento map_conceptual_class e do atributo name do elemento nav_attrib ou perspective attrib em questão, separados por ponto (.) e entre aspas (“), conforme mostrado abaixo:

fields = {“*<valor_db_class>*.*<valor_atributo_name>*”

senão

Inclua, após a string “fields = {“, o valor do atributo name do elemento conceptual_class, cujo atributo id tem o mesmo valor do atributo conceptual_class do elemento map_conceptual_class, e o valor do atributo db_attribute separados por ponto (.) e entre aspas (“), conforme mostrado abaixo:

fields = “<*valor_name_conceptual_class*>.<*valor_name*>”

fim-Se.

fim-Se.

fim-Para.

Tratar_Elementos_Context_Class:

Crie a estrutura Lua class_ctx incluindo em queries o formato abaixo:

class_ctx =

{

Para cada elemento *navigational_context* especificado no atributo contexts do elemento *context_class* que está sendo tratado faça

Inclua na estrutura Lua class_ctx o valor do atributo name do elemento *navigational_context* seguido dos caracteres “=” e “{“ da seguinte forma:

<valor_do_atributo_name> =

{

Para cada elemento *nav_attrib* filho do elemento *context_class* faça

Inclua na estrutura Lua class_ctx o valor do atributo name do elemento nav_attrib seguido dos caracteres “=” e “{“ da seguinte forma:

<valor_do_atributo_name> =

{

Chame Tratar_Atributos_Type_Size, informando os valores dos atributos type, size e *conceptual_class* do elemento *nav_attrib*.

Se o elemento *nav_attrib* é o último elemento filho e não existem elementos perspective_attrib filhos do elemento context_class

então

Inclua na estrutura Lua class_ctx o símbolo }

senão

Inclua na estrutura Lua class_ctx o símbolo } seguido de vírgula

fim-Se

fim-Para.

Para cada elemento perspective_attrib filho do elemento context_class faça

Inclua na estrutura Lua class_ctx o valor do atributo name do elemento perspective_attrib seguido dos caracteres “=” e “{“ da seguinte forma:

<valor_do_atributo_name> =

{

Chame Tratar_Atributos_Type_Size, informando os valores dos atributos type, size e conceptual_class do elemento perspective_attrib.

Se o elemento perspective_attrib é o último elemento filho tratado

então

Inclua na estrutura Lua class_ctx o símbolo }

senão

Inclua na estrutura Lua class_ctx o símbolo } seguido de vírgula

fim-Se

fim-Para.

Inclua na estrutura Lua class_ctx o símbolo }

Se o elemento navigational_context não é o último elemento tratado

então

Inclua uma vírgula seguindo o símbolo } incluído anteriormente.

fim-Se

fim-Para.

Complete a estrutura class_ctx incluindo o símbolo "}".

Especificação da folha de estilo map_classes_ctx.xsl

A folha de estilo *map_classes_ctx.*xsl é responsável pela descrição do conteúdo da tabela Lua *map_classes_ctx*, que representa o mapeamento das classes em contexto (representadas na tabela lua queries) nas tabelas de classe do banco de dados. Este mapeamento é especificado no documento OOHDM-ML através do elemento *map_context_class*. A especificação desta folha de estilo é a seguinte:

Para todos os elementos *context_class* encontrados no documento XML associado a folha de estilos faça

Inclua na estrutura Lua map_classes_ctx o valor do atributo name do elemento *navigational_class*, cujo atributo id tenha o mesmo valor do atributo base_class do elemento *context_class* que está sendo tratado, seguido dos caracteres "=" e "{" da seguinte forma:

<valor_do_atributo_name_navigational_class> =

{

Inclua na estrutura Lua map_classes_ctx a string *db_class_ctx* = seguida pelo valor do atributo db_class_ctx do elemento *map_context_class*, cujo atributo context_class tem o mesmo valor do atributo id do elemento *context_class* que está sendo tratado. Este valor deve estar entre aspas (") como mostrado abaixo:

db_class_ctx = "*<valor_do_atributo_db_class_ctx >*"

Se existem elementos *map_attribute*, filhos do elemento *map_context_class*, cujo atributo context_class tem o mesmo valor do atributo id do elemento *context_class* que está sendo tratado

então

Inclua uma vírgula após o valor do atributo db_class_ctx.

Inclua na estrutura Lua map_classes_ctx a string *attrib_class_ctx* = seguida do caracter "{" da seguinte forma:

attrib_class_ctx =

{

Chame Tratar_Elementos_Map_Attribute

fim-Se.

Inclua na estrutura Lua map_classes_ctx o símbolo }

Se o elemento *context_class* não é o último elemento tratado

então

Inclua uma vírgula seguindo o símbolo } incluído anteriormente

fim-Se

fim-Para.

Tratar_Elementos_Map_Attribute:

Para cada elemento map_attribute filho do elemento *map_context_class*, cujo atributo context_class tem o mesmo valor do atributo id do elemento *context_class* que está sendo tratado faça

Inclua o valor do atributo name, seguido pelo símbolo "=" e pelo valor do atributo db_attribute entre aspas ("), da seguinte forma:

<valor_do_atributo_name> = "*<valor_do_atributo_db_attribute>*"

Se o elemento *map_attribute* não é o último elemento tratado

então

Inclua uma vírgula após o valor do atributo db_attribute

fim-Se

fim-Para

Inclua na estrutura Lua map_classes_ctx o símbolo "}"

Especificação da folha de estilo nav_contexts.xsl

A folha de estilo *nav_contexts.xsl* é responsável pela descrição do conteúdo da tabela Lua nav_contexts, que representa os contextos do projeto de navegação de uma aplicação hipermídia. Os contextos de navegação são especificados no documento OOHDM-ML utilizando-se o elemento *navigational_context*. A especificação desta folha de estilo é a seguinte:

Para todos os elementos *navigational_context* encontrados no documento XML associado a folha de estilos faça

Inclua na estrutura Lua nav_contexts o valor do atributo name seguido dos caracteres "=" e "{" da seguinte forma:

<valor_do_atributo_name> =

{

Se existe mais de um elemento *order*, filho do elemento *navigational_context* que está sendo tratado

então

Chame Verificar_Atributo_Type, informando 'M' (múltipla ordenação)

senão

Chame Verificar_Atributo_Type, sem informar valor

fim-Se.

Inclua na estrutura nav_contexts a string '*nav_type* = ' seguida por um dos valores determinados de acordo com o valor do atributo navigation_type, conforme a tabela abaixo. Os valores devem estar entre aspas (") seguido de vírgula.

Navigation_type	Valor
Sequential	"S"
Circular	"C"
Index	"I"
Free	"L"
sequential_index	"SI"
Circular_index	"CI"

Se existe algum elemento *nav_parameter* descendente de um elemento selection, filho do elemento *navigational_context*

então

Chame Tratar_Elementos_Nav_Parameter.

senão

Se o valor do atributo type do elemento *navigational_context* informado for igual a “by_query”

então

Inclua a string *parameters = {“sql_statement”},*

fim-Se.

fim-Se.

Inclua na estrutura Lua nav_contexts a string ‘*elements* = {‘.

Para cada valor especificado no atributo element_class faça

Inclua na estrutura nav_contexts o valor do atributo name do elemento navigational_class, cujo atributo id tem valor igual ao valor do atributo element_class que está sendo tratado seguido pelos símbolos ‘=’ e ‘{‘ como mostrado abaixo:

<valor_do_atributo_name> =

{

Se existe algum elemento *order_element*

então

Se existe mais de um valor no atributo element_class cujo atributo navigational_class tem valor igual ao valor do atributo element_class que está sendo tratado

então

Chame Tratar_Elementos_Order_Element, informando o elemento order_element cujo atributo navigational_class tem valor igual ao valor do atributo element_class que está sendo tratado.

senão

Chame Tratar_Elementos_Order_Element, informando apenas o elemento order_element.

fim-Se.

fim-Se.

Inclua na estrutura nav_contexts a string ‘*dest_page* = ‘, seguida pelo valor do elemento interface_element (filho do elemento OOHDM-WEB), cujo valor do atributo target seja igual ao valor do atributo element_class que está sendo tratado. Esse valor deve estar entre aspas (“), como mostrado abaixo:

dest_page = “*<valor_do_elemento_interface_element>*”

Se existe algum elemento *selection* filho do elemento navigational_context que está sendo tratado

então

Inclua na estrutura nav_contexts uma vírgula seguida pela string ‘*selection* = ‘ e pelo símbolo { da seguinte forma:

selection =

{

Para cada elemento selection encontrado faça

Se o valor do atributo selected_element_class for igual ao valor do atributo element_class que está sendo tratado **OU** se não existe valor especificado para esse atributo

então

Se existe valor especificado para o atributo link de algum elemento attribute descendente do elemento selection que está sendo tratado

então

Inclua na estrutura nav_contexts a string ‘link_name = {‘

Para cada elemento *attribute*, descendente do elemento selection em questão, que possui um atributo link especificado faça

Inclua valor do atributo name do elemento relationship cujo atributo id tem o mesmo valor do atributo link especificado entre aspas (“) .

Se o elemento atributo não é o último

então

Inclua uma vírgula.

fim-Se.

fim-Para.

Inclua o símbolo fecha-chaves “}” seguido de virgula.

fim-Se.

Inclua na estrutura nav_contexts a string ‘condition = “(‘.

Se o elemento filho do elemento selection em questão for um dos seguintes elementos: *equal*, *different*, *greater*, *greater_equal*, *lesser* ou *lesser_equal*

então

Chame Tratar_Condicao_Relacional, informando o elemento filho.

senão

Chame Tratar_Condicao_Logica, informando o elemento filho.

fim-se.

Inclua o símbolo fecha-parenteses ‘)’ seguido de aspas (“).

fim-Se.

fim-Para.

Inclua na estrutura Lua nav_contexts o símbolo } fechando a estrutura de selection.

senão (se nao existe elemento selection)

Se existe algum elemento *instance*, filho do elemento *navigational_context*, cujo atributo navigational_class tem valor igual ao valor do atributo element_class que está sendo tratado

então

Chame Incluir_Condicao_Contexto_Arbitrario

senão

Se o valor do atributo type do elemento navigational_context que está sendo tratado for igual a “by_query”

então

Inclua na estrutura nav_contexts uma vírgula seguida pela string abaixo:

selection =

{

condition = “(#sql_statement)”

fim-Se.

fim-Se.

fim-Se.

Inclua na estrutura Lua nav_contexts o símbolo }, fechando a estrutura do valor de element_class que está sendo tratado.

fim-Para

Inclua na estrutura Lua nav_contexts o símbolo } fechando a estrutura de element_class.

Inclua na estrutura Lua nav_contexts o símbolo } fechando a estrutura do valor do atributo name do elemento navigational_context.

Se o elemento navigational_context não é o último elemento tratado

então

Inclua uma vírgula seguindo o símbolo } incluído anteriormente

fim-Se

fim-Para.

Verificar_Atributo_Type:

Se o valor do atributo type do elemento navigational_context que está sendo tratado for igual a **static**

então

Chame Derminar_Tipo_Contexto, informando 'E' ou 'EM' (caso o valor informado seja 'M')

senão

Se o valor do atributo type do elemento navigational_context que está sendo tratado for igual a **dynamic**

então

Chame Derminar_Tipo_Contexto, informando 'D' ou 'DM' (caso o valor informado seja 'M')

senão

Se o valor do atributo type do elemento navigational_context que está sendo tratado for igual a **temporary**

então

Chame Derminar_Tipo_Contexto, informando 'T' ou 'TM' (caso o valor informado seja 'M')

senão (**by_query**)

Inclua na estrutura nav_context a string 'context = ', seguida pela string "DC" ou "DCM" (caso o valor informado seja 'M') seguida de vírgula.

fim-Se.

fim-Se.

fim-Se.

Determinar_Tipo_Contexto:

Se existe um elemento *nav_parameter* descendente de um elemento selection, filho do elemento navigational_context que está sendo tratado (**grupo de contexto**)

então

Inclua na estrutura nav_contexts a string 'context = ' seguida pela string "EG" (caso o valor informado seja 'E'), "EGM" (caso o valor informado seja 'EM'), "DG" (caso o valor informado seja D), "DGM" (caso o valor informado seja D), "TG" (caso o valor informado seja T) ou "TGM" (caso o valor informado seja TM), seguida de vírgula

senão

Se existe algum elemento *instance*, filho do elemento navigational_context que está sendo tratado (contexto arbitrário)

então

Inclua na estrutura nav_contexts a string 'context = ' seguida pela string "EA" (caso o valor informado seja 'E'), "EAM" (caso o valor informado seja 'EM'), "DA" (caso o valor informado seja D), "DAM" (caso o valor informado seja D), "TA" (caso o valor informado seja T) ou "TAM" (caso o valor informado seja TM), seguida de vírgula.

senão

Inclua na estrutura nav_contexts a string 'context = ', seguida pela string "ES" (caso o valor informado seja 'E'), "ESM" (caso o valor informado seja 'EM'), "DS" (caso o valor informado seja D), "DSM" (caso o valor informado seja DM), "TS" (caso o valor informado seja T) ou "TSM" (caso o valor informado seja TM), seguida de vírgula.

fim-Se.

fim-se.

Tratar_Elementos_Nav_Parameter:

Inclua a string 'parameters = {'

Para todos os elementos *nav_parameter* existentes faça

Inclua o valor do elemento nav_parameter entre aspas (“).

Se o elemento nav_parameter não é o último elemento tratado

então

Inclua uma vírgula após o valor incluído.

fim-Se

fim-Para.

Inclua o símbolo } seguido de vírgula

Tratar_Elemento_Order_Element:

Inclua na estrutura nav_contexts a string ‘*order* = { ‘

Para **todos** os elementos order_element, cujo atributo navigational_class tem valor igual ao valor do atributo element_class que está sendo tratado faça

Inclua, após a string ‘*order* = { ‘, o valor do atributo name seguido pelo símbolo ‘=’ e pelo valor do atributo criteria entre aspas (“) como mostrado abaixo:

order = { *<valor_atributo_name>* = “*<valor_atributo_criteria>*”

Se o elemento element_order é o último elemento tratado

então

Inclua o símbolo } seguido de vírgula

senão

Inclua uma vírgula após o último valor incluído.

fim-Se

fim-Para.

Incluir_Condicao_Contexto_Arbitrario:

Inclua na estrutura nav_contexts a string ‘*selection* = ‘ seguida pelo símbolo { e pela string ‘condition = ’da seguinte forma:

selection =

{

condition =

Para todos os elementos instance, filhos do elemento navigational_context, cujo atributo navigational_class tem valor igual ao valor do atributo element_class que está sendo tratado, faça

Inclua aspas (“) e o símbolo abre-parenteses.

Para cada elemento instance_attrib filho do elemento instance que está sendo tratado faça

Inclua o valor do atributo name do elemento filho instance_attrib, seguido pelo símbolo “=”.

Se o valor do atributo type do elemento attrib ou perspective_attrib, cujo atributo name tem o mesmo valor do atributo name do elemento instance_attrib acima, for diferente de **integer**, **real** e **boolean**

então

Inclua o valor do elemento value (PCDATA) entre aspas simples (‘), como mostrado abaixo:

‘*<valor_elemento_instance_attrib>*’

senão

Inclua o valor do elemento value (PCDATA) seguido do símbolo e o valor (PCDATA) desse elemento.

fim-Se.

Se o elemento instance_attrib que está sendo tratado **não** é o último
então
Inclua uma vírgula após o valor incluído.
fim-Se.
fim-Para.
Se o elemento instance que está sendo tratado **não** é o último
então
Inclua o símbolo fecha-paranteses e a string 'OR' seguida de aspas (") e dois pontos finais (..)
senão
Inclua o símbolo fecha-paranteses, seguido de aspas (").
fim-Se.
fim-Para.
Após o último elemento instance ser tratado deve-se obter o seguinte formato:

condition = "(<*valor_atributo_name1*> = '<*valor_instance_attrib*>') OR ".. "(<*valor_atributo_name2*> = '<*valor_instance_attrib*>') OR ".. "(<*valor_atributo_name3*> = '<*valor_instance_attrib*>')"

Tratar_Condicao_Relacional:

Se o *primeiro* elemento filho do elemento informado é um elemento ***attribute***
então
Inclua o valor do atributo name do elemento attribute seguido pelo símbolo correspondente ao elemento informado.
senão
Chame Tratar_Condicao_Aritmetica, informando o primeiro elemento filho.
Inclua o símbolo correspondente ao elemento informado (ver tabela de símbolos)
fim-Se.
Se o *segundo* elemento filho do elemento informado (parâmetro) é um elemento ***attribute***
então
Inclua o valor do atributo name do segundo elemento filho attribute.
senão
Se o segundo elemento filho é um elemento ***value***
então
Chame Incluir_Valor_Elemento_Value
senão
Se o segundo elemento filho é um elemento ***nav_parameter***
então
Inclua o símbolo # seguido pelo valor do segundo elemento filho nav_parameter
senão (o segundo elemento filho é plus, minus, mult ou div)
Chame Tratar_Condicao_Aritmetica, informando o valor do segundo elemento filho.
fim-Se.
fim-Se.
fim-Se.

Incluir_Valor_Elemento_Value:

Se o primeiro elemento filho é attribute e existe algum elemento nav_attrib ou perspective_attrib, filhos do elemento navigational_class cujo valor do atributo id é igual ao

valor do atributo navigational_class do primeiro elemento filho, cujo valor do atributo name é igual ao valor do atributo name do primeiro elemento filho

então

Se o valor do atributo name do elemento attrib (ou perspective filho do elemento attrib), filho do elemento conceptual_class cujo atributo id tem o mesmo valor do atributo conceptual_class do elemento nav_attrib ou perspective_attrib considerado acima, é igual ao valor do atributo name do primeiro elemento filho

então

Se o valor do atributo type do elemento attrib ou perspective_attrib encontrado acima for diferente de **integer**, **real** e **boolean**

então

Inclua o valor do elemento value (PCDATA) entre aspas simples (') seguido do símbolo fecha-parenteses ")", como mostrado abaixo:

'*<valor_elemento_value>*')

senão

Inclua o valor do elemento value (PCDATA) seguido do símbolo fecha-parenteses ")".

fim-Se.

fim-Se.

senão

Inclua o valor do elemento value (PCDATA) seguido do símbolo fecha-parenteses ")".

fim-Se.

Tratar_Condicao_Aritmetica:

Inclua o valor do attribute name do primeiro elemento filho attribute do elemento informado seguido pelo símbolo correspondente deste(ver tabela de simbolos).

Se o *segundo* elemento filho do elemento informado (parâmetro) é um elemento ***attribute***

então

Inclua o valor do atributo name do segundo elemento filho attribute.

senão

Se o segundo elemento filho é um elemento ***value***

então

Chame Incluir_Valor_Elemento_Value

senão

Se o segundo elemento filho é um elemento ***nav_parameter***

então

Inclua o símbolo # seguido pelo valor do segundo elemento filho nav_parameter

fim-Se.

fim-Se.

fim-Se.

Tratar_Condicao_Logica:

Se o *primeiro* elemento filho do elemento informado for um dos seguintes elementos: equal, different, greater, greater_equal, lesser ou lesser_equal

então

Chame Tratar_Condicao_Relacional, informando o valor desse primeiro elemento filho.

Inclua o símbolo correspondente ao elemento informado (ver tabela de símbolos) seguido de aspas (") e dois pontinhos (..), como no exemplo abaixo.

<valor>) **AND "..**

Inclua aspas (“) seguida do símbolo abre-parênteses “(“.

Chame Tratar_Condicao_Relacional, informando o valor do segundo elemento filho.

senão (AND ou OR)

Chame Tratar_Condicao_Logica, informando o valor do primeiro elemento filho.

Inclua o símbolo correspondente ao elemento informado (ver tabela de símbolos), seguido de aspas (“) e dois pontinhos (..), como no exemplo abaixo.

<valor>) **AND “..**

Inclua aspas (“) seguida do símbolo abre-parênteses “(“.

Se o segundo elemento filho for um dos seguintes elementos: equal, different, greater, greater_equal, lesser ou lesser_equal

então

Chame Tratar_Condicao_Relacional, informando o valor desse segundo elemento filho.

senão

Chame Tratar_Condicao_Logica, informando o valor desse segundo elemento filho.

fim-Se.

fim-Se.

Tabela de Símbolos:

Elemento	Símbolo
AND	AND
OR	OR
equal	=
different	<>
greater	>
greater_equal	>=
lesser	<
lesser_equal	<=
plus	+
minus	-
mult	*
div	/

Especificação da folha de estilo nav_indexes.xsl

A folha de estilo *nav_indexes.xsl* é responsável pela descrição do conteúdo da tabela Lua nav_indexes, que representa as estruturas de acesso do projeto de navegação de uma aplicação hipermídia. As estruturas de acesso (índices) são especificadas no documento OOHDM-ML utilizando-se os elementos *index* e/ou *hierarch_index*. A especificação desta folha de estilo é a seguinte:

Para todos os elementos *index* e *hierarch_index* encontrados no documento XML associado a folha de estilos faça

Inclua na estrutura Lua nav_indexes o valor do atributo name seguido dos caracteres “=” e “{“ da seguinte forma:

<valor_do_atributo_name> =
{

Se existe um elemento *element_context*, filho do elemento que está sendo tratado, e o elemento *navigational_context* informado nele possui mais de um elemento *order*

então

Chame Determinar_Tipo_Indice, informando 'SM' (caso o elemento que está sendo tratado seja index) ou 'HM' (caso seja hierarch_index)

senão

Chame Determinar_Tipo_Indice, informando 'S' (caso o elemento seja index) ou 'H' (caso seja hierarch_index)

fim-Se.

Se existe algum elemento nav_parameter descendente de um elemento selection, filho do elemento *index* ou *hierarch_index* que está sendo tratado, ou filho do elemento navigational_context informado no atributo context elemento element_context (filho do index ou hierarch_index)

então

Chame Tratar_Elementos_Nav_Parameter (Especificação nav_contexts.doc)

senão

Se existe um elemento element_context especificado e o valor do atributo type do elemento navigational_context informado nele é igual a "by_query" (contexto por consulta)

então

Inclui a string 'parameters = {"sql_statement"},'

fim-Se.

fim-Se.

Inclua na tabela Lua nav_indexes a string 'elements = { '

Se o elemento opcional element_context, filho do elemento que está sendo tratado, foi especificado no documento XML associado

então

Inclua, após a string 'elements = {', a string 'group = {'.

Inclua a string 'context = "' seguida pelo valor do atributo name do elemento navigational_context, cujo atributo id tem o mesmo valor do atributo context do elemento element_context encontrado acima.

Inclua aspas (") seguida de vírgula.

Se existe algum elemento order_element para o elemento que está sendo tratado

então

Chame Tratar_Elemento_Order_Element (Especificação nav_contexts.doc), informando o elemento order_element.

fim-Se.

Chame Criar_Estruturas_Attributes_Selectors

Inclua o símbolo }, fechando o campo group

senão

Para cada valor existente no atributo element_class do elemento que está sendo tratado faça

Inclua a string 'group' concatenada com o valor do atributo name do element_class que está sendo tratado e com o símbolo '='

Inclua o símbolo abre-chaves '{'.

Inclua a string 'class = ' seguida pelo valor do atributo name do elemento navigational_class, cujo atributo id tem valor igual ao valor em questão, entre aspas (") e seguido de vírgula.

Se existe algum elemento *selection*, cujo atributo selected_element_class tem valor igual ao valor do atributo element_class que está sendo tratado OU se existe apenas um elemento selection sem valor especificado para o atributo selected_element_class

então

Inclua na estrutura nav_indexes a string 'condition = "('.

Se o elemento filho do elemento selection em questão for um dos seguintes elementos: equal, different, greater, greater_equal, lesser ou lesser_equal

então

Chame Tratar_Condicao_Relacional (Especificação nav_contexts.doc), informando o elemento filho.

senão

Chame Tratar_Condicao_Logica (Especificação nav_contexts.doc), informando o elemento filho.

fim-se.

Inclua o símbolo fecha-parenteses ‘)’ seguido de aspas (“).

fim-Se.

Se existe algum elemento *order_element*, cujo atributo navigational_class tem valor igual ao valor do atributo element_class que está sendo tratado

então

Chame Tratar_Elemento_Order_Element (Especificação nav_contexts.doc), informando o elemento order_element cujo atributo navigational_class tem valor igual ao valor do atributo element_class que está sendo tratado.

fim-Se.

Chame Criar_Estruturas_Attributes_Selectors.

Se o valor é o último valor existente em element_class

então

Inclua o símbolo }.

senão

Inclua o símbolo } seguido de vírgula.

fim-Se

fim-Para.

fim-Se.

Inclua o símbolo }, fechando o campo elements.

Inclua o símbolo }, fechando a estrutura do índice em questão

Se o elemento index (ou hierarch_index) não é o último elemento tratado

então

Inclua uma vírgula.

fim-Se.

fim-para.

Determinar_Tipo_Indice:

Se o valor do atributo type do elemento que está sendo tratado for igual a ***static***

então

Inclua na estrutura nav_indexes a string ‘type = ‘ seguida pelo caracter ‘E’ concatenado com o valor informado (‘S’,‘SM’, ‘H’ ou ‘HM’), entre aspas e seguido de vírgula como mostrado abaixo:

type = “ESM”,

senão

Se o valor do atributo type do elemento que está sendo tratado for igual a ***dynamic***

então

Inclua na estrutura nav_indexes a string ‘type = ‘ seguida pelo caracter ‘D’ concatenado com o valor informado (‘S’,‘SM’, ‘H’ ou ‘HM’), entre aspas e seguido de vírgula.

senão (**temporary**)

Inclua na estrutura nav_indexes a string ‘type = ‘ seguida pelo caracter ‘T’ concatenado com o valor informado (‘S’,‘SM’, ‘H’ ou ‘HM’), entre aspas e seguido de vírgula

fim-se.

fim-Se.

Criar_Estruturas_Attributes_Selectors:

Inclua na estrutura nav_indexes a string ‘attributes = {‘

Se o elemento que está sendo tratado é um elemento *index*

então

Chame Incluir_Atributos informando os elementos shown_attrib (descendentes do elemento index) e os elementos static_selector (filhos do elemento index).

senão (*hierarch_index*)

Para todos os elementos *level* filhos do elemento *hierarch_index* faça

Inclua o simbolo ‘{‘.

Chame Incluir_Atributos informando os elementos shown_attrib (**descendentes** do elemento level) e os elementos static_selector (**filhos** do elemento level).

Se o elemento level em questão é o último elemento tratado

então

Inclua o simbolo ‘}‘,.

senão

Inclua uma vírgula.

fim-Se.

fim-Para.

fim-Se.

Inclua na estrutura nav_indexes a string ‘selectors’ seguida pelos símbolos “=” e “{“ da seguinte forma:

selectors =
{

Para todos os elementos *shown_attrib* filhos dos elementos *dynamic_selector* existentes para o elemento *index* ou *level* faça

Inclua na estrutura nav_indexes o valor do atributo name seguido pelos símbolos “=” e “{“.

Inclua a string ‘*destination* = “’ seguida pelo valor do atributo name do elemento (navigational_context, index ou hierarch_index), cujo atributo id tem o mesmo valor do atributo target do elemento selector_destination.

Se o elemento navigational_context informado no atributo context do elemento element_context possui algum elemento nav_parameter descendente de um elemento selection

então

Se o elemento navigational_context informado no atributo target do elemento selector_destination for igual ao elemento informado no atributo context do elemento element_context **ou** o índice que está sendo tratado é um índice hierárquico

então

Chama o template Incluir_Valor_Elemento_Nav_Parameter.

fim-Se.

senão

Se o elemento informado no atributo target é um elemento navigational_context e o valor do seu atributo type é igual a “*by_query*”

então

Inclua uma vírgula e a string ‘#sql_statement’

fim-Se.

fim-Se.

Inclua aspas (“) e uma vírgula.

Chame Incluir_Campo_Dest_Page.

Chame Incluir_Campo_Inst, informando o valor do atributo id do índice

Inclua o símbolo “}”

Se o elemento em questão não é o último elemento tratado

então

Inclua uma vírgula.

fim-Se.

fim-Para.

Para todos os elementos static_selector, filhos do elemento que está sendo tratado faça

Inclua na estrutura nav_indexes o valor de seu elemento filho name seguido pelos símbolos “=” e “{“.

Inclua a string ‘*anchor* = ‘ seguida pelo valor de seu elemento filho name entre aspas (“) e seguido de vírgula.

Inclua a string ‘*destination* = ’ seguida pelo valor do atributo name do elemento (navigational_context, index ou hierarch_index), cujo atributo id tem o mesmo valor do atributo target do elemento selector_destination, entre aspas (“) e seguido de vírgula.

Chame Incluir_Campo_Dest_Page.

Inclua o símbolo “}” .

fim-Para.

Inclua o símbolo “}” fechando a estrutura selectors.

Incluir_Atributos:

Para todos os elementos shown_attrib e static_selector informados faça

Inclua o valor do atributo name do elemento shown_attrib ou o valor do elemento name filho do elemento static_selector, entre aspas (“)

Se o elemento em questão é o último elemento tratado

então

Inclua o símbolo } seguido de vírgula.

senão

Inclua uma vírgula após o valor incluído.

fim-Se.

fim-Para.

Incluir_Valor_Elemento_Nav_Parameter:

Inclua uma vírgula.

Para cada elemento nav_parameter existente faça

Inclua o símbolo # seguido pelo valor do elemento nav_parameter

Se o elemento em questão não é o último

então

Inclua uma vírgula.

fim-Se.

fim-Para.

Incluir_Campo_Dest_Page:

Inclua a string ‘*dest_page* = ‘.

Se o elemento especificado no atributo target do elemento selector_destination, filho do elemento dynamic_selector ou static_selector é um elemento index ou hierarch_index

então

Inclua o valor do elemento interface_element (filho do elemento OOHDM-WEB), cujo valor do atributo target seja igual ao valor do atributo target do elemento selector_destination que está sendo tratado, entre aspas (“).

senão (se o elemento é um navigational_context)

Se o elemento que está sendo tratado é um dynamic_selector

então

Inclua o valor do elemento interface_element (filho do elemento OOHDM-WEB), cujo valor do atributo target seja igual ao valor do atributo source do elemento shown_attrib, entre aspas (“).

senão (o elemento é um static_selector)

Inclua o valor do elemento interface_element (filho do elemento OOHDM-WEB), cujo valor do atributo target seja igual ao valor do atributo id do elemento static_selector, entre aspas (“).

fim-se.

fim-Se.

Incluir_Campo_Inst:

Inclua a string ‘*inst ={* ’.

Se existe algum elemento map_attribute com atributo primary_key = “yes”, filho do elemento map_conceptual_class cujo atributo conceptual_class tem valor igual ao valor do atributo id do elemento conceptual_class ao qual pertence o atributo informado no elemento shown_attrib que está sendo tratado (**chave composta por mais de um atributo**)

então

Para cada elemento map_attribute existente faça

Chame Incluir_Valor_Campo_Inst, informando os valores dos atributos id e db_attribute do elemento map_attribute.

fim-Para.

senão

Chame Incluir_Valor_Campo_Inst, informando o valor do atributo name do primeiro elemento filho attrib do elemento conceptual_class ao qual pertence o atributo informado no elemento shown_attrib que está sendo tratado (**chave é o primeiro atributo da classe**), e o valor do atributo id do elemento map_attribute correspondente ao primeiro elemento filho, caso exista.

fim-Se.

Incluir_Valor_Campo_Inst:

Se o elemento index ou hierarch_index que está sendo tratado possui um elemento element_context

então

Se existe algum elemento attribute com atributo link, descendente de um elemento selection, filho do elemento navigational_context informado no atributo target do elemento element_context

então

Para cada elemento attribute encontrado faça

Se o valor do atributo navigational_class do elemento shown_attrib em questão for igual ao valor do atributo ***source_class*** do elemento link, cujo atributo id tem o mesmo valor do atributo link do elemento attribute que está sendo tratado

então

Se o valor do atributo id do elemento map_attribute foi informado

então

Recupera o valor do elemento *db_source_field*, correspondente ao elo informado no atributo link do elemento attribute, cujo atributo map_attribute tem valor igual ao informado para o atributo id do elemento map_attribute.

senão

Recupera o valor do elemento *db_source_field* correspondente ao elo informado no atributo link do elemento attribute.

fim-Se.

senão

Se o valor do atributo navigational_class do elemento shown_attrib em questão for igual ao valor do atributo ***target_class*** do elemento link, cujo atributo id tem o mesmo valor do atributo link do elemento attribute que está sendo tratado

então

Se o valor do atributo id do elemento map_attribute foi informado

então

Recupera o valor do elemento *db_destination_field*, correspondente ao elo informado no atributo link do elemento attribute, cujo atributo map_attribute tem valor igual ao informado para o atributo id do elemento map_attribute.

senão

Recupera o valor do elemento *db_destination_field* correspondente ao elo informado no atributo link do elemento attribute.

fim-Se.

fim-Se.

fim-Se.

fim-Para.

fim-Se.

fim-Se.

Se nenhum valor foi recuperado do elemento *db_source_field* ou *db_destination_field*
então

Inclua o valor do atributo db_attribute do elemento map_attribute **ou** o valor do atributo name do primeiro elemento filho attrib do elemento conceptual_class, informado na chamada desse procedimento.

fim-Se.

Apêndice IV

Código Fonte do OOHDM-Translation

Como a folha de estilo OOHDM-Translation foi criada em módulos, ou seja, é composta por outras folhas de estilo, inicialmente apresentamos o código fonte da folha de estilo principal e em seguida o código de cada folha de estilo que a compõe.

Folha de Estilo: oohdm_translation.xsl

```
<?xml version='1.0'?>
<!--========================================================================
|   Departamento de Informatica - PUC-Rio                                  |
|   Projeto: Ferramenta de Especificacao OOHDM para o ambiente             |
|            OOHDM-Web 2.0                                                 |
|   Programa: oohdm_translation.xsl                                        |
|   Funcao: Criar as estruturas das tabelas da linguagem Lua, que contem   |
|           a descricao do projeto de navegacao que sera processado pelo   |
|           OOHDM-WEB.                                                     |
|   Autor(a): Adriana Pereira de Medeiros                                  |
|   Data: 23/05/2000                                                       |
|   Versao:   1.0                                                          |
|   Sub-programas:con_classes.xsl,map_classes.xsl,con_relationships.xsl    |
|                 map_relationships.xsl,queries.xsl,map_classes_ctx.xsl    |
|   Alteracoes:                                                            |
|   Data  | Responsavel | Motivo                                           |
|==========================================================================-->
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">
  <!-- O resultado do processamento desta folha de estilos sera um arquivo
       texto -->
  <xsl:output method="text" indent="yes" omit-xml-declaration="yes"/>
  <!--=======================================================================
      Template para criar as estruturas Lua con_classes, map_classes,
      con_relationships e map_relationships, com base nos elementos do
      modelo conceitual e do mapeamento OODHM_WEB, representados,
      respectivamente, pelos elementos conceptual_model e OOHDM_WEB
      especificados no documento XML associado.
      ======================================================================-->
  <xsl:template match="conceptual_model">
    <!-- Cria a tabela Lua con_classes e processa todos os elementos
         conceptual_class filhos do elemento conceptual_model,
         especificados no documento XML associado -->
    con_classes =
    {
      <xsl:apply-templates select="conceptual_class"/>
    }
    <!-- Cria a tabela Lua map_classes e processa todos os elementos
         conceptual_class filhos do elemento conceptual_model, e os
         elementos map_conceptual_class relacionados a eles -->
```

```
  map_classes =
  {
    <xsl:apply-templates select="conceptual_class" mode="map_classes"/>
  }
  <!-- Se o elemento conceptual_model possui algum elemento relationship,
       entao crie a tabela Lua con_relationships e processe todos os
       elementos relationship filhos do elemento conceptual_model,
       especificados no documento XML associado-->
  <xsl:if test="relationship">
    con_relationships =
    {
      <xsl:apply-templates select="relationship"/>
    }
    map_relationships =
    {
      <xsl:apply-templates select="relationship"
                           mode="map_relationships"/>
    }
  </xsl:if>
</xsl:template>
<!--=========================================================================
    Template para criar as estruturas Lua queries, map_classes_ctx,
    nav_contexts e nav_indexes, com base nos elementos do modelo de
    navegacao e do mapeamento OODHM_WEB representados, respectivamente,
    pelos elementos navigational_model e OOHDM_WEB especificados no
    documento XML associado.
    =========================================================================-->
<xsl:template match="navigational_model">
  <!-- Cria a tabela Lua queries e processa todos os elementos
       navigational_class filhos do elemento navigational_model,
       especificados no documento XML associado -->
  queries =
  {
    <xsl:apply-templates select="navigational_class"/>
  }
  <!-- Se o elemento navigational_model possui algum elemento
       context_class, entao crie a tabela Lua map_classes_ctx e processe
       todos os elementos context_class filhos do elemento
       conceptual_model, especificados no documento XML associado-->
  <xsl:if test="context_class">
    map_classes_ctx =
    {
      <xsl:apply-templates select="context_class"/>
    }
  </xsl:if>
  <!-- Cria a tabela Lua nav_contexts e processa todos os elementos
       navigational_context filhos do elemento navigational_model,
       especificados no documento XML associado -->
  nav_contexts =
  {
    <xsl:apply-templates select="navigational_context"/>
  }
  <!-- Cria a tabela Lua nav_indexes e processa todos os elementos index
       e hierarch_index filhos do elemento navigational_model,
       especificados no documento XML associado -->
  nav_indexes =
```

```
    {
      <xsl:apply-templates select="index|hierarch_index"/>
    }
  </xsl:template>
  <!-- Inclui o codigo da folha de estilo con_classes.xsl para recuperar o
       conteudo da tabela lua con_classes -->
  <xsl:include href="con_classes.xsl"/>
  <!-- Inclui o codigo da folha de estilo map_classes.xsl para recuperar o
       conteudo da tabela lua map_classes -->
  <xsl:include href="map_classes.xsl"/>
  <!-- Inclui o codigo da folha de estilo con_relationships.xsl para
        recuperar o conteudo da tabela lua con_relationships -->
  <xsl:include href="con_relationships.xsl"/>
  <!-- Inclui o codigo da folha de estilo map_relationships.xsl para
        recuperar o conteudo da tabela lua map_relationships -->
  <xsl:include href="map_relationships.xsl"/>
  <!-- Inclui o codigo da folha de estilo queries.xsl para recuperar o
        conteudo da tabela lua queries -->
  <xsl:include href="queries.xsl"/>
  <!-- Inclui o codigo da folha de estilo map_classes_ctx.xsl para
        recuperar o conteudo da tabela lua map_classes_ctx -->
  <xsl:include href="map_classes_ctx.xsl"/>
  <!-- Inclui o codigo da folha de estilo nav_contexts.xsl para recuperar o
       conteudo da tabela lua nav_contexts -->
  <xsl:include href="nav_contexts.xsl"/>
  <!-- Inclui o codigo da folha de estilo nav_indexes.xsl para recuperar o
       conteudo da tabela lua nav_indexes -->
  <xsl:include href="nav_indexes.xsl"/>
  <!-- Template vazio para cancelar o funcionamento da regra de template
        default para nos textos -->
  <xsl:template match="text()"/>
</xsl:stylesheet>
```

Folha de Estilo: con_classes.xsl

```
<?xml version='1.0'?>
<!--=========================================================================
|   Departamento de Informatica - PUC-Rio                                   |
|   Projeto:  Ferramenta de Especificacao OOHDM para o ambiente             |
|             OOHDM-Web 2.0                                                 |
|   Programa: con_classes.xsl                                               |
|   Funcao:  Completar a estrutura da tabela Lua con_classes,tratando os    |
|            elementos conceptual_class e seus atributos, especificados     |
|            no documento XML associado a folha de estilo criado de         |
|                       acordo com a DTD OOHDM.                             |
|   Autor(a): Adriana Pereira de Medeiros                                   |
|   Data:   23/05/2000                                                      |
|   Versao:   1.0                                                           |
|   Alteracoes:                                                             |
```

```
|   Data  | Responsavel | Motivo                                                  |
|=========================================================================-->
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">
  <!--=========================================================================
      Template para preencher a estrutura da tabela Lua con_classes,
      utilizando os elementos filhos do elemento conceptual_model e
      navigational_model.
      ========================================================================= -->
  <xsl:template match="conceptual_class">
    <!-- Declaracao da variavel var_id_conceptual_class e atribuicao do
         valor do atributo id do elemento conceptual_class que esta sendo
         tratado -->
    <xsl:variable name="var_id_conceptual_class" select="@id"/>
    <!-- Inclui o valor do atributo name do elemento conceptual_class
         seguido pelos simbolos "=" e "{".  -->
    <xsl:value-of select="@name"/>
    <xsl:text> =
     {
    </xsl:text>
    <!-- Se existe valor especificado para o atributo superclass do
         elemento conceptual_class que esta sendo tratado, chama o template
         Tratar_Atributo_Superclass (para acrescentar a classe conceitual
         todos os atributos da superclasse) .  -->
    <xsl:if test="@superclass">
       <xsl:call-template name="Tratar_Atributo_Superclass"/>
       <!-- Se existe algum elemento attrib especificado para o elemento
            conceptual_class, entao inclui uma virgula -->
       <xsl:if test="attrib">
         <xsl:text>,
         </xsl:text>
       </xsl:if>
    </xsl:if>
    <!-- Para cada elemento attrib filho do elemento conceptual_class chama
         o template que trata seus atributos e seus elementos filhos
         perspective. -->
    <xsl:for-each select="attrib">
      <xsl:call-template name="Tratar_Elementos_Attrib"/>
      <!-- Se o elemento attrib eh o ultimo elemento filho tratado, inclui
           apenas o simbolo "}". Senao, inclui o simbolo "}" seguido de uma
           virgula. -->
      <xsl:choose>
        <xsl:when test="position()=last()">
          <xsl:text>
            }
          </xsl:text>
        </xsl:when>
        <xsl:otherwise>
          <xsl:text>
            },
          </xsl:text>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:for-each>
```

```
  <!-- Inclui o simbolo "}". Se o elemento conceptual_class nao eh o
       ultimo que esta sendo tratado, inclui uma virgula. -->
  <xsl:text>}</xsl:text>
  <xsl:if test="position()!=last()">
    <xsl:text>,
    </xsl:text>
  </xsl:if>
</xsl:template>

<!--========================================================================
    Template para tratar os valores do atributo superclass. Todos os
    atributos de cada classe conceitual informada no atributo superclass
    serao incluidos na estrutura lua da sub-classe.
    ======================================================================== -->

<xsl:template name="Tratar_Atributo_Superclass">
     <!-- Para cada elemento conceptual_class informado no atributo
          superclass -->
   <xsl:for-each select="id(@superclass)">
     <!-- Se existir um atributo superclass especificado chame o proprio
          template Tratar_Atributo_Superclass -->
     <xsl:if test="@superclass">
        <xsl:call-template name="Tratar_Atributo_Superclass"/>
        <!-- Se existe algum elemento attrib especificado para o elemento
             conceptual_class, entao inclui uma virgula -->
        <xsl:if test="attrib">
          <xsl:text>,
          </xsl:text>
        </xsl:if>
     </xsl:if>
     <!-- Para cada elemento attrib filho do elemento conceptual_class
          informado no atributo superclass, chama o template que trata
          seus atributos e seus elementos filhos perspective. -->
     <xsl:for-each select="attrib">
       <xsl:call-template name="Tratar_Elementos_Attrib"/>
        <!-- Se o elemento attrib eh o ultimo elemento filho tratado,
             inclui apenas o simbolo "}". Senao, inclui o simbolo "}"
             seguido de uma virgula. -->
       <xsl:choose>
         <xsl:when test="position()=last()">
           <xsl:text>
            }</xsl:text>
         </xsl:when>
         <xsl:otherwise>
           <xsl:text>
             },
           </xsl:text>
         </xsl:otherwise>
       </xsl:choose>
     </xsl:for-each>
   </xsl:for-each>
</xsl:template>

<!--========================================================================
    Template para tratar os atributos name, type e size dos elementos
    attrib e de seus elementos filhos perspective. Caso o elemento attrib
    tenha elementos filhos perspective os atributos name, type e size
    tratados serao os do elemento perspective e nao do elemento attrib.
```

```
    ================================================================== -->
<xsl:template name="Tratar_Elementos_Attrib">
  <!-- Declaracao da variavel var_id_conceptual_class e atribuicao do
       valor do atributo id do elemento conceptual_class que esta sendo
       tratado -->
  <xsl:variable name="var_id_conceptual_class" select="../@id"/>
  <xsl:choose>
    <!--Se o elemento attrib que esta sendo tratado possui algum elemento
        filho perspective -->
    <xsl:when test="perspective">
      <!-- entao para cada elemento perspective inclui o valor do seu
           atributo name seguido dos simbolos "=" e "{" e chama o
           template que trata atributos type e size para os atributos do
           elemento perspective passados como parametro.-->
      <xsl:for-each select="perspective">
        <xsl:value-of select="@name"/> <xsl:text>=
        {
        </xsl:text>
        <xsl:call-template name="Tratar_Atributos">
          <xsl:with-param name="par_class"
                          select="$var_id_conceptual_class"/>
          <xsl:with-param name="par_type" select="@type"/>
          <xsl:with-param name="par_size" select="@size"/>
        </xsl:call-template>
        <!-- Se o elemento perspective nao eh o ultimo tratado, inclui o
             simbolo "}" seguido de virgula.-->
        <xsl:if test="position()!=last()">
          <xsl:text>},
          </xsl:text>
        </xsl:if>
        <!-- Se for a ultima perspectiva tratada, acrescente uma linha em
             branco -->
        <xsl:if test="position()=last()">
           <xsl:text>
           </xsl:text>
        </xsl:if>
      </xsl:for-each>
    </xsl:when>
    <!-- Senao, se o elemento attrib que esta sendo tratado nao possui
         elementos perspective, inclui o valor do seu atributo name
         seguido dos simbolos "=" e "{" e chama o template que trata os
         atributos type e size para os seus atributos passados como
         parametro. -->
    <xsl:otherwise>
      <xsl:value-of select="@name"/>
      <xsl:text> =
       {
      </xsl:text>
      <xsl:call-template name="Tratar_Atributos">
         <xsl:with-param name="par_class"
                         select="$var_id_conceptual_class"/>
         <xsl:with-param name="par_type" select="@type"/>
         <xsl:with-param name="par_size" select="@size"/>
      </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
```

```
</xsl:template>

<!--=========================================================================
    Template para tratar os atributos type e size dos elementos attrib
    (filhos do conceptual_class) e dos elementos nav_attrib e
    perspective_attrib (filhos do context_class). Converte os tipos do
    attributo type especificados no documento XML para os tipos validos
    para o Banco de Dados que esta sendo usado e inclui os valores dos
    atributos type e size desses elementos na estrutura con_classes.
   ========================================================================= -->

<xsl:template name="Tratar_Atributos">
  <!-- Declaracao dos parametros par_type e par_size que recebem os
       valores dos atributos type e size que serao convertidos e
       incluidos na con_classes, e o parametro par_class que recebe o
       valor do atributo id do elemento conceptual_class ao qual
       pertencem os atributos tratados. -->
  <xsl:param name="par_type"/>
  <xsl:param name="par_size"/>
  <xsl:param name="par_class"/>
  <!-- Declaracao da variavel var_name_attrib e atribuicao do valor do
       atributo name do elemento attrib (perspective, nav_attrib ou
       perspective_attrib) que esta sendo tratado. -->
  <xsl:variable name="var_name_attrib" select="@name"/>
  <!-- Declaracao da variavel var_map_attribute para armazenar o elemento
       map_attribute, cujo atributo name tem o mesmo valor do atributo
       name do elemento attrib (perspective, nav_attrib ou
       perspective_attrib) que esta sendo tratado, e da variavel
       var_db_type para armazernar o valor do atributo db_type deste
       elemento map_attribute. Caso o atributo db_type nao tenha sido
       especificado pelo usuario no documento XML associado, sera
       atribuido a variavel o seu valor default definido na DTD
       OOHDM_WEB. -->
  <xsl:variable name="var_map_attribute"
            select="//map_conceptual_class[@conceptual_class=$par_class]/m
            ap_attribute[@name=$var_name_attrib]"/>
  <xsl:variable name="var_db_type" select="$var_map_attribute/@db_type"/>
  <!-- Se existe valor especificado para o atributo primary_key do
       elemento map_attribute correspondente ao elemento
       attrib(perspective, nav_attrib ou perspective_attrib) que esta
       sendo tratado, ou este elemento e o primeiro elemento filho de um
       elemento conceptual_class e nao e um elemento perspective,
       attribui a string "yes" a  variavel var_primary_key -->
  <xsl:variable name="var_primary_key">
     <xsl:if test="$var_map_attribute/@primary_key = 'yes' or
                  (position()=1 and name(..) = 'conceptual_class' and
                  name(.) != 'perspective')">
       <xsl:value-of select="'yes'"/>
     </xsl:if>
  </xsl:variable>
  <xsl:choose>
     <!-- Se existe um elemento map_attribute para o atributo que esta
          sendo tratado -->
     <xsl:when test="$var_map_attribute">
       <xsl:choose>
         <!-- Se o atributo db_type nao esta especificado no documento
```

```
            XML. Neste caso, a variavel var_db_type tera o valor
            default desse atributo que possui o caracter "_" -->
        <xsl:when test="contains($var_db_type, '_')">
          <!-- Chama o template que converte o tipo do atributo
               especificado no documento XML para um tipo valido para o
               Banco de Dados que sera usado. -->
          <xsl:call-template name="Converter_Tipo_BD">
             <xsl:with-param name="par_type" select="$par_type"/>
             <xsl:with-param name="par_size" select="$par_size"/>
             <xsl:with-param name="par_primary_key"
                             select="$var_primary_key"/>
          </xsl:call-template>
        </xsl:when>
        <!-- Se o atributo db_type esta especificado no documento XML
             associado -->
        <xsl:otherwise>
          <!-- Chama o template que incluira o valor do atributo db_type
               e dos parametros par_size e par_primary_key na estrutura
               con_classes -->
          <xsl:call-template name="Incluir_Estrutura_Type_Size_PrimKey">
            <xsl:with-param name="par_type" select="$var_db_type"/>
            <xsl:with-param name="par_size" select="$par_size"/>
            <xsl:with-param name="par_primary_key"
                           select="$var_primary_key"/>
          </xsl:call-template>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:when>
    <!-- Se nao existe um elemento map_attribute para o atributo que
         esta sendo tratado -->
    <xsl:otherwise>
      <!-- Chama o template que converte o tipo do atributo especificado
           no documento XML para um tipo valido para o Banco de Dados
           que sera usado. -->
      <xsl:call-template name="Converter_Tipo_BD">
         <xsl:with-param name="par_type" select="$par_type"/>
         <xsl:with-param name="par_size" select="$par_size"/>
         <xsl:with-param name="par_primary_key"
                        select="$var_primary_key"/>
      </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--====================================================================
    Template para converter o valor do atributo type especificado nos
    elementos attrib, perspective, nav_attrib e perspective attrib para
    um tipo valido para o Banco de Dados que sera usado durante o
    processamento do OOHDM-Web.
    ================================================================== -->
<xsl:template name="Converter_Tipo_BD">
  <!-- Declaracao dos parametros par_type, par_size e par_primary_key que
       recebem os valores dos atributos type, size e primary_key que
       serao tratados e incluidos na tabela con_classes.   -->
  <xsl:param name="par_type"/>
  <xsl:param name="par_size"/>
```

```
  <xsl:param name="par_primary_key"/>
  <!-- Declaracao da variavel var_db_type_default e atribuicao do valor
       default do atributo db_type definido na DTD OOHDM_WEB. -->
  <xsl:variable name="var_db_type_default"
                select="//map_conceptual_class/map_attribute/@db_type"/>
  <xsl:choose>
    <!-- Se o valor default armazenado na variavel var_db_type_default
         contem o valor do atributo type que esta sendo tratado seguido
         de ":" -->
    <xsl:when
         test="contains($var_db_type_default,concat($par_type,'_'))">
      <!-- Chama o template que incluira o valor valido para o atributo
           type e os valores dos parametros par_size e par_primary_key na
           estrutura Lua con_classes -->
      <xsl:call-template name="Incluir_Estrutura_Type_Size_PrimKey">
        <xsl:with-param name="par_type"
                        select="substring-before(substring-after
                                ($var_db_type_default,concat($par_type,'_'
                                )),'-')"/>
        <xsl:with-param name="par_size" select="$par_size"/>
        <xsl:with-param name="par_primary_key"
                        select="$par_primary_key"/>
      </xsl:call-template>
    </xsl:when>
    <!-- Se o valor default do atributo db_type nao contem o valor do
         atributo type que esta sendo tratado -->
    <xsl:otherwise>
      <!-- Chama o template que incluira o valor dos parametros par_type,
           par_size e par_primary_key na estrutura Lua con_classes -->
      <xsl:call-template name="Incluir_Estrutura_Type_Size_PrimKey">
        <xsl:with-param name="par_type" select="$par_type"/>
        <xsl:with-param name="par_size" select="$par_size"/>
        <xsl:with-param name="par_primary_key"
                        select="$par_primary_key"/>
      </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--====================================================================
    Template para incluir na estrutura Lua con_classes os valores dos
    atributos type e size dos elementos attrib, perspective, nav_attrib e
    perspective attrib e indicar quais atributos compoem a chave da
    classe que esta sendo tratada.
    ==================================================================== -->
<xsl:template name="Incluir_Estrutura_Type_Size_PrimKey">
  <!-- Declaracao dos parametros par_type, par_size e par_primary_key que
       recebem os valores dos atributos type, size e primary_key que
       serao incluidos na tabela con_classes.   -->
  <xsl:param name="par_type"/>
  <xsl:param name="par_size"/>
  <xsl:param name="par_primary_key"/>
  <!-- Inclui a string 'attrib_type =' seguida pelo valor do parametro
       par_type entre aspas (").  -->
  <xsl:text>attrib_type = "</xsl:text>
  <xsl:value-of select="$par_type"/>
```

```
    <xsl:text>"</xsl:text>
    <!-- Se existe valor para o parametro par_size, inclui uma virgula e a
         string 'attrib_size =' seguida pelo valor deste parametro. -->
    <xsl:if test="$par_size and $par_type != 'integer'">
      <xsl:text>,
      </xsl:text>
      <xsl:text>attrib_size = </xsl:text>
      <xsl:value-of select="$par_size"/>
    </xsl:if>
    <!-- Se existe valor para o parametro par_primary_key, inclui uma
         virgula e a string  'prim_key =' seguida pelo valor deste
         parametro entre aspas ("). -->
    <xsl:if test="$par_primary_key">
      <xsl:text>,
      </xsl:text>
      <xsl:text>prim_key = "</xsl:text>
      <xsl:value-of select="$par_primary_key"/>
      <xsl:text>"</xsl:text>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Folha de Estilo: map_classes.xsl

```
<?xml version='1.0'?>
<!--=========================================================================
|   Departamento de Informatica - PUC-Rio                                   |
|   Projeto: Ferramenta de Especificacao OOHDM para o ambiente              |
|            OOHDM-Web 2.0                                                  |
|   Programa: map_classes.xsl                                               |
|   Funcao:   Completar a estrutura da tabela Lua map_classes, tratando     |
|              os elementos conceptual_class, context_class e seus          |
|              atributos, especificados no documento XML associado a        |
|              folha de estilo criado de acordo com a DTD OOHDM-ML.         |
|   Autor(a): Adriana Pereira de Medeiros                                   |
|   Data:   14/08/2000                                                      |
|   Versao:   1.0                                                           |
|   Alteracoes:                                                             |
|   Data  | Responsavel | Motivo                                            |
|===========================================================================-->
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">
  <!--=======================================================================
      Template para preencher a estrutura da tabela Lua map_classes,
      utilizando os elementos conceptual_class e map_conceptual_class,
      filhos dos elementos conceptual_model e OOHDM-WEB, respectivamente.
      Para cada elemento conceptual_class existente no documento XML, deve
      existir seu mapeamento na tabela Lua map_classes.
      ======================================================================= -->
  <xsl:template match="conceptual_class" mode="map_classes">
    <!-- Declaracao da variavel var_id_conceptual_class e atribuicao do
         valor do atributo id do elemento conceptual_class que esta sendo
         tratado -->
    <xsl:variable name="var_id_conceptual_class" select="@id"/>
    <!-- Declaracao da variavel var_map_conceptual_class para armazenar o
```

```
      elemento map_conceptual_class, cujo atributo conceptual_class tem
      o mesmo valor do atributo id do element conceptual_class que esta
      sendo tratado, caso exista.  -->
  <xsl:variable name="var_map_conceptual_class"
              select="//OOHDM_WEB/map_conceptual_class[@conceptual_class=
              $var_id_conceptual_class]"/>
  <!-- Se existe um valor definido para o atributo db_class do elemento
       map_conceptual_class encontrado para o elemento conceptual_class
       que esta sendo tratado, a variavel var_db_class recebe este valor,
       senao, a variavel recebe o valor do atributo name do elemento
       conceptual_class. -->
  <xsl:variable name="var_db_class">
     <xsl:choose>
       <xsl:when test="$var_map_conceptual_class/@db_class">
           <xsl:value-of select="$var_map_conceptual_class/@db_class"/>
       </xsl:when>
       <xsl:otherwise>
          <xsl:value-of select="@name"/>
       </xsl:otherwise>
     </xsl:choose>
  </xsl:variable>
  <!-- Inclui, na tabela map_classes, o valor do atributo name do
       elemento conceptual_class seguido pelos simbolos "=" e "{" e a
       string 'db_class =' seguida pelo valor da variavel var_db_class
       definido acima.Se existe um elemento map_attribute, filho do
       elemento map_conceptual_class, cujo atributo conceptual_class tem
       o mesmo valor do atributo id do elemento conceptual_class que esta
       sendo tratado, inclui uma virgula apos o valor de var_db_class, a
       string 'attributes = {' e chame o template
       Tratar_Elementos_Map_Attribute, passando como parametro o valor do
       elemento map_conceptual_class correspondente. Inclui o simbolo
       "}". Se o elemento conceptual_class nao eh o ultimo que esta sendo
       tratado, inclui uma virgula -->
  <xsl:value-of select="@name"/> =
   {
    db_class = "<xsl:value-of select="$var_db_class"/>"
    <xsl:if test="$var_map_conceptual_class/map_attribute">,
    attributes =
    {
      <xsl:call-template name="Tratar_Elementos_Map_Attribute">
        <xsl:with-param name="par_map_class"
                        select="$var_map_conceptual_class"/>
      </xsl:call-template>
    }</xsl:if>
   }<xsl:if test="position()!=last()">,
    </xsl:if>
</xsl:template>

<!--=========================================================================
    Template para tratar os elementos map_attribute, incluindo na tabela
    Lua map_classes os valores dos seus atributos name e db_attribute.
    =========================================================================-->
<xsl:template name="Tratar_Elementos_Map_Attribute">
  <!-- Declaracao dos parametros par_class e par_map_class que recebem,
       respectivamente, o valor do atributo id do elemento
       conceptual_class e o elemento map_conceptual_class correspondente,
       cujos elementos filhos map_attribute serao tratados. -->
  <xsl:param name="par_map_class"/>
```

```
    <!-- Incluir na estrutura Lua map_classes, a string attributes = {.
         Para cada elemento map_attribute, filho do elemento
         map_conceptual_class em questao, inclua o valor do seu atributo
         name seguido pelo simbolo "=" e pelo valor do seu atributo
         db_attribute. Se o elemento map_attribute nao eh o ultimo elemento
         filho tratado, inclua uma virgula. -->
     <xsl:for-each select="$par_map_class/map_attribute">
      <xsl:value-of select="@name"/> = "<xsl:value-of
                                        select="@db_attribute"/>"<xsl:if
                                        test="position()!=last()">,
      </xsl:if>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Folha de Estilo: con_relationships.xsl

```
<?xml version='1.0'?>
<!--=========================================================================
|   Departamento de Informatica - PUC-Rio                                   |
|   Projeto: Ferramenta de Especificacao OOHDM para o ambiente              |
|           OOHDM-Web 2.0                                                   |
|   Programa:  con_relationship.xsl                                         |
|   Funcao:    Completar a estrutura da tabela Lua con_relationships,       |
|              tratando os elementos relationship e seus tributos,          |
|              especificados no documento XML associado a folha de          |
|              estilo, criado de acordo com a DTD OOHDM-ML.                 |
|   Autor(a): Adriana Pereira de Medeiros                                   |
|   Data:     30/05/2000                                                    |
|   Versao:   1.0                                                           |
|   Alteracoes:                                                             |
|   Data  | Responsavel | Motivo                                            |
|===========================================================================-->
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">

  <!--=======================================================================
      Template para preencher a estrutura da tabela Lua con_relationships,
      utilizando os elementos relationship do modelo conceitual e
      Map_relationship do elemento OOHDM_WEB
      ======================================================================= -->

  <xsl:template match="relationship">

    <!-- Declaracao da variavel var_id_relationship e atribuicao do valor
         do atributo id do elemento relationship que esta sendo tratado.
         Sera usada para encontrar o elemento map_relationship
         correspondente -->

    <xsl:variable name="var_id_relationship" select="@id"/>

    <!-- Declaracao da variavel var_atributo_bidirectional. A esta variavel
         esta sendo atribuido o valor do atributo bidirectional do elemento
         map_relationship, cujo atributo id_relationship tem o mesmo valor
         do atributo id do elemento relationship -->

    <xsl:variable name="var_atributo_bidirectional"
         select="//map_relationship[@id_relationship=$var_id_relationship]/
         @bidirectional"/>

    <!-- Inclui o valor do atributo name seguido do simbolo "{", a string
```

```
        source_class = seguida pelo valor do atributo name do elemento
        conceptual_class retornado pela funcao id(), a string
        destination_class = seguida pelo valor do atributo name do
        elemento conceptual_class tambem retornado pela funcao id(), as
        strings source_card = e dest_car =, seguidas pelos valores dos
        atributos souce_cardinality e target_cardinality, respectivamente.
        Se o atributo bidirectional estiver especificado para o
        relacionamento que esta sendo tratado em seu elemento
        map_relationship correspondente, inclui uma virgula apos o valor
        do atributo target_cardinality e a string bidirectional = seguida
        pelo valor da variavel var_atributo_bidirectional.  -->
      <xsl:value-of select="@name"/> =
      {
        source_class = "<xsl:value-of select="id(@source_class)/@name"/>",
        destination_class = "<xsl:value-of
                                  select="id(@target_class)/@name"/>",
        source_card = "<xsl:value-of select="@source_cardinality"/>",
        dest_card = "<xsl:value-of select="@target_cardinality"/>" <xsl:if
                                  test="$var_atributo_bidirectional">,
        bidirectional = <xsl:value-of
                        select="$var_atributo_bidirectional"/> </xsl:if>
                        }<xsl:if test="position()!=last()">, <!-- se o
                        elemento nao e o ultimo elemento encontrado, inclua
                        uma virgula apos a } -->
      </xsl:if>
  </xsl:template>
<xsl:stylesheet>
```

Folha de Estilo: map_relationships.xsl

```
<?xml version='1.0'?>
<!--=========================================================================
|   Departamento de Informatica - PUC-Rio                                   |
|   Projeto:  Ferramenta de Especificacao OOHDM para o ambiente             |
|                       OOHDM-Web 2.0                                       |
|   Programa: map_relationships.xsl                                         |
|   Funcao: Completar a estrutura da tabela Lua map_relationships,          |
|           tratando os elementos map_relationships, seus elementos         |
|           filhos e seus atributos  especificados no documento XML         |
|           associado a folha de estilo criado de acordo com a DTD OOHDM |
|   Autor(a): Adriana Pereira de Medeiros                                   |
|   Data:   15/08/2000                                                      |
|   Versao:   1.0                                                           |
|   Alteracoes:                                                             |
|   Data  | Responsavel | Motivo                                            |
|===========================================================================-->
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">

  <!--=======================================================================
      Template para preencher a estrutura da tabela Lua map_relationships,
      utilizando os elementos relationship e map_relationship, filhos dos
      elementos conceptual_model e OOHDM-WEB, respectivamente. Para cada
      elemento relationship existente no documento XML, deve existir seu
      mapeamento na tabela Lua map_relationships.
      ======================================================================= -->

  <xsl:template match="relationship" mode="map_relationships">
```

```
<!-- Declaracao das variaveis var_id_relationship, var_source_class,
     var_target_class, var_db_source_field e var_db_destination_field
     que recebem, respectivamente, e os valores dos atributos id,
     source_class e target_class do elemento relationship que esta
     sendo tratado, o  elemento db_source_field e o elemento
     db_destination_field -->
<xsl:variable name="var_id_relationship" select="@id"/>
<xsl:variable name="var_source_class" select="@source_class"/>
<xsl:variable name="var_target_class" select="@target_class"/>
<xsl:variable name="var_db_source_field"
              select="//map_relationship[@relationship=
              $var_id_relationship]/db_source_field"/>
<xsl:variable name="var_db_destination_field"
              select="//map_relationship[@relationship=
              $var_id_relationship]/db_destination_field"/>
<!-- Inclui o valor do atributo name do elemento relationship seguido
     pelos simbolos "=" e "{". -->
<xsl:value-of select="@name"/> =
{
<!-- Se existe algum elemento db_source_field, filho do elemento
     map_relationship, cujo atributo relationship tem o mesmo valor do
     atributo id do elemento relationship que esta sendo tratado,
     inclui a string 'db_source_field = {'. -->
<xsl:if test="$var_db_source_field">
  <xsl:text>db_source_field =
  {
  </xsl:text>
  <!-- Para cada elemento db_source_field existente, verifica se existe
       valor especificado para o seu atributo map_attribute. Se existe,
       inclui o valor do atributo name do elemento map_attribute, cujo
       atributo id tem o mesmo valor do atributo map_attribute, seguido
       pelo simbolo "=" e pelo valor do elemento db_source_field.
       Senao, inclui o valor do atributo name do primeiro elemento
       attrib, filho do elemento conceptual_class, cujo atributo id tem
       o mesmo valor do atributo source_class do elemento relationship
       que esta sendo tratado. -->
  <xsl:for-each select="$var_db_source_field">
     <xsl:choose>
       <xsl:when test="@map_attribute">
          <xsl:variable name="var_map_attribute"
                        select="@map_attribute"/>
          <xsl:value-of select="//map_attribute[@id =
                        $var_map_attribute]/@name"/>
          <xsl:text>=</xsl:text> "<xsl:value-of select="text()"/>"
       </xsl:when>
       <xsl:otherwise>
          <xsl:value-of select="//conceptual_class[@id =
                 $var_source_class]/attrib[position() = 1]/@name"/>
          <xsl:text>=</xsl:text> "<xsl:value-of select="text()"/>"
       </xsl:otherwise>
     </xsl:choose>
     <!-- Se o elemento db_source_field nao e o ultimo tratado, inclui
          uma virgula; se for o ultimo inclui o simbolo } seguido de
          virgula. -->
     <xsl:choose>
       <xsl:when test="position()!= last()">
          <xsl:text>,</xsl:text>
```

```
      </xsl:when>
      <xsl:otherwise>
         <xsl:text>},
         </xsl:text>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:for-each>
</xsl:if>
<!-- Se existe algum elemento db_destination_field, filho do elemento
     map_relationship, cujo atributo relationship tem o mesmo valor do
     atributo id do elemento relationship que esta sendo tratado,
     inclui a string   'db_source_field = {'. -->
<xsl:if test="$var_db_destination_field">
  <xsl:text>db_destination_field =
  {
  </xsl:text>
  <!-- Para cada elemento db_destination_field existente, verifica se
       existe valor especificado para o seu atributo map_attribute. Se
       existe, inclui o valor do atributo name do elemento
       map_attribute, cujo atributo id tem o mesmo valor do atributo
       map_attribute, seguido pelo simbolo "=" e pelo valor do elemento
       db_destination_field. Senao, inclui o valor do atributo name do
       primeiro elemento attrib, filho do elemento
       conceptual_class, cujo atributo id tem o mesmo valor do atributo
       trget_class do elemento relationship
       que esta sendo tratado. -->
  <xsl:for-each select="$var_db_destination_field">
    <xsl:choose>
      <xsl:when test="@map_attribute">
         <xsl:variable name="var_map_attribute"
                       select="@map_attribute"/>
         <xsl:value-of select="//map_attribute[@id =
                       $var_map_attribute]/@name"/>
         <xsl:text>=</xsl:text> "<xsl:value-of select="text()"/>"
      </xsl:when>
      <xsl:otherwise>
         <xsl:value-of select="//conceptual_class[@id =
               $var_target_class]/attrib[position() = 1]/@name"/>
         <xsl:text>=</xsl:text> "<xsl:value-of select="text()"/>"
      </xsl:otherwise>
    </xsl:choose>
    <!-- Se o elemento db_destination_field nao e o ultimo tratado,
         inclui uma virgula; se for o ultimo inclui o simbolo }
         seguido de virgula. -->
    <xsl:choose>
      <xsl:when test="position()!= last()">
         <xsl:text>,</xsl:text>
      </xsl:when>
      <xsl:otherwise>
         <xsl:text>},
         </xsl:text>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:for-each>
</xsl:if>
<!-- Inclui a string 'db_relationship = ' seguida pelo valor do
```

```
        atributo db_relationship do elemento map_relationship, cujo
        atributo relationship tem o mesmo valor do atributo id do elemento
        relationship que esta sendo tratado -->
    <xsl:text>db_relationship = "</xsl:text>
    <xsl:value-of select="//map_relationship[@relationship=
                  $var_id_relationship]/@db_relationship"/>
    <xsl:text>"
    </xsl:text>
    <!--Inclui o simbolo "}". Se o elemento relationship nao e o ultimo que
        esta sendo tratado, inclui uma virgula-->
    <xsl:text>}</xsl:text>
    <xsl:if test="position()!=last()">
      <xsl:text>,
      </xsl:text>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Folha de Estilo: queries.xsl

```
<?xml version='1.0'?>
<!--=========================================================================
|   Departamento de Informatica - PUC-Rio                                   |
|   Projeto:  Ferramenta de Especificacao OOHDM para o ambiente             |
|                      OOHDM-Web 2.0                                        |
|   Programa: queries.xsl                                                   |
|   Funcao: Completar a estrutura da tabela Lua queries, tratando os        |
|          elementos navigational_class,context_class e seus atributos,     |
|           especificados no documento XML associado a folha de estilo      |
|           criado de acordo com a DTD OOHDM-ML.                            |
|   Autor(a): Adriana Pereira de Medeiros                                   |
|   Data:   16/08/2000                                                      |
|   Versao:   1.0                                                           |
|   Sub-Programas: tratar_atributos.xsl                                     |
|   Alteracoes:                                                             |
|   Data  | Responsavel | Motivo                                            |
|===========================================================================-->
<xsl:stylesheet
     version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:cd="http://www.adriana/saxon/ClassesDistintas">

  <!--=======================================================================
      Template para preencher a estrutura da tabela Lua queries (que
      representa o mapeamento das classes de navegacao em tabelas de BD),
      utilizando os elementos filhos do elemento navigational_model.
      ======================================================================= -->

  <xsl:template match="navigational_class">

    <!--Declaracao da variavel var_id_navigational_class e atribuicao do
        valor do atributo id do elemento navigational_class que esta sendo
        tratado -->

    <xsl:variable name="var_id_navigational_class" select="@id"/>

    <!-- Funcao de extensao criada para executar metodos da classe Java
         chamada "ClassesDistintas" identificada pelo namespace "cd" criado
         na declaracao desta folha de estilos. Esta funcao cria uma
         instancia da classe "ClassesDistintas" atraves da palavra
```

```
     reservada new e para cada elemento nav_attrib ou
     perspective_attrib, filho do elemento navigational_class que esta
     sendo tratado, executa o metodo "distinguirClasses" passando como
     parametro a instancia criada, o valor do atributo conceptual_class
     do elemento nav_attrib ou perpective_attrib e o valor do atributo
     name do elemento conceptual_class, cujo atributo id tem o mesmo
     valor do atributo conceptual_class considerado. O metodo
     "distinguirClasses" insere os pares distintos "conceptual_class,
     name" em uma tabela Hash. Depois a funcao executa o metodo
     "obterClassesNames",armazenando na variavel var_classes os valores
     distintos do atributo name incluidos na HashTable. -->

<xsl:variable name="instance" select="cd:new()"/>
<xsl:for-each select="nav_attrib|perspective_attrib">
  <xsl:variable name="var_result"
                select="cd:distinguirClasses($instance,
                @conceptual_class,id(@conceptual_class)/@name)"/>
</xsl:for-each>
<xsl:variable name="var_classes"
              select="cd:obterClassesNames($instance)"/>

<!-- Inclui o valor do atributo name do elemento navigational_class
     seguido pelos simbolos "=" e "{".  -->

<xsl:value-of select="@name"/> =
{
<!-- Inclui a string 'class =' seguida pelos valores do atributo name
     dos elementos conceptual_class, armazenados na variavel
     var_classes, entre chaves {} e seguido de virgula. Em seguida,
     verifica se existe uma virgula na string armazenada na variavel
     var_classes. Se existir, os elementos nav_attrib e
     perspective_attrib possuem valores distintos para seus atributos
     conceptual_class(tabelas distintas), entao, chama o template
     Tratar_Tabelas_Distintas para criar as estruturas tables e fields,
     passando como parametro o valor da variavel var_classes que contem
     as classes distintas, e o template Criar_Estrutura_Where para
     criar a estrutura where. Senao, se os elementos possuem valores
     iguais para seus atributos conceptual_class(tabela unica), entao
     chama o template Tratar_Tabela_Unica para criar as estruturas
     tables e fields passando como parametro a    variavel var_classes
     contendo apenas uma classe e o valor do atributo conceptual_class
     do primeiro elemento nav_attrib, filho do elemento
     navigational_class que esta sendo tratado. -->

class = {<xsl:value-of select="$var_classes"/>},
<xsl:choose>
  <xsl:when test="substring-after($var_classes,',')">
    <xsl:call-template name="Tratar_Tabelas_Distintas">
      <xsl:with-param name="par_classes" select="$var_classes"/>
    </xsl:call-template>
  </xsl:when>
  <xsl:otherwise>
    <xsl:call-template name="Tratar_Tabela_Unica">
      <xsl:with-param name="par_conceptual_class"
                      select="$var_classes"/>
      <xsl:with-param name="par_id_class"
                      select="nav_attrib[1]/@conceptual_class"/>
    </xsl:call-template>
  </xsl:otherwise>
</xsl:choose>

<!-- Se existe um elemento context_class no documento XML associado,
```

```
      cujo valor do atributo base_class seja igual ao valor do atributo
      id do elemento navigational_class que esta sendo tratado, inclui
      uma virgula apos o ultimo valor incluido na tabela lua queries e
      chama o template Tratar_Elementos_Context_Class, passando como
      parametro o valor do atributo id do elemento context_class em
      questao. -->
  <xsl:if test="//context_class[@base_class=$var_id_navigational_class]">
    <xsl:text>,</xsl:text>
    <xsl:call-template name="Tratar_Elementos_Context_Class">
      <xsl:with-param name="par_id_context_class"
                      select="//context_class[@base_class=
                      $var_id_navigational_class]/@id"/>
    </xsl:call-template>
  </xsl:if>
  <!-- Inclui o simbolo "}". Se o elemento navigational_class nao e o
       ultimo tratado inclui uma virgula apos o simbolo "}" incluido.-->
  } <xsl:if test="position()!=last()">,
    </xsl:if>
</xsl:template>

<!--=========================================================================
    Template para criar as estruturas tables, fields e where da tabela
    Lua queries, utilizando o elementos nav_attrib e perspective_attrib,
    filhos do elemento navigational_class, que tem como "origem" classes
    conceituais DISTINTAS, ou seja, os elementos tem valores distintos
    para o atributo conceptual_class.
    ========================================================================= -->
<xsl:template name="Tratar_Tabelas_Distintas">
   <!-- Declaracao do parametro par_classes que recebe os nomes das
        classes conceituais, armazenados na variavel var_classes, que
        serao usados para a criacao da estrutura tables e fields na
        tabela Lua queries -->
  <xsl:param name="par_classes"/>
 <!--Inclui na estrutura Lua queries a string 'tables = {'. Para cada
     elemento map_conceptual_class filho do elemento OOHDM_WEB existente
     no documento XML associado, verifica se o valor do atributo name do
     elemento conceptual_class, cujo atributo id tem o mesmo valor do
     atributo conceptual_class do elemento map_conceptual_class em
     questao, faz parte das classes conceituais passadas como parametro.
     Se este valor faz parte, entao, se existe valor especificado para o
     atributo db_class do elemento map_conceptual_class em questao,
     inclui este valor na estrutura tables, senao, inclui o valor do
     atributo name do elemento conceptual_class correspondente. Em ambos
     os casos, se o valor que esta sendo inserido nao pertence ao ultimo
     elemento map_concepual_class tratado, inclui uma virgula apos o
     valor. Para cada elemento relationship existente no documento XML,
     se os valores de seus atributos source_class e target_class fazem
     parte das classes conceituais passadas como parametro, e os valores
     dos atributos source_cardinality e target_cardinality sao iguais a
     "n", entao, inclui na estrutura tables o valor do atributo
     db_relationship do elemento map_relationship, cujo atributo
     relationship tem o mesmo valor do atributo id do elemento
     relationship que esta sendo tratado. Chama os templates
     Criar_Estrutura_Fields e Criar_Estrutura_Where para preencher as
     estruturas lua fields e where, respectivamente. -->
  tables =
  {
  <xsl:for-each select="//OOHDM_WEB/map_conceptual_class">
```

```
    <xsl:if test="contains($par_classes,id(@conceptual_class)/@name)">
      <xsl:choose>
        <xsl:when test="@db_class">"<xsl:value-of
                  select="@db_class"/>"<xsl:if
                  test="position()!=last()">,</xsl:if>
        </xsl:when>
        <xsl:otherwise>"<xsl:value-of
                       select="id(@conceptual_class)/@name"/>"<xsl:if
                       test="position()!=last()">,</xsl:if>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:if>
  </xsl:for-each>
  <xsl:for-each select="//relationship">
    <xsl:if test="(contains($par_classes,id(@source_class)/@name) and
                   contains($par_classes,id(@target_class)/@name)) and
                   @source_cardinality = 'n'and @target_cardinality='n'">
      <xsl:variable name="var_id_relationship" select="@id"/>
      ,"<xsl:value-of
             select="//map_relationship[@relationship=
             $var_id_relationship]/@db_relationship"/>"
    </xsl:if>
  </xsl:for-each>
  },
  <xsl:call-template name="Criar_Estrutura_Fields"/>
   <xsl:text>,</xsl:text>
  <xsl:call-template name="Criar_Estrutura_Where">
    <xsl:with-param name="par_classes" select="$par_classes"/>
  </xsl:call-template>
</xsl:template>

<!--=========================================================================
    Template para criar as estruturas tables e fields da tabela Lua
    queries, utilizando os elementos nav_attrib e perspective_attrib,
    filhos do elemento navigational_class, que tem como "origem" uma
    unica classe conceitual, ou seja, todos eles tem o mesmo valor para o
    atributo conceptual_class.
    ========================================================================= -->

<xsl:template name="Tratar_Tabela_Unica">

  <!--Declaracao dos parametros par_conceptual_class e par_id_class que
      recebem, respectivamente, o nome da classe conceitual armazenado na
      variavel var_classes e o valor do atributo conceptual_class, que
      serao usados para a criacao da estrutura tables e fields na tabela
      Lua queries.-->

  <xsl:param name="par_conceptual_class"/>
  <xsl:param name="par_id_class"/>

  <!--Inclui na estrutura Lua queries a string 'tables =". Se existe
      algum elemento map_conceptual_class,cujo valor do atributo
      conceptual_class seja igual ao valor recebido no parametro
      par_id_class, verifica se existe valor especificado para o atributo
      db_class do elemento map_conceptual_class. Se existir, inclui o
      valor deste atributo entre aspas e delimitado por chaves {} apos a
      string 'tables =". Senao, inclui o nome da classe conceitual
      delimitado por chaves {} armazenado no parametro
      par_conceptual_class. Se NAO existe elemento map_conceputal_class,
      inclui o nome da classe conceitual delimitado por chaves {}
      armazenado no parametro par_conceptual_class. Chama o template
      Criar_Estrutura_Fields para criar a estrutura lua fields. -->
```

```
  <xsl:variable name="var_path_class"
              select="//OOHDM_WEB/map_conceptual_class[@conceptual_class=
              $par_id_class]"/>
  tables =
  <xsl:choose>
    <xsl:when test="$var_path_class">
      <xsl:choose>
        <xsl:when test="$var_path_class/@db_class">
          {"<xsl:value-of select="$var_path_class/@db_class"/>"},
        </xsl:when>
        <xsl:otherwise>
          {<xsl:value-of select="$par_conceptual_class"/>},
        </xsl:otherwise>
      </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      {<xsl:value-of select="$par_conceptual_class"/>},
    </xsl:otherwise>
  </xsl:choose>
  <xsl:call-template name="Criar_Estrutura_Fields"></xsl:call-template>
</xsl:template>

<!--========================================================================
    Template para criar a estrutura fields da tabela Lua queries,
    utilizando os elementos nav_attrib e perspective_attrib, filhos do
    elemento navigational_class.
    ======================================================================== -->

<xsl:template name="Criar_Estrutura_Fields">
  <!-- Inclui na estrutura Lua queries a string 'fields = {'. Para cada
       elemento nav_attrib ou perspective_attrib, filho do elemento
       navigational_class que esta sendo tratado, verifica se existe um
       elemento map_attribute, cujo atributo name tem o mesmo valor do
       atributo name do elemento nav_attrib ou perspective_attrib que
       esta sendo tratado, e tenha como elemento pai um elemento
       map_conceptual_class, cujo atributo id tem o mesmo valor do que o
       seu atributo conceptual_class. Se existir, chama o template
       Incluir_Campos_Fields passando como parametro o valor do atributo
       db_attribute correspondente, juntamente com os valores das
       variaveis var_path_class e var_id_class que serao utilizadas para
       identificar a classe conceitual a qual pertence o elemento
       nav_attrib ou perspective_attrib tratado. Senao existir elementos
       map_attribute, chama o template Incluir_Campos_Fields passando
       como parametro o valor do atributo name do proprio elemento
       nav_attrib ou perspective attrib que esta sendo tratado. -->
  fields =
  {
  <xsl:for-each select="nav_attrib|perspective_attrib">
    <xsl:variable name="var_name_attrib" select="@name"/>
    <xsl:variable name="var_id_class" select="@conceptual_class"/>
    <xsl:variable name="var_path_class"
       select="//map_conceptual_class[@conceptual_class=$var_id_class]"/>
    <xsl:variable name="var_db_attribute"
           select="$var_path_class/map_attribute[@name=$var_name_attrib]/
           @db_attribute"/>
    <xsl:variable name="var_primary_key"
           select="$var_path_class/map_attribute[@name=$var_name_attrib]/
           @primary_key"/>
    <xsl:choose>
      <xsl:when test="$var_db_attribute">
```

```
        <xsl:if test="$var_primary_key = 'no'">
          <xsl:call-template name="Incluir_Campos_Fields">
            <xsl:with-param name="par_path_class"
                            select="$var_path_class"/>
            <xsl:with-param name="par_id_class" select="$var_id_class"/>
            <xsl:with-param name="par_attribute"
                            select="$var_db_attribute"/>
          </xsl:call-template>
        </xsl:if>
      </xsl:when>
      <xsl:otherwise>
        <xsl:if test="position() != 1">
          <xsl:call-template name="Incluir_Campos_Fields">
            <xsl:with-param name="par_path_class"
                            select="$var_path_class"/>
            <xsl:with-param name="par_id_class" select="$var_id_class"/>
            <xsl:with-param name="par_attribute"
                            select="$var_name_attrib"/>
          </xsl:call-template>
        </xsl:if>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:for-each>
  }
</xsl:template>

<!--=====================================================================
    Template para incluir, na estrutura fields da tabela Lua queries, os
    valores dos atributos db_attribute e name dos elementos nav_attrib,
    perspective_attrib ou map_attribute, respeitando o formato
    <nome_tabela>.<nome_attributo>.
    ===================================================================== -->

<xsl:template name="Incluir_Campos_Fields">
  <!--Declaracao dos parametros par_path_class, par_id_class e
      par_attribute que recebem, respectivamente, o caminho para
      identificar se existe um elemento map_conceptual_class
      correspondente, o valor do atributo conceptual_class do elemento
      nav_attrib ou perspective_attrib e o valor do seu atributo name
      caso nao exista um elemento map_attribute correspondente, se
      existir, par_attribute recebera o valor do atributo db_attribute.
      -->
  <xsl:param name="par_path_class"/>
  <xsl:param name="par_id_class"/>
  <xsl:param name="par_attribute"/>
  <!-- Se existir um elemento map_conceptual_class, cujo atributo
       conceptual_class tem o mesmo valor do atributo conceptual_class do
       elemento nav_attrib ou perspective_attrib que esta sendo tratado,
       e seu atributo db_class tem um valor especificado, entao inclui na
       estrutura fields o valor deste atributo seguido de ponto (.) e do
       valor do parametro par_attribute. Senao, inclui o valor do
       atributo name do elemento conceptual_class, cujo atributo id tem o
       mesmo valor do parametro par_id_class, seguido de ponto(.) e do
       valor do parametro par_attribute.-->
  <xsl:choose>
    <xsl:when test="$par_path_class/@db_class">
      "<xsl:value-of select="$par_path_class/@db_class"/>.
       <xsl:value-of select="$par_attribute"/>"<xsl:if
       test="position()!=last()">,</xsl:if>
    </xsl:when>
```

```
    <xsl:otherwise>
      "<xsl:value-of select="id($par_id_class)/@name"/>.
       <xsl:value-of select="$par_attribute"/>"<xsl:if
       test="position()!=last()">,</xsl:if>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--========================================================================
    Template para criar a estrutura where da tabela Lua queries,
    utilizando os elementos relationship, map_relationship,
    map_conceptual_class e map_attribute.
    ======================================================================= -->

<xsl:template name="Criar_Estrutura_Where">
  <!--Declaracao do parametro par_classes que recebe os nomes das
      classes conceituais distintas, armazenados na variavel var_classes,
      que serao usados para a criacao da estrutura where na tabela Lua
      queries. -->
  <xsl:param name="par_classes"/>
  <!-- Inclui na tabela lua queries a string 'where = "'. Em seguida,
       para cada elemento relationship, declara quatro variaveis e
       atribui a elas, os valores dos seus atributos id, source_class,
       target_class e o caminho relativo para se obter o elemento
       map_relationship correspondente. As atribuicoes dos valores de
       atributo a variaveis sao necessarias para que esses valores possam
       ser usados em caminhos relativos, uma vez que nao e possivel usar
       atributos de elementos diferentes em um mesmo caminho.-->
  <xsl:text>where = </xsl:text>
  <xsl:for-each select="//relationship">
    <xsl:variable name="var_id_relationship" select="@id"/>
    <xsl:variable name="var_id_source_class" select="@source_class"/>
    <xsl:variable name="var_id_target_class" select="@target_class"/>
    <xsl:variable name="var_path_relationship"
                  select="//OOHDM_WEB/map_relationship[@relationship=
                  $var_id_relationship]"/>
    <!-- Se os valores dos atributos name dos elementos conceptual_class,
         cujos atributos id tem os mesmos valores dos atributos
         source_class e target_class do elemento relationship que esta
         sendo tratado, fazem parte dos nomes das classes distintas
         armazenados no parametro par_classes -->
    <xsl:if test="contains($par_classes,id(@source_class)/@name) and
                  contains($par_classes,id(@target_class)/@name)">
      <!--Entao determina os valores para as variaveis
          var_target_class_bd e var_source_class_bd, que armazenarao os
          nomes das tabelas de banco de dados para cada uma das classes
          consideradas no relacionamento, de acordo com o resultado da
          execucao do template Determinar_Class_Bd. -->
      <xsl:variable name="var_target_class_bd">
        <xsl:call-template name="Determinar_Class_Bd">
          <xsl:with-param name="par_id_class"
                          select="$var_id_target_class"/>
        </xsl:call-template>
      </xsl:variable>
      <xsl:variable name="var_source_class_bd">
        <xsl:call-template name="Determinar_Class_Bd">
          <xsl:with-param name="par_id_class"
                          select="$var_id_source_class"/>
```

```
  </xsl:call-template>
</xsl:variable>
<!--Se os valores dos atributos source_cardinality e
    target_cardinality do elemento relationship sao ambos iguais a
    'n' (relacionamento n pra n), entao chama o template
    Incluir_Condicao_Where duas vezes, incluindo uma string 'AND'
    entre os resultados das duas execucoes. Na primeira chamada ao
    template, sao passados os parametros necessarios para se obter
    a condicao com o formato:
    <valor_target_class_bd>.<valor_atributo_chave> =
    <valor_db_relationship>.<valor_db_destination_field> e na
    segunda chamada sao passados os parametros para se obter a
    condicao com o formato:
    <valor_source_class_bd>.<valor_atributo_chave> =
    <valor_db_relationship>.<valor_db_source_field>. Senao
    (relacionamentos 1 pra 1 ou 1 pra n), verifica se o valor do
    atributo db_relationship do elemento map_relationship, cujo
    atributo relationship tem o mesmo valor do atributo id do
    elemento relationship que esta sendo tratado
    (var_path_relationship),e igual ao valor armazenado na variavel
    var_target_class_bd (nome da tabela no BD), se for chama o
    template Incluir_Condicao_Where passando os parametros
    necessarios para obter a condicao com o formato:
    <valor_source_class_bd>.<valor_atributo_chave> =
    <valor_db_relationship>.<valor_db_source_field>, senao, se o
    valor for diferente chama o template Incluir_Condicao_Where
    passando os parametros necessarios para obter a condicao com o
    formato: <valor_target_class_bd>.<valor_atributo_chave> =
    <valor_db_relationship>.<valor_db_destination_field>. O valor
    para o parametro par_flag so sera informado quando for
    necessario utilizar o valor do elemento db_destination_field
    para criar a estrutura where. Por fim, se elemento relationship
    nao for o ultimo tratado, entao inclui uma string 'AND'. Quando
    todos os elementos relationship tiverem sido considerados
    inclui uma aspas (") no final da estrutura where.-->
<xsl:choose>
  <xsl:when test="@source_cardinality = 'n' and
                  @target_cardinality = 'n'">
    <xsl:text>"</xsl:text>
    <xsl:call-template name="Incluir_Condicao_Where">
      <xsl:with-param name="par_class_bd"
                      select="$var_target_class_bd"/>
      <xsl:with-param name="par_flag" select="'target'"/>
      <xsl:with-param name="par_id_class"
                    select="$var_id_target_class"/>
      <xsl:with-param name="par_path_relationship"
                      select="$var_path_relationship"/>
        <xsl:with-param name="par_path_class"
                      select="//OOHDM_WEB/map_conceptual_class[@con
                      ceptual_class=$var_id_target_class]"/>
    </xsl:call-template><xsl:text> AND </xsl:text>
    <xsl:call-template name="Incluir_Condicao_Where">
      <xsl:with-param name="par_class_bd"
                      select="$var_source_class_bd"/>
      <xsl:with-param name="par_id_class"
                      select="$var_id_source_class"/>
      <xsl:with-param name="par_path_relationship"
                      select="$var_path_relationship"/>
      <xsl:with-param name="par_path_class"
```

```
                            select="//OOHDM_WEB/map_conceptual_class[
                            @conceptual_class=$var_id_source_class]"/>
          </xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
          <xsl:choose>
            <xsl:when test="$var_path_relationship/@db_relationship =
                            $var_target_class_bd">
              <xsl:text>"</xsl:text>
              <xsl:call-template name="Incluir_Condicao_Where">
                <xsl:with-param name="par_class_bd"
                                select="$var_source_class_bd"/>
                <xsl:with-param name="par_id_class"
                                select="$var_id_source_class"/>
                <xsl:with-param name="par_path_relationship"
                               select="$var_path_relationship"/>
                <xsl:with-param name="par_path_class"
                               select="//OOHDM_WEB/map_conceptual_class[
                               @conceptual_class=$var_id_source_class]"/
                               >
              </xsl:call-template>
            </xsl:when>
            <xsl:otherwise>
              <xsl:text>"</xsl:text>
              <xsl:call-template name="Incluir_Condicao_Where">
                <xsl:with-param name="par_class_bd"
                                select="$var_target_class_bd"/>
                <xsl:with-param name="par_flag"     select="'source'"/>
                <xsl:with-param name="par_id_class"
                                select="$var_id_target_class"/>
                <xsl:with-param name="par_path_relationship"
                                select="$var_path_relationship"/>
                <xsl:with-param name="par_path_class"
                                select="//OOHDM_WEB/map_conceptual_class[
                                @conceptual_class=$var_id_target_class]"/
                                >
              </xsl:call-template>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:if>
    <xsl:if test="position()!=last()"> AND "..
    </xsl:if>
  </xsl:for-each>
  <xsl:text>"
  </xsl:text>
</xsl:template>

<!--========================================================================
    Template para determinar o nome da classe no Banco de Dados (tabela)
    que sera considerado na estrutura where da tabela Lua queries.
    ======================================================================== -->

<xsl:template name="Determinar_Class_Bd">
  <!--Declaracao do parametro par_id_class que recebe o valor do
      atributo target_class ou source_class do elemento relationship que
      esta sendo tratado. -->
  <xsl:param name="par_id_class"/>
  <xsl:variable name="var_path_class"
```

```
                select="//OOHDM_WEB/map_conceptual_class[
                @conceptual_class=$par_id_class]"/>
  <!-- Se existe um elemento map_conceptual_class, cujo atributo
       conceptual_class e igual ao valor do atributo id armazenado no
       parametro par_id_class, entao verifica se seu atributo db_class
       possui algum valor. Se possui entao retorne este valor para a
       variavel var_target_class_bd (ou var_source_class_bd). Se nao
       existe valor para o atributo db_class ou nao existe elemento
       map_conceptual_class, entao retorne o valor do atributo name do
       elemento conceptual_class, cujo atributo id tem valor igual ao
       armazenado no parametro par_id_class.  -->
  <xsl:choose>
    <xsl:when test="$var_path_class">
      <xsl:choose>
        <xsl:when test="$var_path_class/@db_class">
          <xsl:value-of select="$var_path_class/@db_class"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="id($par_id_class)/@name"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="id($par_id_class)/@name"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--=========================================================================
    Template para incluir, na estrutura where da tabela Lua queries, os
    valores definitivos para criar a condicao que vai determinar os
    elementos do no de navegacao.
    ========================================================================= -->

<xsl:template name="Incluir_Condicao_Where">
  <!-- Declaracao dos parametros: par_class_bd que recebe o nome da
       tabela do banco de dados que representa uma das classes no
       relacionamento; par_flag que recebe como valor a string 'source'
       ou 'target'; par_id_class que recebe como valor o id da classe
       conceitual considerada no relacionamento; par_path_relationship
       que recebe como  valor o caminho relativo para verificar a
       existencia de um elemento map_relationship correspondente e
       par_path_class que recebe como valor o caminho relativo para
       verificar a existencia de um elemento map_conceptual_class
       correspondente a classe conceitual considerada no relacionamento.
       -->
  <xsl:param name="par_class_bd"/>
  <xsl:param name="par_flag"/>
  <xsl:param name="par_id_class"/>
  <xsl:param name="par_path_relationship"/>
  <xsl:param name="par_path_class"/>
  <!-- Se existe pelo menos um elemento map_attribute cujo valor do
       atributo primary_key seja igual a 'yes', entao para todos os
       elementos map_attribute com atributo primary_key = 'yes', inclui
       na estrutura where, o nome da tabela no banco de dados seguido
       pelo valor do seu atributo db_attribute, respeitando o formato
       '<nome_tabela>.<valor_atributo> = '. Se existe valor para o
       parametro par_flag, entao inclui o valor do atributo
       db_relationship do elemento map_relationship, correspondente ao
```

```
     elemento relationship que esta sendo tratado, seguido pelo valor
     do elemento db_destination_field, cujo atributo attribute tem o
     mesmo valor do atributo id do elemento map_attribute considerado.
     Se nao existe valor para o parametro par_flag, entao considere o
     valor do elemento db_source_field, respeitando o formato:
     '<nome_relacionamento_bd>.<valor_db_source_field>'. Se o elemento
     map_attribute nao e o ultimo tratado, inclui uma string AND'. Se
     nao existe elemento map_attribute com atributo primary_key = yes',
     entao inclui na estrutura where, o nome da tabela no banco de
     dados seguido pelo valor do atributo name do elemento
     conceptual_class, cujo atributo id tem valor igual ao do parametro
     par_id_class, respeitando o formato <nome_tabela>.<valor_atributo>
     = '. Se existe valor para o parametro par_flag, entao inclui o
     valor do atributo db_relationship, seguido pelo valor do elemento
     db_destination_field. Caso contrario, considere o valor elemento
     db_source_field, respeitando o formato:
     '<nome_relacionamento_bd>.<valor_db_source_field>'. -->
<xsl:choose>
  <xsl:when test="$par_path_class/map_attribute[@primary_key = 'yes']">
    <xsl:for-each select="$par_path_class/map_attribute[
                          @primary_key = 'yes']">
      <xsl:variable name="var_id_attrib" select="@id"/>
      <xsl:value-of select="$par_class_bd"/>.<xsl:value-of
                                    select="@db_attribute"/> =
      <xsl:choose>
        <xsl:when test="$par_flag">
          <xsl:value-of
                    select="$par_path_relationship/@db_relationship"/>.
          <xsl:value-of
                    select="$par_path_relationship/db_destination_field
                    [@map_attribute=$var_id_attrib]"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of
                    select="$par_path_relationship/@db_relationship"/>.
          <xsl:value-of
                    select="$par_path_relationship/db_source_field[
                    @map_attribute=$var_id_attrib]"/>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:if test="position()!=last()"> AND "..
         <xsl:text>"</xsl:text>
      </xsl:if>
    </xsl:for-each>
  </xsl:when>
  <xsl:otherwise>
    <xsl:value-of select="$par_class_bd"/>.<xsl:value-of
         select="id($par_id_class)/attrib[1]/@name"/> = <xsl:choose>
      <xsl:when test="$par_flag">
        <xsl:value-of
             select="$par_path_relationship/@db_relationship"/>.
        <xsl:value-of
             select="$par_path_relationship/db_destination_field"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of
             select="$par_path_relationship/@db_relationship"/>.
          <xsl:value-of
             select="$par_path_relationship/db_source_field"/>
```

```
          </xsl:otherwise>
        </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<!--========================================================================
    Template para tratar os elementos context_class que representam as
    classes em contexto no documento XML. Cria a estrutura class_ctx,
    composta pelo valor do atributo name de cada elemento
    navigational_context (nome de cada contexto no qual a classe
    conceitual possui classe em contexto), os valores dos atributos name,
    type e size de cada elemento attrib e perspective_attrib do elemento
    context_class.
    ======================================================================== -->
<xsl:template name="Tratar_Elementos_Context_Class">
  <!-- Declaracao do parametro par_id_context_class que armazena o valor
       o atributo id do elemento context_class, caso ele exista.-->
  <xsl:param name="par_id_context_class"/>
  <!-- Declaracao da variavel var_name_perspective_attrib e atribuicao do
       valor do atributo name de um elemento perspective_attrib filho do
       elemento context_class, caso exista algum. -->
  <xsl:variable name="var_name_perspective_attrib"
           select="id($par_id_context_class)/perspective_attrib/@name"/>
  <!-- Cria a estrutura class_ctx, incluindo a string "class_ctx ="
       seguida do simbolo "{" -->
  class_ctx =
  {
  <!-- Para cada elemento navigational_context especificado no atributo
       contexts do elemento context_class, cujo valor do atributo id eh o
       mesmo do parametro par_id_context_class, inclui o valor do
       atributo name seguido dos simbolos "=" e "{". -->
  <xsl:for-each
       select="id(//context_class[@id=$par_id_context_class]/@contexts)">
    <xsl:value-of select="@name"/> =
    {
    <!--Para cada elemento nav_attrib filho do elemento context_class,
        cujo valor do atributo id eh o mesmo do parametro
        par_id_context_class, armazena o valor de seu atributo name na
        variavel var_name_nav_attrib, inclui este valor seguido dos
        simbolos "=" e "{" na estrutura class_ctx e chama o template que
        trata atributos type e size para os atributos do elemento attrib,
        cujo atributo name tem o mesmo valor do atributo name do elemento
        nav_attrib, passados como parametro. -->
    <xsl:for-each select="id($par_id_context_class)/nav_attrib">
      <xsl:variable name="var_name_nav_attrib" select="@name"/>
      <xsl:value-of select="@name"/> =
      {
      <xsl:call-template name="Tratar_Atributos">
        <xsl:with-param name="par_type"
                        select="id(@conceptual_class)/attrib[@name=
                        $var_name_nav_attrib]/@type"/>
        <xsl:with-param name="par_size"
                        select="id(@conceptual_class)/attrib[@name=
                        $var_name_nav_attrib]/@size"/>
      </xsl:call-template>
        <!--Se o elemento nav_attrib eh o ultimo tratado e o elemento
```

```
            context_class nao tem elementos filhos perspective_attrib,
            inclui o simbolo "}", senao inclui este simbolo seguido de
            virgula. -->
        <xsl:when test="position()=last() and
                  not($var_name_perspective_attrib)">
        }
        </xsl:when>
        <xsl:otherwise>
        },
        </xsl:otherwise>
      </xsl:choose>
    </xsl:for-each>
    <!--Para cada elemento perspective_attrib filho do elemento
        context_class, identificado pelo parametro par_id_context_class,
        armazena os valores de seus atributos name e conceptual_attrib
        nas variaveis var_name_perspective e var_name_attrib,
        respectivamente. Inclui o valor do atributo name seguido pelos
        simbolos "=" e "{" na estrutura class_ctx e chama o template que
        trata atributos type e size para os atributos do elemento
        perspective_attrib passados como parametro. -->
    <xsl:for-each select="id($par_id_context_class)/perspective_attrib">
      <xsl:variable name="var_name_perspective" select="@perspective"/>
      <xsl:variable name="var_name_attrib" select="@conceptual_attrib"/>
      <xsl:value-of select="@name"/> =
      {
        <xsl:call-template name="Tratar_Atributos">
          <!--estes parametros recebem os valores dos atributos type e
              size do elemento perspective(filho do elemento attrib),
              cujo atributo name tem valor igual ao da variavel
              var_name_perspective. -->
          <xsl:with-param name="par_type"
                  select="id(@conceptual_class)/attrib[@name=$var_name_att
                  rib]/perspective[@name=$var_name_perspective]/@type"/>
          <xsl:with-param name="par_size"
                  select="id(@conceptual_class)/attrib[@name=$var_name_att
                  rib]/perspective[@name=$var_name_perspective]/@size"/>
        </xsl:call-template>
        <xsl:choose>
          <!--Se o elemento perspective_attrib eh o ultimo tratado e o
              elemento context_class nao tem elementos filhos
              perspective_attrib, inclui o simbolo "}", senao inclui este
              simbolo seguido de virgula. -->
          <xsl:when test="position()=last()">
          }
          </xsl:when>
          <xsl:otherwise>
          },
          </xsl:otherwise>
        </xsl:choose>
    </xsl:for-each>
    <!-- Inclui o simbolo "}".Se o elemento navigational_context nao eh o
         ultimo tratado, inclui uma virgula -->
    }<xsl:if test="position()!=last()">,
     </xsl:if>
  </xsl:for-each>
  }  <!-- inclui um simbolo "}" para completar a estrutura class_ctx. -->
</xsl:template>
```

```
</xsl:stylesheet>
```

Folha de Estilo: map_classes_ctx.xsl

```
<?xml version='1.0'?>
<!--=========================================================================
|   Departamento de Informatica - PUC-Rio                                   |
|   Projeto: Ferramenta de Especificacao OOHDM para o ambiente              |
|               OOHDM-Web 2.0                                               |
|   Programa: map_classes_ctx.xsl                                           |
|   Funcao: Completar a estrutura da tabela Lua map_classes_ctx,            |
|           tratando os elementos navigational_class, context_class e       |
|           seus atributos, especificados no documento XML associado a      |
|           folha de estilo criado de acordo com a DTD OOHDM.               |
|   Autor(a): Adriana Pereira de Medeiros                                   |
|   Data:   29/09/2000                                                      |
|   Versao:   1.0                                                           |
|   Sub-programas: template Tratar_Elementos_Map_Attribute - definido na    |
|                  folha de estilos map_classes.xsl                         |
|   Alteracoes:                                                             |
|   Data  | Responsavel | Motivo                                            |
|===========================================================================-->
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">

  <!--=======================================================================
      Template para preencher a estrutura da tabela Lua map_classes_ctx,
      utilizando os elementos context_class e map_context_class, filhos dos
      elementos navigational_model e OOHDM-WEB, respectivamente. Para cada
      elemento context_class existente no documento XML, deve existir seu
      mapeamento na tabela Lua map_classes_ctx.
      ======================================================================= -->

  <xsl:template match="context_class">

    <!--Declaracao da variavel var_id_context_class e atribuicao do valor
        do atributo id do elemento context_class que esta sendo tratado-->

    <xsl:variable name="var_id_context_class" select="@id"/>

    <!--Declaracao das variaveis var_map_context_class e var_db_class_ctx
        para armazenar, respectivamente, o elemento map_context_class filho
        do elemento OOHDM_WEB, cujo atributo context_class tem o mesmo
        valor do atributo id do elemento context_class que esta sendo
        tratado, caso exista, e o valor de seu atributo db_class_ctx.-->

    <xsl:variable name="var_map_context_class"
                  select="//OOHDM_WEB/map_context_class[@context_class=
                  $var_id_context_class]"/>

    <xsl:variable name="var_db_class_ctx"
                  select="$var_map_context_class/@db_class_ctx"/>

    <!-- Inclui na tabela map_classes_ctx o valor do atributo name do
         elemento navigational_class, cujo atributo id tem o mesmo valor do
         atributo base_class do elemento context_class que esta sendo
         tratado,  seguido pelos simbolos "=" e "{". Inclui a string
         'db_class_ctx =', seguida pelo valor da variavel var_db_class_ctx
         definida acima. Se existe um elemento map_attribute, filho do
         elemento map_context_class armazenado na variavel
         var_map_context_class, inclui uma virgula apos o valor de
         var_db_class_ctx, a string 'attrib_class_ctx = {' e chame o
         template Tratar_Elementos_Map_Attribute, passando como parametro a
```

```
        variavel var_map_context_class. Inclui o simbolo "}". Se o
        elemento context_class nao eh o ultimo que esta sendo tratado,
        inclui uma virgula -->
    <xsl:value-of select="id(@base_class)/@name"/> =
     {
      db_class_ctx = "<xsl:value-of select="$var_db_class_ctx"/>"
      <xsl:if test="$var_map_context_class/map_attribute">,
      attrib_class_ctx =
      {
        <xsl:call-template name="Tratar_Elementos_Map_Attribute">
          <xsl:with-param name="par_map_class"
                          select="$var_map_context_class"/>
        </xsl:call-template>
        </xsl:if>
      }
     }<xsl:if test="position()!=last()">,
      </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Folha de Estilo: nav_contexts.xsl

```
<?xml version='1.0'?>
<!--=========================================================================
|   Departamento de Informatica - PUC-Rio                                   |
|   Projeto:  Ferramenta de Especificacao OOHDM para o ambiente             |
|             OOHDM-Web 2.0                                                 |
|   Programa: nav_contexts.xsl                                              |
|   Funcao: Completar a estrutura da tabela Lua nav_contexts,tratando os    |
|           elementos navigational_context e seus atributos,                |
|           especificados no documento XML associado a folha de estilo      |
|           criado de acordo com a DTD OOHDM-ML.                            |
|  Autor(a): Adriana Pereira de Medeiros                                    |
|   Data:   04/10/2000                                                      |
|   Versao:   1.0                                                           |
|   Sub-Programas:                                                          |
|   Alteracoes:                                                             |
|   Data  | Responsavel  | Motivo                                           |
|===========================================================================-->
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">

  <!--=======================================================================
      Template para preencher a estrutura da tabela Lua nav_contexts,
      utilizando os elementos flhos do elemento navigational_model.
      ======================================================================= -->

  <xsl:template match="navigational_context">

    <!-- Declaracao da variavel var_id_context e atribuicao do valor do
         atributo id do elemento navigational_context que esta sendo
         tratado -->

    <xsl:variable name="var_id_context" select="@id"/>

    <!--Declaracao da variavel var_path_nav_context e atribuicao do
        caminho relativo que identifica o elemento navigational_context que
        esta sendo tratado, quando houver alteracao do no de contexto. -->

    <xsl:variable name="var_path_nav_context"
                  select="//navigational_context[@id = $var_id_context]"/>
```

```
<!-- Declaracao da variavel var_tipo_contexto e atribuicao de seu valor
     de acordo com a seguinte condicao: se existe mais de um elemento
     order filho do elemento navigational_context, chama o template
     Determinar_Tipo_Contexto passando como parametro o caracter "M"
     (indicando que o contexto possui multipla ordenacao); senao, chama
     este mesmo template sem informar valor para o parametro. -->
<xsl:variable name="var_tipo_contexto">
  <xsl:choose>
     <xsl:when test="count(order) > 1">
        <xsl:call-template name="Determinar_Tipo_Contexto">
          <xsl:with-param name="par_ordenacao" select="'M'"/>
        </xsl:call-template>
     </xsl:when>
     <xsl:otherwise>
        <xsl:call-template name="Determinar_Tipo_Contexto"/>
     </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
<!--Declaracao da variavel var_tipo_navegacao e atribuicao de seu
    valor de acordo com a seguinte condicao: se o valor do atributo
    navigation_type do elemento navigational_context for igual a
    'sequential', atribui o caracter "S"; se for 'circular' atribui o
    caracter "C"; se for 'index' atribui o caracter "I"; se for 'free'
    atribui o caracter "L"; se for 'sequential_index' atribui os
    caracteres "SI" e se for 'circular_index'atribui os caracteres
    "CI". -->
<xsl:variable name="var_tipo_navegacao">
  <xsl:choose>
     <xsl:when test="@navigation_type = 'sequential'">
        <xsl:value-of select="'S'"/>
     </xsl:when>
     <xsl:when test="@navigation_type = 'circular'">
        <xsl:value-of select="'C'"/>
     </xsl:when>
     <xsl:when test="@navigation_type = 'index'">
        <xsl:value-of select="'I'"/>
     </xsl:when>
     <xsl:when test="@navigation_type = 'free'">
        <xsl:value-of select="'L'"/>
     </xsl:when>
     <xsl:when test="@navigation_type = 'sequential_index'">
        <xsl:value-of select="'SI'"/>
     </xsl:when>
     <xsl:when test="@navigation_type = 'circular_index'">
        <xsl:value-of select="'CI'"/>
     </xsl:when>
  </xsl:choose>
</xsl:variable>
<!-- Inclui o valor do atributo name do elemento navigational_context
     seguido pelos simbolos "=" e "{"; a string 'context = ' seguida
     pelo tipo do contexto armazenado na variavel var_tipo_contexto
     entre aspas,      seguido de virgula e a string 'nav_type = '
     seguida pelo tipo de navegacao no contexto armazenado na variavel
     var_tipo_navegacao entre aspas e seguido de virgula. Se existe
     algum elemento nav_parameter descendente do elemento selection,
     filho do navigational_context que esta sendo tratado,  entao chame
     o template Tratar_Elementos_Nav_Parameter passando como parametro
     o elemento nav_parameter. Senao, se o valor do atributo type do
```

```
     elemento navigational_context que esta sendo tratado e igual a
     'by_query', entao inclui a string 'parameters =
     {"sql_statement'},'. -->
<xsl:value-of select="@name"/> =
{
 context = "<xsl:value-of select="$var_tipo_contexto"/>",
 nav_type = "<xsl:value-of select="$var_tipo_navegacao"/>",
<xsl:choose>
  <xsl:when test="$var_path_nav_context/selection//nav_parameter">
    <xsl:call-template name="Tratar_Elementos_Nav_Parameter">
       <xsl:with-param name="par_nav_parameter"
             select="$var_path_nav_context/selection//nav_parameter"/>
    </xsl:call-template>
  </xsl:when>
  <xsl:otherwise>
    <xsl:if test="@type = 'by_query'">
       <xsl:text>parameters = {"sql_statement"},</xsl:text>
    </xsl:if>
  </xsl:otherwise>
</xsl:choose>
<!-- Declaracao da variavel var_qtd_element_class e atribuicao de seu
     valor que sera igual a quantidade de valores existentes no
     atributo element_class do elemento navigational_context em
     questao. O atributo element_class e do tipo IDREFS e contem o id
     dos nos que compoem o contexto -->
<xsl:variable name="var_qtd_element_class"
              select="count(id(@element_class))"/>
<!-- Inclui na estrutura nav_contexts a string 'elements = {'. Para
     cada valor (id) existente no atributo element_class, inclui o
     valor do seu atributo name seguido pelos simbolos '= {'. Se existe
     algum elemento order_element para o contexto, chama o template
     Tratar_Elementos_Order_Element passando como parametro o elemento
     order_element cujo atributo navigational_class tem valor igual ao
     do element_class (caso exista mais de um valor no atributo
     element_class), ou apenas o elemento order_element (caso so exista
     um valor em element_class). Inclui a string 'dest_page = ' seguida
     pelo valor do elemento interface_element (filho do elemento OOHDM-
     WEB), cujo atributo element_class tem o mesmo valor do atributo
     element_class em questao. -->
elements =
{
<xsl:for-each select="id(@element_class)">
  <xsl:variable name="var_element_class" select="@id"/>
  <xsl:value-of select="@name"/> =
  {
  <xsl:if test="$var_path_nav_context/order/order_element">
    <xsl:choose>
      <xsl:when test="$var_qtd_element_class > 1">
        <xsl:call-template name="Tratar_Elementos_Order_Element">
         <xsl:with-param name="par_order_element"
                    select="$var_path_nav_context/order/order_element[
                    @navigational_class = $var_element_class]"/>
        </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:call-template name="Tratar_Elementos_Order_Element">
          <xsl:with-param name="par_order_element"
               select="$var_path_nav_context/order/order_element"/>
```

```
        </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
</xsl:if>
<xsl:text>dest_page = "</xsl:text>
<xsl:value-of select="//OOHDM_WEB/interface_element[contains(@target,
                    $var_element_class)]"/>
<xsl:text>"</xsl:text>
<!--Verifica se existe algum elemento selection filho do elemento
    navigational_context. Para cada elemento selection existente, se o
    valor do atributo selected_element_class e igual ao valor do
    element_class ou se nao existe valor especificado para ele, entao,
    inclui a string 'selection = {'. Se existe valor especificado para
    o atributo link de algum elemento attribute filho do elemento
    selection em questao, entao inclui a           string "link_name =
    {'. Para cada elemento attribute filho do elemento selection que
    possui o atributo link especificado, inclui o valor do atributo
    name do elemento relationship, cujo atributo id tem valor igual ao
    do atributo link, entre aspas. Inclui a string 'condition = "('. Se
    o elemento filho do elemento selection e um elemento equal,
    different, etc., entao chama o template Tratar_Condicao_Relacional
    passando como parametro todos os elementos filhos do elemento
    selection (pela DTD havera apenas um). Senao, (se o elemento filho
    for AND ou OR), chama o template Tratar_Condicao_Logica passando
    como parametro todos os elementos filhos do selection. Inclui o
    simbolo ')' seguido de aspas. Se o atributo link foi especificado,
    entao inclui o simbolo '}'. Inclui outro simbolo '}' fechando a
    estrutura selection. -->
<xsl:choose>
  <xsl:when test="$var_path_nav_context/selection">
    <xsl:for-each select="$var_path_nav_context/selection">
      <xsl:if test="@selected_element_class = $var_element_class or
                    not(@selected_element_class)">
        <xsl:text>,
        selection =
        {
        </xsl:text>
        <xsl:if test=".//attribute[@link]">
          <xsl:text>link_name = {</xsl:text>
          <xsl:for-each select=".//attribute[@link]">
             <xsl:text>"</xsl:text><xsl:value-of
                    select="id(./@link)/@name"/><xsl:text>"</xsl:text>
             <xsl:if test="position()!=last()">
                      <xsl:text>,</xsl:text>
           </xsl:if>
         </xsl:for-each>
            <xsl:text>},
            </xsl:text>
        </xsl:if>
        <xsl:text>condition = "(</xsl:text>
        <xsl:choose>
          <xsl:when test="equal or different or greater or
                          greater_equal or lesser or lesser_equal">
             <xsl:call-template name="Tratar_Condicao_Relacional">
                <xsl:with-param name="par_path_operador_relacional"
                          select="$var_path_nav_context/selection/*"/>
             </xsl:call-template>
          </xsl:when>
          <xsl:otherwise>
```

```
            <xsl:call-template name="Tratar_Condicao_Logica">
               <xsl:with-param name="par_path_operador_logico"
                        select="$var_path_nav_context/selection/*"/>
            </xsl:call-template>
          </xsl:otherwise>
        </xsl:choose>
        <xsl:text>)"
        </xsl:text>
      </xsl:if>
    </xsl:for-each>
    }
  </xsl:when>
  <!--Se nao existe elemento selection para o contexto, verifica se
      existe algum elemento instance filho do elemento
      navigational_context, cujo atributo navigational_class e igual ao
      element_class em questao. Se existe (contexto arbitrario), entao
      chama o template Incluir_Condicao_Contexto_Arbitrario, passando
      como parametros o caminho que identifica o contexto que esta
      sendo tratado e o valor atual de seu atributo element_class.
      Senao, se o valor do atributo type do contexto e "by_query"
      (contexto por consulta), entao inclui a string 'selection = {.
      Inclui a string 'condition = "(#sql_statement)". Inclui tres
      simbolos '}'fechando, respectivamente, as estruturas  selection,
      element_class e elements. -->
  <xsl:otherwise>
    <xsl:choose>
       <xsl:when test="$var_path_nav_context/instance[
                       @navigational_class = $var_element_class]">
         <xsl:text>,</xsl:text>
         <xsl:call-template
                   name="Incluir_Condicao_Contexto_Arbitrario">
            <xsl:with-param name="par_path_nav_context"
                 select="$var_path_nav_context"/>
            <xsl:with-param name="par_element_class"
                 select="$var_element_class"/>
         </xsl:call-template>
       </xsl:when>
       <xsl:when test="$var_path_nav_context/@type = 'by_query'">
         <xsl:text>,
         selection =
         {
          condition = "(#sql_statement)"
         }
         </xsl:text>
       </xsl:when>
    </xsl:choose>
  </xsl:otherwise>
</xsl:choose>
<xsl:text>
}</xsl:text>
<xsl:if test="position()!=last()">
    <xsl:text>,
    </xsl:text>
  </xsl:if>
</xsl:for-each>
}
<!-- Inclui o simbolo "}". Se o elemento navigational_context nao eh o
     ultimo que esta sendo tratado, inclui uma virgula. -->
```

```
  }<xsl:if test="position()!=last()">,
   </xsl:if>
</xsl:template>

<!--=========================================================================
    Template para determinar o tipo do contexto que sera incluido na
    estrutura nav_contexts.
    Resumo: Este tipo sera determinado com base no atributo type do
    elemento navigational_context, que indica o tipo do contexto que esta
    sendo tratado (estatico, dinamico, temporario ou por consulta), e nos
    elementos nav_parameter, selection, value e instance, que indicam se
    o contexto forma um grupo de contexto ou se ele e um contexto
    arbitrario (instance). O tipo do contexto deve ser determinado de
    acordo com as combinacoes estabelecidas no OOHDM-Web.
    Parametros: par_ordenacao -> tipo de ordenacao do contexto
                valores: "M" (multipla ordenacao) ou vazio
    ========================================================================= -->
<xsl:template name="Determinar_Tipo_Contexto">
  <xsl:param name="par_ordenacao"/>
  <!-- Declaracao da variavel var_type e atribuicao de seu valor de
       acordo com o valor do atributo type do elemento
       navigational_context que esta sendo tratado. Se o valor do
       atributo type  for "static", a variavel recebera o caracter "E";
       se for "dynamic", recebera o caracter "D"; se for "temporary",
       recebera o caracter "T"; senao (se for "by_query") recebera o
       caracter "DC". Esta variavel sera usada para determinar o tipo do
       contexto que sera incluido na estrutura lua nav_contexts.-->
  <xsl:variable name="var_type">
      <xsl:choose>
         <xsl:when test="@type = 'static'">
            <xsl:value-of select="'E'"/>
         </xsl:when>
         <xsl:when test="@type = 'dynamic'">
            <xsl:value-of select="'D'"/>
         </xsl:when>
         <xsl:when test="@type = 'temporary'">
            <xsl:value-of select="'T'"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:value-of select="'DC'"/>
         </xsl:otherwise>
      </xsl:choose>
  </xsl:variable>
  <!--Concatenacao dos valores da variavel var_type, do parametro
      par_ordenacao e dos caracteres "G", "A" e "S", indicando
      respectivamente um grupo de contexto (caso o elemento
      navigational_context possua um elemento  nav_parameter descendente
      de um elemento selection; um contexto arbitrario (caso exista algum
      elemento instance, filho de navigational_context); e um contexto
      simples. Caso o valor do atributo type do elemento
      navigational_context seja igual a "by_query" (contexto por
      consulta), nao havera concatenacao de caracter pois este tipo de
      contexto estara indicado na variavel var_type. O resultado dessa
      concatenacao sera o tipo   definitivo do contexto que sera
      atribuido a variavel var_tipo_contexto no template que chamou este.
      -->
    <xsl:choose>
       <xsl:when test="selection//nav_parameter">
```

```
            <xsl:value-of select="concat($var_type,'G',$par_ordenacao)"/>
         </xsl:when>
         <xsl:when test="instance">
            <xsl:value-of select="concat($var_type,'A',$par_ordenacao)"/>
         </xsl:when>
         <xsl:when test="@type = 'by_query'">
            <xsl:value-of select="concat($var_type,$par_ordenacao)"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:value-of select="concat($var_type,'S',$par_ordenacao)"/>
         </xsl:otherwise>
      </xsl:choose>
</xsl:template>

<!--========================================================================
    Template para tratar os elementos nav_parameter descendentes de um
    elemento selection, filho do elemento navigational_context.
    Resumo: Inclui a string parameters = { seguida pelo valor de cada
    elemento nav_parameter entre aspas. Se o elemento tratado nao e o
    ultimo elemento do loop inclui uma virgula.
    Parametros: par_nav_parameter -> elementos nav_parameter
                valor: nav_parameter
    ======================================================================== -->
<xsl:template name="Tratar_Elementos_Nav_Parameter">
   <xsl:param name="par_nav_parameter"/>
   <xsl:text>parameters = {</xsl:text>
   <xsl:for-each select="$par_nav_parameter">
      <xsl:text>"</xsl:text>
      <xsl:value-of select="."/><xsl:text>"</xsl:text>
      <xsl:if test="position()!=last()">
         <xsl:text>,</xsl:text>
      </xsl:if>
   </xsl:for-each>
   <xsl:text>},</xsl:text>
</xsl:template>

<!--========================================================================
    Template para tratar o elemento order_element descendente do elemento
    navigational_context.
    Resumo: Inclui na estrutura lua nav_contexts a string 'order = {'.
    Para cada elemento em par_order_element, inclui o valor do atributo
    name seguido pelo simbolo "=" e pelo valor do atributo criteria entre
    aspas. Se o elemento tratado nao e o ultimo elemento do loop, entao
    inclui uma virgula. Inclui os simbolos '},' fechando o campo order.
    Parametros: par_order_element - criterio de ordenacao para os
    elementos de um contexto.
                valor: order_element
    ======================================================================== -->
<xsl:template name="Tratar_Elementos_Order_Element">
  <xsl:param name="par_order_element"/>
  <xsl:if test="$par_order_element">
    order = {
    <xsl:for-each select="$par_order_element">
      <xsl:value-of select="@name"/> =
      "<xsl:value-of select="@criteria"/>"<xsl:if
      test="position()!=last()">,</xsl:if>
    </xsl:for-each>
    },
```

```
  </xsl:if>
</xsl:template>

<!--=========================================================================
    Template para tratar os elementos equal, different, greater,
    greater_equal, lesser e lesser_equal, filhos do elemento selection.
    Resumo: Se o primeiro elemento filho do elemento informado no
    parametro for o elemento attribute, entao inclui na estrutura lua
    nav_contexts o valor do seu atributo name seguido pelo valor da
    variavel var_simbolo. Senao, chama o template
    Tratar_Condicao_Aritmetica, passando como parametro o valor do
    primeiro elemento filho (plus, minus, mult ou div). Se o segundo
    elemento filho do elemento informado for o elemento attribute, entao
    inclui na estrutura lua nav_contexts o valor do seu atributo name; se
    for o elemento value, chama o template Incluir_Valor_Elemento_Value,
    passando como parametro os valores do primeiro e do segundo elemento
    filho do elemento informado. Senao (se o segundo elemento filho for
    plus, minus,mult ou div), chama o template Tratar_Condicao_Aritmetica
    passando como parametro o valor do segundo elemento filho.
    Parametros: par_path_operador_relacional - operador relacional dentro
    de uma selecao
         valores: equal, different, greater, greater_equal, lesser ou
                                        lesser_equal.
    ========================================================================= -->
<xsl:template name="Tratar_Condicao_Relacional">
  <xsl:param name="par_path_operador_relacional"/>
  <!-- Atribui a variavel var_simbolo o simbolo corresponde ao elemento
       informado no parametro -->
  <xsl:variable name="var_simbolo">
    <xsl:choose>
       <xsl:when test="name($par_path_operador_relacional) = 'equal'">
         <xsl:value-of select="'='"/>
      </xsl:when>
      <xsl:when test="name($par_path_operador_relacional) = 'different'">
         <xsl:value-of select="'&lt;&gt;'"/>
      </xsl:when>
      <xsl:when test="name($par_path_operador_relacional) = 'greater'">
         <xsl:value-of select="'&gt;'"/>
      </xsl:when>
      <xsl:when test="name($par_path_operador_relacional) =
                      'greater_equal'">
         <xsl:value-of select="'&gt;='"/>
      </xsl:when>
      <xsl:when test="name($par_path_operador_relacional) = 'lesser'">
         <xsl:value-of select="'&lt;'"/>
      </xsl:when>
      <xsl:when test="name($par_path_operador_relacional) =
                      'lesser_equal'">
         <xsl:value-of select="'&lt;='"/>
      </xsl:when>
   </xsl:choose>
  </xsl:variable>
  <!-- Trata o PRIMEIRO elemento filho do elemento dado no parametro -->
  <xsl:choose>
    <xsl:when test="name($par_path_operador_relacional/*[position()=1])=
                         'attribute'">
       <xsl:value-of
```

```
          select="$par_path_operador_relacional/*[position()=1]/@name"/>
          <xsl:value-of select="$var_simbolo"/>
    </xsl:when>
    <xsl:otherwise>
       <xsl:call-template name="Tratar_Condicao_Aritmetica">
          <xsl:with-param name="par_path_operador_aritmetico"
             select="$par_path_operador_relacional/*[position() = 1]"/>
       </xsl:call-template>
       <xsl:value-of select="$var_simbolo"/>
    </xsl:otherwise>
  </xsl:choose>
  <!-- Trata o SEGUNDO elemento filho do elemento dado no parametro -->
  <xsl:choose>
    <xsl:when test="name($par_path_operador_relacional/*[position()=2])=
                    'attribute'">
      <xsl:value-of
          select="$par_path_operador_relacional/*[position()=2]/@name"/>
    </xsl:when>
    <xsl:when test="name($par_path_operador_relacional/*[position()=2])=
                    'value'">
      <xsl:call-template name="Incluir_Valor_Elemento_Value">
         <xsl:with-param name="par_primeiro_filho"
              select="$par_path_operador_relacional/*[position() = 1]"/>
         <xsl:with-param name="par_segundo_filho"
              select="$par_path_operador_relacional/*[position() = 2]"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:when test="name($par_path_operador_relacional/*[position()=2])=
                    'nav_parameter'">
      <xsl:text>#</xsl:text><xsl:value-of
           select="$par_path_operador_relacional/*[position() = 2]"/>
    </xsl:when>
    <xsl:otherwise>
       <xsl:call-template name="Tratar_Condicao_Aritmetica">
          <xsl:with-param name="par_path_operador_aritmetico"
               select="$par_path_operador_relacional/*[position() = 2]"/>
       </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--=========================================================================
    Template para tratar e incluir o valor do elemento value no campo
    condition, formando a condicao para obter os elmentos de um contexto.
    Resumo: Verifica se o primeiro elemento filho do elemento selection e
    um elemento attribute e se foi possivel recuperar o tipo de seu
    atributo name. Se o tipo desse atributo e integer, real ou boolean,
    entao, inclui no campo condition o valor do elemento value; senao,
    inclui no campo condition o valor do elemento value entre aspas
    simples.Se o primeiro elemento filho nao e um elemento attribute,
    entao simplesmente inclui o valor do elemento value.
    Parametros: par_primeiro_filho - primeiro elemento filho de um
    operador relacional ou operador aritmetico
                valores: elementos attribute, plus, minus, mult ou div
                par_segundo_filho - segundo elemento filho de um elemento
                                    operador relacional ou aritmetico
                valor: value
    ========================================================================= -->
```

```
<xsl:template name="Incluir_Valor_Elemento_Value">
  <xsl:param name="par_primeiro_filho"/>
  <xsl:param name="par_segundo_filho"/>
  <!-- Declaracao da variavel var_tipo_atributo. Esta variavel recebera o
       valor retornado pelo template Determinar_Tipo_Atributo, que sera o
       tipo do atributo especificado no atributo name do elemento
       attribute (filho dos elementos equal, different, plus, minus,
       etc.)-->
  <xsl:variable name="var_tipo_atributo">
     <xsl:call-template name="Determinar_Tipo_Atributo">
       <xsl:with-param name="par_navigational_class"
                 select="$par_primeiro_filho/@navigational_class"/>
       <xsl:with-param name="par_nome_atributo"
                 select="$par_primeiro_filho/@name"/>
    </xsl:call-template>
  </xsl:variable>
  <!-- Inclui na estrutura condition o valor do elemento value -->
  <xsl:choose>
    <xsl:when test="name($par_primeiro_filho) = 'attribute'">
       <xsl:choose>
         <xsl:when test="contains('integer real boolean',
                         $var_tipo_atributo)">
            <xsl:value-of select="$par_segundo_filho"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:text>'</xsl:text>
            <xsl:value-of select="$par_segundo_filho"/>
            <xsl:text>'</xsl:text>
         </xsl:otherwise>
       </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$par_segundo_filho"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--=========================================================================
    Template para tratar os elementos plus, minus, mult e div, filhos de
    um elemento equal, different, greater, greater_equal, lesser ou
    lesser_equal.
    Resumo: Inclui na estrutura condition o valor do atributo name do
    elemento attribute (primeiro elemento filho do operador aritmetico
    informado no parametro par_path_operador_aritmetico), seguido pelo
    valor da variavel var_simbolo. Caso o segundo elemento filho do
    operador aritmetico informado no parametro seja elemento attribute,
    entao inclui o valor do seu atributo name. Se for um elemento value
    chama o template Incluir_Valor_Elemento_Value, passando como
    parametros os valores do primeiro e do segundo elemento filho do
    elemento informado. Se for um elemento nav_parameter inclui o simbolo
    "#" seguido pelo valor deste elemento.
    Parametros: par_path_operador_aritmetico - operador aritmetico dentro
    de uma expressao relacional
                valores: plus, minus, mult ou div.
    ========================================================================= -->
<xsl:template name="Tratar_Condicao_Aritmetica">
  <xsl:param name="par_path_operador_aritmetico"/>
```

```
  <!-- Atribui a variavel var_simbolo o simbolo corresponde ao elemento
       informado no parametro -->
  <xsl:variable name="var_simbolo">
    <xsl:choose>
       <xsl:when test="name($par_path_operador_aritmetico) = 'plus'">
         <xsl:value-of select="'+'"/>
      </xsl:when>
      <xsl:when test="name($par_path_operador_aritmetico) = 'minus'">
         <xsl:value-of select="'-'"/>
      </xsl:when>
      <xsl:when test="name($par_path_operador_aritmetico) = 'mult'">
         <xsl:value-of select="'*'"/>
      </xsl:when>
      <xsl:when test="name($par_path_operador_aritmetico) = 'div'">
         <xsl:value-of select="'/'"/>
      </xsl:when>
   </xsl:choose>
  </xsl:variable>
  <!-- Inclui os valores do primeiro e do segundo elemento filho do
       operador aritmetico informado no parametro -->
  <xsl:value-of select="$par_path_operador_aritmetico/attribute/@name"/>
  <xsl:value-of select="$var_simbolo"/>
  <xsl:choose>
     <xsl:when test="name($par_path_operador_aritmetico/*[position()=2])=
                          'attribute'">
        <xsl:value-of
           select="$par_path_operador_aritmetico/*[position()=2]/@name"/>
     </xsl:when>
     <xsl:when test="name($par_path_operador_aritmetico/*[position()=2])=
                          'value'">
        <xsl:call-template name="Incluir_Valor_Elemento_Value">
          <xsl:with-param name="par_primeiro_filho"
               select="$par_path_operador_aritmetico/*[position()=1]"/>
          <xsl:with-param name="par_segundo_filho"
               select="$par_path_operador_aritmetico/*[position() = 2]"/>
        </xsl:call-template>
     </xsl:when>
     <xsl:when test="name($par_path_operador_aritmetico/*[position()=2])=
                          'nav_parameter'">
        <xsl:text>#</xsl:text>
        <xsl:value-of
             select="$par_path_operador_aritmetico/*[position()=2]"/>
     </xsl:when>
   </xsl:choose>
</xsl:template>

<!--=========================================================================
    Template para tratar os elementos AND e OR, filhos do elemento
    selection do contexto.
    Resumo: Se o primeiro elemento filho do elemento informado no
    parametro for um elemento equal, different, greater, greater_equal,
    lesser ou lesser_equal, entao chama o template
    Tratar_Condicao_Relacional, passando como parametro os elementos
    filhos do elemento logico em questao. Inclui o simbolo "(" seguido
    pelo valor da variavel var_simbolo e pelos simbolos'"..'. Inclui os
    simbolos '"(' e chama novamente o template
    Tratar_Condicao_Relacional, passando como parametro o segundo
    elemento filho do operador logico. Senao (se o primeiro elemento
    filho nao e um operador relacional), chama recursivamente o template
```

```
    Tratar_Condicao_logica, passando como parametros os elementos filhos
    do elemento logico. Inclui o simbolo "(" seguido pelo valor da
    variavel var_simbolo e pelos simbolos'"..'. Inclui os simbolos '"('.
    Se o segundo elemento filho do elemento informado no parametro e um
    operador relacional (equal, different, etc.) entao chama o template
    Tratar_Condicao_Relacional, passando como parametro o segundo
    elemento filho. Senao, chama recursivamente o  template
    Tratar_Condicao_logica, passando como parametro o segundo elemento
    filho.
    Parametros: par_path_operador_logico - operador logico dentro de uma
    selecao
                valores: AND ou OR.
    ================================================================= -->
<xsl:template name="Tratar_Condicao_Logica">
  <xsl:param name="par_path_operador_logico"/>
  <!-- Atribui a variavel var_simbolo o simbolo corresponde ao elemento
       informado no parametro -->
  <xsl:variable name="var_simbolo">
    <xsl:choose>
       <xsl:when test="name($par_path_operador_logico) = 'AND'">
          <xsl:value-of select="'AND'"/>
     </xsl:when>
     <xsl:when test="name($par_path_operador_logico) = 'OR'">
        <xsl:value-of select="'OR'"/>
     </xsl:when>
  </xsl:choose>
  </xsl:variable>
  <!-- Constroi a expressao logica especificada na sub-arvore do elemento
       selection -->
  <xsl:choose>
   <xsl:when test="contains('equal different greater greater_equal lesser
         lesser_equal',name($par_path_operador_logico/*[position()=1]))">
      <xsl:call-template name="Tratar_Condicao_Relacional">
         <xsl:with-param name="par_path_operador_relacional"
                         select="$par_path_operador_logico/*"/>
      </xsl:call-template>
      <xsl:text>) </xsl:text>
      <xsl:value-of select="$var_simbolo"/><xsl:text> ".."
      "(</xsl:text>
      <xsl:call-template name="Tratar_Condicao_Relacional">
         <xsl:with-param name="par_path_operador_relacional"
              select="$par_path_operador_logico/*[position() = 2]"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:call-template name="Tratar_Condicao_Logica">
      <xsl:with-param name="par_path_operador_logico"
                      select="$par_path_operador_logico/*"/>
    </xsl:call-template>
    <xsl:text>) </xsl:text>
     <xsl:value-of select="$var_simbolo"/><xsl:text> ".."
    "( </xsl:text>
      <xsl:choose>
        <xsl:when test="contains('equal different greater greater_equal
             lesser lesser_equal',name($par_path_operador_logico/*
             [position() = 2]))">
          <xsl:call-template name="Tratar_Condicao_Relacional">
            <xsl:with-param name="par_path_operador_relacional"
```

```
                  select="$par_path_operador_logico/*[position() = 2]"/>
          </xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
          <xsl:call-template name="Tratar_Condicao_Logica">
           <xsl:with-param name="par_path_operador_logico"
                 select="$par_path_operador_logico/*[position() = 2]"/>
        </xsl:call-template>
        </xsl:otherwise>
     </xsl:choose>
  </xsl:otherwise>
</xsl:choose>
</xsl:template>

<!--========================================================================
    Template para determinar o tipo do atributo especificado no atributo
    name do elemento attribute, filho de um elemento equal, different,
    etc. (operador relacional) ou de um elemento plus, minus, etc.
    (aritmetico)
    Resumo: Se existe algum elemento nav_attrib (ou perspective_attrib),
    filho do elemento navigacional_class informado no parametro
    var_navigational_class, cujo atributo name tem o mesmo valor do
    parametro par_nome_atributo, entao retorna o valor do atributo type
    do elemento attrib (ou de seu filho perspective), filho do elemento
    conceptual_class cujo atributo id tem o mesmo valor do atributo
    conceptual_class do elemento nav_attrib (ou perspective_attrib)
    existente.
    Parametros: par_navigational_class - classe de navegacao a qual
                                         pertende o atributo
                valor: id do elemento navigational_class correspondente
                par_nome_atributo - nome do atributo
                valor: valor especificado no atributo name do elemento
                       attribute
    ======================================================================== -->
<xsl:template name="Determinar_Tipo_Atributo">
  <xsl:param name="par_navigational_class"/>
  <xsl:param name="par_nome_atributo"/>
  <!-- Retorna o valor do atributo type do elemento attrib ou
       perspective, especificado no atributo name do elemento attribute
       na selecao dos elementos do contexto -->
  <xsl:choose>
    <xsl:when test="id($par_navigational_class)/nav_attrib[
                      @name = $par_nome_atributo]">
       <xsl:variable name="var_conceptual_class"
              select="id($par_navigational_class)/nav_attrib[@name =
              $par_nome_atributo]/@conceptual_class"/>
       <xsl:value-of select="id($var_conceptual_class)/attrib[@name =
              $par_nome_atributo]/@type"/>
    </xsl:when>
    <xsl:when test="id($par_navigational_class)/perspective_attrib[@name=
                      $par_nome_atributo]">
       <xsl:variable name="var_conceptual_class"
            select="id($par_navigational_class)/perspective_attrib[@name=
                      $par_nome_atributo]/@conceptual_class"/>
       <xsl:value-of select="id($var_conceptual_class)/attrib/
                     perspective[@name = $par_nome_atributo]/@type"/>
    </xsl:when>
  </xsl:choose>
```

```
</xsl:template>

<!--=========================================================================
    Template para incluir na tabela lua nav_contexts a estrutura
    selection para os contextos arbitrarios (identificados quando o
    elemento navigational_context nao possui elementos filhos selection e
    tem pelo menos um elemento filho instance).
    Resumo: Inclui na tabela nav_contexts, as strings 'selection = {' e
    'condition = ' seguida pelos simbolos '"('. Para todos os elementos
    instance_attrib, filhos do elemento instance cujo atributo
    navigational_class tem o mesmo valor do parametro par_element_class,
    chama o template Determinar_Tipo_Atributo. Se o tipo retornado para o
    atributo espeficicado em seu atributo name for integer, real ou
    boolean, entao inclui o valor desse atributo name seguido pelo
    simbolo '=' e pelo valor do elemento instance_attrib que esta sendo
    tratado. Se o elemento instance_attrib nao e o ultimo a ser tratado,
    entao inclui uma virgula, se for, inclui o valor da variavel
    var_formato que contem a string ') OR "..'(caso existam outros
    elementos instance para o contexto) ou apenas ')"' (se o elemento
    instance for o ultimo tratado). Senao, se o tipo retornado nao e
    integer, boolean ou real entao, inclui os mesmos valores
    acrescentando aspas simples delimitando o valor do elemento
    instance_attrib. Inclui o simbolo '}', fechando a estrutura
    selection.
    Parametros: par_path_nav_context - caminho relativo para a
                identificacao do contexto que esta sendo tratado.
                valor: //navigational_context[@id = <id_contexto_atual>]
                par_element_class - valor do atributo element_class que
                esta sendo tratado.
                valor: id da classe que compoem os elementos do contexto.
    ========================================================================= -->
<xsl:template name="Incluir_Condicao_Contexto_Arbitrario">
  <xsl:param name="par_path_nav_context"/>
  <xsl:param name="par_element_class"/>
  selection =
  {
  <xsl:text>condition = "(</xsl:text>
  <xsl:for-each select="$par_path_nav_context/instance[
                        @navigational_class = $par_element_class]">
    <xsl:variable name="var_formato">
      <xsl:choose>
      <xsl:when test="position()!=last()">
        <xsl:value-of select="') OR '"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="'))&amp;quot;'"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:text>(</xsl:text>
    <xsl:for-each select="instance_attrib">
      <xsl:variable name="var_tipo_atributo">
         <xsl:call-template name="Determinar_Tipo_Atributo">
          <xsl:with-param name="par_navigational_class"
                          select="$par_element_class"/>
          <xsl:with-param name="par_nome_atributo" select="@name"/>
       </xsl:call-template>
      </xsl:variable>
```

```
        <xsl:choose>
          <xsl:when test="contains('integer real boolean',
                          $var_tipo_atributo)">
            <xsl:value-of select="@name"/>
            <xsl:text> = </xsl:text>
            <xsl:value-of select="."/>
            <xsl:choose>
              <xsl:when test="position()!=last()">
                 <xsl:text>,</xsl:text>
              </xsl:when>
              <xsl:otherwise>
                 <xsl:value-of select="$var_formato"/>
              </xsl:otherwise>
            </xsl:choose>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="@name"/>
            <xsl:text> = '</xsl:text>
            <xsl:value-of select="."/>
            <xsl:text>'</xsl:text>
            <xsl:choose>
              <xsl:when test="position()!=last()">
                <xsl:text>,</xsl:text>
              </xsl:when>
              <xsl:otherwise>
                <xsl:value-of select="$var_formato"/>
              </xsl:otherwise>
            </xsl:choose>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:for-each>
    </xsl:for-each>
    <xsl:text>
    }
    </xsl:text>
  </xsl:template>
</xsl:stylesheet>
```

Folha de Estilo: nav_indexes.xsl

```
<?xml version='1.0'?>
<!--========================================================================
|   Departamento de Informatica - PUC-Rio                                  |
|   Projeto:  Ferramenta de Especificacao OOHDM para o ambiente            |
|                OOHDM-Web 2.0                                             |
|   Programa: nav_indexes.xsl                                              |
|   Funcao: Completar a estrutura da tabela Lua nav_indexes, tratando      |
|           os elementos index e hierarch_index e seus atributos,          |
|           especificados no documento XML associado a folha de estilo     |
|           criado de acordo com a DTD OOHDM.                              |
|   Autor(a): Adriana Pereira de Medeiros                                  |
|   Data:   09/11/2000                                                     |
|   Versao:   1.0                                                          |
|   Sub-Programas:                                                         |
|   Alteracoes:                                                            |
|   Data  | Responsavel | Motivo                                           |
|=========================================================================-->
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">
  <!--=======================================================================
      Template para preencher a estrutura da tabela Lua nav_indexes,
      utilizando os elementos index e hierarch_index filhos do elemento
      navigational_model.
      ======================================================================= -->
  <xsl:template match="index|hierarch_index">
     <!-- Declaracao da variavel var_id_indice e atribuicao do valor do
          atributo id do elemento index ou hierarch_index que esta sendo
          tratado -->
    <xsl:variable name="var_id_indice" select="@id"/>
    <!-- Declaracao da variavel var_tipo_indice e atribuicao de seu valor
         de acordo com a seguinte condicao: se existe um elemento
         element_context, filho do elemento que esta sendo tratado, e o
         elemento navigational_context informado no seu atributo context
         possui mais de um elemento order_element descendente, filhos do
         mesmo elemento order, entao chama o template
         Determinar_Tipo_Indice passando como parametro o caracter "M"
         (indicando que o contexto possui multipla ordenacao); senao, chama
         este mesmo template sem informar valor para o parametro. -->
    <xsl:variable name="var_tipo_indice">
      <xsl:choose>
         <xsl:when test="count(order) > 1">
            <xsl:call-template name="Determinar_Tipo_Indice">
              <xsl:with-param name="par_ordenacao" select="'M'"/>
            </xsl:call-template>
         </xsl:when>
         <xsl:otherwise>
            <xsl:call-template name="Determinar_Tipo_Indice"/>
         </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <!-- Inclui o valor do atributo name do elemento index ou
```

```
     hierarch_index seguido pelos simbolos "=" e "{", e a string
     'type=' seguida pelo tipo do indice. Se existe algum elemento
     nav_parameter descendente de um elemento selection, filho do
     elemento index, hierarch_index ou navigational_context, entao
     chame o template Tratar_Element_Nav_Parameter para incluir o campo
     parameters e seus respectivos valores na tabela nav_indexes.
     Senao, se o valor do atributo type do elemento
     navigational_context informado no atributo context do elemento
     element_context e igual a 'by_query', entao inclui a string
     'parameters = {"sql_statement"},' -->
<xsl:value-of select="@name"/> =
{
 type = "<xsl:value-of select="$var_tipo_indice"/>",
<xsl:choose>
  <xsl:when test="id($var_id_indice)/selection//nav_parameter">
    <xsl:call-template name="Tratar_Elementos_Nav_Parameter">
       <xsl:with-param name="par_nav_parameter"
                 select="id($var_id_indice)/selection//nav_parameter"/>
    </xsl:call-template>
  </xsl:when>
  <xsl:when
       test="id(element_context/@context)/selection//nav_parameter">
    <xsl:call-template name="Tratar_Elementos_Nav_Parameter">
       <xsl:with-param name="par_nav_parameter"
                      select="id(element_context/@context)/selection//
                      nav_parameter"/>
    </xsl:call-template>
  </xsl:when>
  <xsl:otherwise>
    <xsl:if test="id(element_context/@context)/@type = 'by_query'">
       <xsl:text>parameters = {"sql_statement"},</xsl:text>
    </xsl:if>
  </xsl:otherwise>
</xsl:choose>
<!--Inclui a string 'elements = {'. Se o elemento index ou
    hierarch_index em questao possui elemento filho element_context
    especificado, entao inclui a string 'group = {' e a string 'context
    = ' seguida pelo valor do atributo name do elemento
    navigational_context informado no atributo context de
    element_context. Se existe algum elemento order_element para o
    elemento index ou hierarch_index, entao chame o template
    Tratar_Elementos_Order_Element, passando este elemento como
    parametro. Chame o template Tratar_Estruturas_Attributes_Selectors
    -->
 elements =
 {
<xsl:choose>
  <xsl:when test="element_context">
    <xsl:text>group =
    {
     context = "</xsl:text><xsl:value-of
                       select="id(element_context/@context)/@name"/>",
    <xsl:if test="order/order_element">
      <xsl:call-template name="Tratar_Elementos_Order_Element">
         <xsl:with-param name="par_order_element"
                        select="order/order_element"/>
      </xsl:call-template>
  </xsl:if>
  <xsl:call-template name="Criar_Estruturas_Attributes_Selectors">
```

```
    <xsl:with-param name="par_id_indice" select="$var_id_indice"/>
  </xsl:call-template>
  }
  </xsl:when>
  <!--Se nao existe elemento element_context especificado, entao para
      cada valor (id) existente no atributo element_class, inclui a
      string 'group' concatenada com o alor do atributo name do
      element_class e com o simbolo '='. Incluir o simbolo '{' e a
      string 'class = ' seguida pelo valor do atributo name do elemento
      navigational_class, cujo atributo id tem o mesmo valor do
      element_class seguido pelos simbolos '= {'. Se existe algum
      elemento selection para o indice, chama o template
      Criar_Estrutura_Condition. Se existe algum elemento order_element
      para o elemento index ou hierarch_index, cujo atributo
      navigational_class tem o mesmo valor do element_class entao chame
      o template Tratar_Elementos_Order_Element, passando este elemento
      como parametro. Chame o template
      Tratar_Estruturas_Attributes_Selectors-->
  <xsl:otherwise>
    <xsl:for-each select="id(@element_class)">
      <xsl:variable name="var_element_class" select="@id"/>
      <xsl:value-of select="concat('group_', @name, ' = ')"/>
      {
        class = "<xsl:value-of select="@name"/>",
      <xsl:choose>
        <xsl:when test="name(id($var_id_indice)) = 'index'">
          <xsl:for-each select="//index[@id=$var_id_indice]/selection">
             <xsl:if test="@selected_element_class = $var_element_class
                         or not(@selected_element_class)">
              <xsl:call-template name="Criar_Estrutura_Condition"/>
            </xsl:if>
          </xsl:for-each>
       </xsl:when>
       <xsl:otherwise>
         <xsl:for-each select="//hierarch_index[@id=
                              $var_id_indice]/selection">
            <xsl:if test="@selected_element_class = $var_element_class
                         or not(@selected_element_class)">
             <xsl:call-template name="Criar_Estrutura_Condition"/>
           </xsl:if>
         </xsl:for-each>
       </xsl:otherwise>
       </xsl:choose>
       <xsl:if test="id($var_id_indice)/order/order_element[
                       @navigational_class = $var_element_class]">
         <xsl:call-template name="Tratar_Elementos_Order_Element">
           <xsl:with-param name="par_order_element"
                     select="id($var_id_indice)/order/order_element[
                     @navigational_class = $var_element_class]"/>
         </xsl:call-template>
       </xsl:if>
       <xsl:call-template name="Criar_Estruturas_Attributes_Selectors">
          <xsl:with-param name="par_id_indice" select="$var_id_indice"/>
       </xsl:call-template>
     }<xsl:if test="position()!=last()">,
       </xsl:if>
     </xsl:for-each>
  </xsl:otherwise>
</xsl:choose>
```

```
    }
  }<xsl:if test="position()!=last()">,
  </xsl:if>
</xsl:template>
<!--=========================================================================
    Template para determinar o tipo do indice que sera incluido na
    estrutura nav_indexes.
    Resumo: Este tipo sera determinado com base no atributo type do
    elemento index ou hierarch_index que indica o tipo do indice que esta
    sendo tratado (estatico, dinamico ou temporario), no valor do
    parametro par_ordenacao e no proprio nome do elemento que esta sendo
    tratado (indice simples (index) ou hierarquico (hierarch_index). O
    tipo do indice deve ser determinado de acordo com as combinacoes
    estabelecidas no OOHDM-Web.
    Parametros: par_ordenacao -> tipo de ordenacao do contexto base para
                                 os elementos do indice
                valores: "M" (multipla ordenacao) ou vazio
    ========================================================== -->
<xsl:template name="Determinar_Tipo_Indice">
  <xsl:param name="par_ordenacao"/>
  <!-- Declaracao da variavel var_type e atribuicao de seu valor de
       acordo com o valor do atributo type do elemento que esta sendo
       tratado(index ou hierarch). Se o valor do atributo type  for
       "static", a variavel recebera o caracter "E"; se for "dynamic",
       recebera o caracter "D" e se for "temporary". Esta variavel sera
       usada para determinar o tipo do indice que sera incluido na
       estrutura lua nav_indexes.-->
  <xsl:variable name="var_type">
      <xsl:choose>
         <xsl:when test="@type = 'static'">
            <xsl:value-of select="'E'"/>
         </xsl:when>
         <xsl:when test="@type = 'dynamic'">
            <xsl:value-of select="'D'"/>
         </xsl:when>
         <xsl:when test="@type = 'temporary'">
            <xsl:value-of select="'T'"/>
         </xsl:when>
      </xsl:choose>
  </xsl:variable>
  <!-- Concatenacao dos valores da variavel var_type, do parametro
       par_ordenacao e dos caracteres "S" ou "H" que indicam,
       respectivamente, um indice simples (elemento index) ou um indice
       hierarquico (elemento         hierarch_index). O resultado dessa
       concatenacao sera o tipo definitivo do indice que sera atribuido a
       variavel var_tipo_indice no template "chamador". -->
  <xsl:choose>
     <xsl:when test="name(.) = 'hierarch_index'">
        <xsl:value-of select="concat($var_type,'H',$par_ordenacao)"/>
     </xsl:when>
     <xsl:otherwise>
         <xsl:value-of select="concat($var_type,'S',$par_ordenacao)"/>
     </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<!--=========================================================================
    Template para criar a estrutura condition da tabela nav_indexes
```

```
    utilizando um dos elementos filhos do elemento selection (elemento
    equal, different, greater, greater_equal, lesser, lesser_equal, AND
    ou OR).
    Resumo: Inclui a string 'condition = "('. Se o elemento filho do
    elemento selection e um elemento equal, different, greater,
    greater_equal, lesser ou lesser_equal, entao chama o template
    Tratar_Condicao_Relacional passando como parametro todos os elementos
    filhos do elemento selection (pela DTD sera sempre um). Senao, (se o
    elemento filho for AND ou OR), chama o template
    Tratar_Condicao_Logica passando como parametro todos os elementos
    filhos do selection. Inclui o simbolo ')' seguido de aspas.
    ================================================================= -->
<xsl:template name="Criar_Estrutura_Condition">
  <xsl:text>condition = "(</xsl:text>
 <xsl:choose>
  <xsl:when test="equal or different or greater or greater_equal or
                  lesser or lesser_equal">
    <xsl:call-template name="Tratar_Condicao_Relacional">
      <xsl:with-param name="par_path_operador_relacional" select="./*"/>
    </xsl:call-template>
  </xsl:when>
  <xsl:otherwise>
    <xsl:call-template name="Tratar_Condicao_Logica">
       <xsl:with-param name="par_path_operador_logico" select="./*"/>
    </xsl:call-template>
  </xsl:otherwise>
  </xsl:choose>
  <xsl:text>)",</xsl:text>
</xsl:template>
<!--=========================================================================
    Template para criar as estruturas attributes e selectors da tabela
    nav_indexes utilizando os elementos shown_attrib, dynamic_selector e
    static_selector, entre outros.
    Parametros: par_id_indice -> identificador do elemento index ou
                                 hierarch_index que esta sendo tratado.
                valores: valor do atributo id do elemento index ou
                         hierach_index
   ========================================================================= -->
<xsl:template name="Criar_Estruturas_Attributes_Selectors">
  <xsl:param name="par_id_indice"/>
  <!-- Inclui a string 'attributes = {'. Chame o template
       Incluir_Atributos passando como parametro os elementos
       shown_attrib e static_selector filhos do elemento index ou do
       elemento level (hierach_index) que esta sendo tratado. -->
  attributes = {
  <xsl:choose>
    <xsl:when test="name(id($par_id_indice)) = 'index'">
      <xsl:call-template name="Incluir_Atributos">
        <xsl:with-param name="par_shown_attrib"
                        select="id($par_id_indice)//shown_attrib"/>
        <xsl:with-param name="par_static_selector"
                        select="id($par_id_indice)/static_selector"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:when test="name(id($par_id_indice)) = 'hierarch_index'">
      <xsl:for-each select="id($par_id_indice)/level">
        {
```

```
      <xsl:call-template name="Incluir_Atributos">
        <xsl:with-param name="par_shown_attrib"
                        select=".//shown_attrib"/>
        <xsl:with-param name="par_static_selector"
                        select="static_selector"/>
      </xsl:call-template>
      <xsl:choose>
        <xsl:when test="position()!=last()">},</xsl:when>
        <xsl:otherwise>}</xsl:otherwise>
      </xsl:choose>
    </xsl:for-each>
  </xsl:when>
 </xsl:choose>
 },
 <!-- Inclui a string 'selectors = {'. Para cada elemento shown_attrib
      filho de um elemento dynamic_selector do indice que esta sendo
      tratado, inclui a string 'destination = ' seguida pelo valor do
      atributo name do elemento informado no atributo target do elemento
      selector_destination; se o elemento informado possui algum
      elemento nav_parameter descendente de um elemento selection, entao
      chame o template Incluir_Valor_Elemento_Nav_Parameter. Senao, se o
      elemento informado no atributo target e um elemento
      navigational_context e o valor do seu atributo type e 'by_query',
      entao inclui a string ',#sql_statement'.-->
 <xsl:variable name="var_context"
               select="id($par_id_indice)/element_context/@context"/>
 selectors =
{
<xsl:for-each select="id($par_id_indice)/dynamic_selector/shown_attrib|
              id($par_id_indice)/level/dynamic_selector/shown_attrib">
  <xsl:value-of select="@name"/> =
  {
  destination = "<xsl:value-of
                  select="id(../selector_destination/@target)/@name"/>
  <xsl:choose>
    <xsl:when test="id($var_context)/selection//nav_parameter">
      <xsl:if test="../selector_destination/@target = $var_context or
                    name(id($par_id_indice)) = 'hierarch_index'">
        <xsl:call-template name="Incluir_Valor_Elemento_Nav_Parameter">
          <xsl:with-param name="par_path_nav_parameter"
                          select="id($var_context)/selection//nav_parameter"/>
        </xsl:call-template>
      </xsl:if>
     </xsl:when>
     <xsl:otherwise>
       <xsl:choose>
         <xsl:when test="name(id(../selector_destination/@target)) =
                         'navigational_context' and
                         id(../selector_destination/@target)/@type =
                         'by_query'">
           <xsl:text>,#sql_statement</xsl:text>
         </xsl:when>
       </xsl:choose>
     </xsl:otherwise>
   </xsl:choose>
  <xsl:text>",</xsl:text>
  <!-- Chame o template Incluir_Campo_Dest_Page. -->
  <xsl:call-template name="Incluir_Campo_Dest_Page">
```

```
    <xsl:with-param name="par_atributo_target"
                    select="../selector_destination/@target"/>
  </xsl:call-template>
  <!-- Chame o template Incluir_Campo_Inst. -->
  <xsl:call-template name="Incluir_Campo_Inst">
    <xsl:with-param name="par_indice" select="$par_id_indice"/>
    </xsl:call-template>
    }<xsl:if test="position()!=last() or
                id($par_id_indice)/static_selector">,
  </xsl:if>
</xsl:for-each>
  <!-- Para todos os elementos static_selector, filhos do elemento index
       ou hierach_index em questao, inclui a string 'anchor = ' seguida
       pelo valor do seu elemento filho name, a string 'destination = '
       seguida pelo valor do atributo name do elemento informado no
       atributo target do elemento selector_destination e chame o
       template Incluir_Campo_Dest_Page  -->
<xsl:for-each select="id($par_id_indice)/static_selector">
   <xsl:value-of select="./name"/> =
   {
   anchor = "<xsl:value-of select="./name"/>",
   destination = "<xsl:value-of
                  select="id(selector_destination/@target)/@name"/>",
     <xsl:call-template name="Incluir_Campo_Dest_Page">
       <xsl:with-param name="par_atributo_target"
                  select="selector_destination/@target"/>
     </xsl:call-template>
     }
</xsl:for-each>
}
</xsl:template>
<!--=========================================================================
    Template para incluir os valores dos atributos que serao exibidos no
    indice na estrutura attributes da tabela nav_indexes, utilizando os
    elementos shown_attrib e static_selector.
    Resumo: Para todos os elementos shown_attrib (descendentes do
    elemento index ou hierarch_index) e static_selector (filhos do
    elemento index ou hierarch_index), inclui na estrutura attributes o
    valor do elemento name, caso o elemento em questao seja
    static_selector, ou o valor do atributo name do elemento shown_attrib
    entre aspas. Se o elemento nao e o ultimo tratado, inclui uma
    virgula.
    Parametros: par_shown_attrib -> elementos shown_attrib descendentes
                                    do elemento index ou hierarch_index.
                valor: caminho relativo para recuperar todos os elementos
                       shown_attrib descendentes.
                par_static_selector -> elementos static_selector filhos
                                    do elemento index ou hierarch_index.
                valor: caminho relativo para recuperar todos os elementos
                       static_selector
    ========================================================================= -->
<xsl:template name="Incluir_Atributos">
  <xsl:param name="par_shown_attrib"/>
  <xsl:param name="par_static_selector"/>
  <xsl:for-each select="$par_shown_attrib|$par_static_selector">
    <xsl:choose>
      <xsl:when test="name(.) = 'static_selector'">
```

```
        "<xsl:value-of select="./name"/>"
          <xsl:if test="position()!=last()">,</xsl:if>
      </xsl:when>
      <xsl:otherwise>
        "<xsl:value-of select="@name"/>"
          <xsl:if test="position()!=last()">,</xsl:if>
      </xsl:otherwise>
    </xsl:choose>
</xsl:for-each>
</xsl:template>
<!--=========================================================================
    Template para incluir os valores dos elementos nav_parameter no campo
    context, quando o contexto informado no indice possui parametros, ou
    no campo destination quando o elemento destino de um selector possui
    parametros.
    Resumo: Inclui uma virgula. Para todos os elementos nav_parameter
    informados no parametro par_path_nav_parameter inclui o simbolo #
    seguido do valor do elemento nav_parameter em questao. Se o elemento
    nao e o ultimo tratado, inclui uma virgula.
    Parametros: par_path_nav_parameter -> sub_arvore dos elementos
                                          nav_parameter
                valor: caminho relativo para recuperar todos os elementos
                       nav_parameter descendentes do elemento selection.
    ========================================================================= -->
<xsl:template name="Incluir_Valor_Elemento_Nav_Parameter">
  <xsl:param name="par_path_nav_parameter"/>
  <xsl:text>,</xsl:text>
  <xsl:for-each select="$par_path_nav_parameter">
    <xsl:text>#</xsl:text><xsl:value-of select="."/>
    <xsl:if test="position()!=last()">
      <xsl:text>,</xsl:text>
    </xsl:if>
  </xsl:for-each>
</xsl:template>
<!--=========================================================================
    Template para incluir o o campo inst na tabela lua nav_indexes.
    Resumo: Inclui a string 'inst = {'. Se existe algum elemento
    map_attribute com valor 'yes' para o atributo primary_key, filho do
    elemento map_conceptual_class cujo atributo conceptual_class tenha o
    mesmo valor do atributo id do elemento conceptual_class ao qual
    pertence o atributo seletor, entao para cada elemento map_attribute
    existente inclui o valor do seu atributo db_attribute. Senao,
    recupere o valor do atributo name do primeiro elemento attrib, filho
    do elemento conceptual_class armazenado na variavel
    var_conceptual_class. Se existe algum elemento map_attribute cujo
    atributo name tenha o mesmo valor do atributo name de attrib, entao
    inclui o valor do seu atributo db_attribute. Senao, inclui o valor do
    atributo name do elemento attrib.
    Parametros: par_indice ->  elemento index ou hierarch_index que esta
                               sendo tratado.
                valor: valor do atributo id do elemento index ou
                       hierarch_index que esta sendo tratado.
    ========================================================================= -->
<xsl:template name="Incluir_Campo_Inst">
  <xsl:param name="par_indice"/>
  <!-- Recupera os valores dos atributos name e navigational_class do
```

```
      elemento shown_attrib em questao -->
  <xsl:variable name="var_name_shown_attrib" select="@name"/>
  <xsl:variable name="var_navigational_class"
                select="@navigational_class"/>
  <!-- Recupera o valor do atributo id do elemento conceptual_class ao
       qual pertence o atributo informado no elemento shown_attrib -->
  <xsl:variable name="var_conceptual_class">
   <xsl:choose>
     <xsl:when test="id($var_navigational_class)/nav_attrib[
                       @name=$var_name_shown_attrib]">
        <xsl:value-of select="id($var_navigational_class)/nav_attrib[
                      @name=$var_name_shown_attrib]/@conceptual_class"/>
     </xsl:when>
     <xsl:when test="id($var_navigational_class)/perspective_attrib[
                     @name=$var_name_shown_attrib]">
        <xsl:value-of select="id($var_navigational_class)/
                      perspective_attrib[@name=$var_name_shown_attrib]/
                      @conceptual_class"/>
     </xsl:when>
   </xsl:choose>
</xsl:variable>
<!-- Cria uma variavel contendo o caminho relativo para encontrar o
     elemento map_conceptual_class cujo atributo conceptual_class tem o
     mesmo valor do atributo id recuperado na variavel
     var_conceptual_class. -->
<xsl:variable name="var_path_map_attribute"
              select="//OOHDM_WEB/map_conceptual_class[
                      @conceptual_class=$var_conceptual_class]"/>
  <!-- Inclui campo inst e os valores dos campos que compoem a chave da
       tabela no banco de dados entre aspas -->
<xsl:text>inst = {</xsl:text>
  <xsl:choose>
  <xsl:when test="$var_path_map_attribute/map_attribute[
                  @primary_key = 'yes']">
    <xsl:for-each select="$var_path_map_attribute/map_attribute[
                          @primary_key='yes']">
      <xsl:text>"</xsl:text><xsl:value-of select="@db_attribute"/>
      <xsl:text>"</xsl:text>
      <xsl:if test="position()!=last()">
          <xsl:text>,</xsl:text>
        </xsl:if>
    </xsl:for-each>
   </xsl:when>
   <xsl:otherwise>
     <xsl:variable name="var_name_chave"
                   select="id($var_conceptual_class)/attrib[1]/@name"/>
     <xsl:variable name="var_db_attribute"
                 select="$var_path_map_attribute/map_attribute[@name =
                         $var_name_chave]/@db_attribute"/>
     <xsl:choose>
        <xsl:when test="$var_db_attribute">
          <xsl:text>"</xsl:text>
          <xsl:value-of select="$var_db_attribute"/>
          <xsl:text>"</xsl:text>
        </xsl:when>
        <xsl:otherwise>
          <xsl:text>"</xsl:text>
```

```
            <xsl:value-of select="$var_name_chave"/>
            <xsl:text>"</xsl:text>
          </xsl:otherwise>
        </xsl:choose>
    </xsl:otherwise>
    </xsl:choose>
    <xsl:text>}</xsl:text>
  </xsl:template>
  <!--====================================================================
      Template para incluir o valor do campo dest_page da tabela lua
      nav_indexes utilizando os elementos shown_attrib, static_selector e
      interface_element(filho do elemento OOHDM_WEB).
      Resumo: Se o nome do elemento cujo atributo id tem o mesmo valor do
      parametro par_atributo_target eh index ou hierarch_index, entao
      atribui o valor desse parametro a variavel var_id_elemento. Senao, se
      o nome do elemento que esta sendo tratado e static_selector, entao
      atribui o valor do atributo id desse elemento a var_id_elemento,
      senao, atribui o valor do atributo source do elemento shown_attrib
      que esta sendo tratado. Inclui a string 'dest_page =' seguida pelo
      valor do elemento interface_element, filho do elemento OOHDM_WEB,
      cujo atributo target contem o valor armazenado na variavel
      var_id_elemento, entre aspas.
      Parametros: par_atributo_target -> destino de um seletor do indice
                  valor: valor do atributo target do elemento
                         selector_destination, filho de um elemento
                         dynamic_selector ou static_selector.
      ==================================================================== -->
  <xsl:template name="Incluir_Campo_Dest_Page">
    <xsl:param name="par_atributo_target"/>
    <xsl:variable name="var_id_elemento">
      <xsl:choose>
        <xsl:when test="contains('index hierarch_index',
                        name(id($par_atributo_target)))">
          <xsl:value-of select="$par_atributo_target"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:choose>
            <xsl:when test="name(.) = 'static_selector'">
              <xsl:value-of select="./@id"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="./@navigational_class"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:text>
      dest_page = "</xsl:text>
    <xsl:value-of select="//OOHDM_WEB/interface_element[contains(@target,
                          $var_id_elemento)]"/>
    <xsl:text>"</xsl:text>
    <xsl:if test="name(.) != 'static_selector'">
      <xsl:text>,
      </xsl:text>
    </xsl:if>
</xsl:template>
</xsl:stylesheet>
```

Apêndice V

Especificação OOHDM-ML do site "Arquitetura Moderna"

Arquivo: rio.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xml" href="../../oohdmXWeb/stylesheets/oohmd_translation.xsl"?>
<!-- ============================================================
    |   Documento XML para especificacao do site Arquitetura Moderna no Rio          |
    |   Data: 06/06/2000                                                             |
    |   Autora: Adriana Pereira de Medeiros                                          |
    | ============================================================== -->
<!DOCTYPE OOHDM SYSTEM "../../oohdmXWeb/DTDs/OOHDM_v1_2.dtd"
[
  <!ENTITY modelo_conceitual   SYSTEM "rio_con_model.xml">
  <!ENTITY modelo_navegacao  SYSTEM "rio_nav_model.xml">
  <!ENTITY mapeamento_oohdmweb SYSTEM "rio_oohdmweb.xml">
  <!ENTITY % MS_ACCESS "INCLUDE">
  <!ENTITY % ORACLE  "IGNORE">
]>
<OOHDM name="Arquiteturas_Rio" version="1.0">
        <!--======= Projeto OOHDM da aplicacao ======= -->
        <OOHDM_BASE>
              <!--***** Definicao do Modelo Conceitual ***** -->
              <conceptual_model>
                    &modelo_conceitual;
              </conceptual_model>
              <!-- ***** Definicao do Modelo de Navegacao *****-->
              <navigational_model>
                    &modelo_navegacao;
              </navigational_model>
              <!--***** Definicao do projeto da Interface Abstrata *****-->
                    <interface_model/>
        </OOHDM_BASE>
        <!-- ===== Mapeamento dos modelos da aplicacao para o ambiente OOHDM-Web ===== -->
        <OOHDM_WEB>
              &mapeamento_oohdmweb;
        </OOHDM_WEB>
</OOHDM>
```

Arquivo: rio_con_model.xml

```
<!-- ============================================================
    |                        Especificação do Modelo Conceitual                      |
    |   * Classes Conceituais (atributos e perspectivas)                             |
    |   * Relacionamentos                                                            |
    | ============================================================== -->
```

```
<!-- ================== Classe Conceitual obras ================== -->
<conceptual_class id="id_obras" name="obras" property="Classe obras">
        <attrib name="chave" type="string" size="20"/>
        <attrib name="nome_obra" type="string" size="70"/>
        <attrib name="ano_construcao_obra" type="string" size="4"/>
        <attrib name="endereco" type="string"  size="50"/>
        <attrib name="descricao_obra" type="text">
                <perspective name="descricao"  type="text" default="yes"/>
                <perspective name="foto_obra" type="image" size="30"/>
        </attrib>
        <attrib name="local_obra" type="string" size="50"/>
        <attrib name="categoria" type="string" size="50"/>
</conceptual_class>
<!-- ================== Classe Conceitual arquitetos ================== -->
  <conceptual_class id="id_arquitetos" name="arquitetos">
        <attrib name="chave_arquit" type="string" size="20"/>
        <attrib name="nome_arquit" type="string" size="50"/>
        <attrib name="descricao_arquit" type="text">
                <perspective name="descricao_arquit"  type="text" default="yes"/>
                <perspective name="foto_arquit" type="image" size="50"/>
        </attrib>
        <attrib name="ano_nasc_arquit" type="string" size="4"/>
</conceptual_class>
<!-- ================== Classe Conceitual construtoras ==================  -->
<conceptual_class id="id_construtoras" name="construtoras">
        <attrib name="chave_construtora" type="string" size="20"/>
        <attrib name="nome_construtora" type="string" size="50"/>
</conceptual_class>
<!-- ==============  Relacionamento arquitetos projeta obras ==============  -->
<relationship id="relacao_projeta" name="arquit_projeta_obra"
                source_class="id_arquitetos" target_class="id_obras"
                source_cardinality="n" target_cardinality="n">
</relationship>
<!-- ============ Relacionamento arquitetos projeta com arquitetos ========== -->
<relationship id="relacao_projeta_com" name="arquit_projeta_com_arquit"
                source_class="id_arquitetos" target_class="id_arquitetos"
                source_cardinality="n" target_cardinality="n">
</relationship>
<!-- ============== Relacionamento construtora constroi obras ============== -->
<relationship id="relacao_constroi" name="construtora_constroi_obra"
                source_class="id_construtoras" target_class="id_obras"
                source_cardinality="1" target_cardinality="n">
</relationship>
```

Arquivo: rio_nav_model.xml

```
<!-- ============================================================
   |                    Especificação do Projeto de Navegação                    |
   |    * Classes Navegacionais e seus atributos                                 |
   |    * Elos                                                                   |
   |    * Contextos de Navegação                                                 |
   |    * Classes em Contexto                                                    |
   |    * Estruturas de Acesso (índices)                                         |
   |    * Landmarks                                                              |
   | ============================================================ -->
   <!-- ================== Classe navegacional obras ==================-->
   <navigational_class id="no_obras" name="obras">
        <nav_attrib name="chave" conceptual_class="id_obras"/>
        <nav_attrib name="nome_obra" conceptual_class="id_obras"/>
        <nav_attrib name="ano_construcao_obra" conceptual_class="id_obras"/>
        <nav_attrib name="nome_construtora"  conceptual_class="id_construtoras"/>
        <nav_attrib name="nome_arquit"  conceptual_class="id_arquitetos"/>
        <nav_attrib name="categoria" conceptual_class="id_obras"/>
        <perspective_attrib name="foto_obra" perspective="Foto"
                            conceptual_attrib="descricao_obra"
                            conceptual_class="id_obras" optional="yes"/>
        <perspective_attrib name="descricao_obra" perspective="Texto"
                            conceptual_attrib="descricao_obra"
                            conceptual_class="id_obras"/>
   </navigational_class>
   <!-- ================== Classe navegacional arquitetos ================== -->
   <navigational_class id="no_arquitetos" name="arquitetos">
        <nav_attrib name="chave_arquit" conceptual_class="id_arquitetos"/>
        <nav_attrib name="nome_arquit" conceptual_class="id_arquitetos"/>
        <perspective_attrib name="foto_arquit" perspective="foto_arquit"
                            conceptual_attrib="descricao_arquit"
                            conceptual_class="id_arquitetos"/>
        <nav_attrib_index name="arq_obra_ind" index="indice_hierarquico_obras_arquit"/>
   </navigational_class>
   <!-- ================== Elo arquiteto projeta obra ======================== -->
   <link id="link_projeta" name="arquit_projeta_obra"
          source_class="no_arquitetos" target_class="no_obras"
          source_cardinality="n" target_cardinality="n">
   </link>
   <!-- ================== Elo arquiteto projeta com arquiteto ================ -->
   <link id="link_projeta_com" name="arquit_projeta_com_arquit"
        source_class="no_arquitetos" target_class="no_arquitetos"
        source_cardinality="n" target_cardinality="n">
   </link>
   <!-- ================== Contexto obras em ordem alfabetica ================== -->
   <navigational_context id="contexto_obras_alfa" name="obras_alfa"
                         element_class="no_obras" type="static"
                         navigation_type="circular">
        <order id="ordenacao_obras" default="yes">
             <order_element name="nome_obra" navigational_class="no_obras"
                            criteria="asc"/>
        </order>
   </navigational_context>
```

```
<!-- ====================== Contexto arquitetos por obra ================== -->
<navigational_context id="contexto_arquit_por_obra" name="arquit_por_obra"
                      element_class="no_arquitetos" type="static"
                      navigation_type="sequential">
    <selection>
        <AND>
            <equal>
                <attribute name="chave" navigational_class="no_obras" link="link_projeta"/>
                <nav_parameter>chave_obra_par</nav_parameter>
            </equal>
            <equal>
                <attribute name="nome_obra" navigational_class="no_obras"/>
                <nav_parameter>nome_obra_par</nav_parameter>
            </equal>
        </AND>
    </selection>
    <order id="ordenacao_por_arquiteto1" default="yes">
        <order_element name="nome_arquit" navigational_class="no_arquitetos"
                       criteria="asc"/>
    </order>
</navigational_context>
<!-- ================== Contexto obras por ano ================== -->
<navigational_context id="contexto_obras_por_ano" name="obras_por_ano"
                element_class="no_obras" type="static"
                navigation_type="sequential">
    <selection>
        <equal>
            <attribute name="ano_construcao_obra" navigational_class="no_obras"/>
            <nav_parameter>ano</nav_parameter>
        </equal>
    </selection>
    <order  id="ordenacao_por_ano" default="yes">
        <order_element name="ano_construcao_obra" navigational_class="no_obras"
                       criteria="asc"/>
        </order>
</navigational_context>
<!-- ================== Contexto obras por arquitetos ================== -->
 <navigational_context id="contexto_obras_por_arquit" name="obras_por_arquit"
                element_class="no_obras" type="static"
                navigation_type="circular">
    <selection>
        <equal>
            <attribute name="chave_arquit" navigational_class="no_arquitetos"
                       link="link_projeta"/>
            <nav_parameter>arquit_key</nav_parameter>
        </equal>
    </selection>
    <order  id="ordenacao_por_arquiteto2" default="yes">
        <order_element name="nome_arquit" navigational_class="no_arquitetos" criteria="asc"/>
    </order>
</navigational_context>
<!-- ================== Contexto obras por categoria ================== -->
 <navigational_context id="contexto_obras_por_categoria" name="obras_por_categoria"
                element_class="no_obras" type="static"
                navigation_type="sequential">
    <selection>
```

```
            <equal>
                <attribute name="categoria" navigational_class="no_obras"/>
                <nav_parameter>categ</nav_parameter>
            </equal>
        </selection>
        <order  id="ordenacao_por_obra" default="yes">
            <order_element name="nome_obra" navigational_class="no_obras" criteria="asc"/>
        </order>
</navigational_context>
<!-- ================== Classe em contexto obras por ano ==================-->
<context_class id="classe_contexto_obras_por_ano" base_class="no_obras"
                    contexts="contexto_obras_por_ano">
        <nav_attrib  name="local_obra" conceptual_class="id_obras"/>
         <!-- Links "proximo" e "anterior" para o contexto corrente (obras_por_ano) -->
        <anchor type="next">
              <name>c_prox.gif</name>
        </anchor>
        <anchor type="previous">
              <name>c_ant.gif</name>
        </anchor>
        <!-- Links "proximo" e "anterior" para o contexto obras_alfa -->
        <anchor type="next">
              <name>c_prox.gif</name>
              <destination target="contexto_obras_alfa"/>
        </anchor>
        <anchor type="previous">
              <name>c_ant.gif</name>
              <destination target="contexto_obras_alfa"/>
        </anchor>
        <!-- Links "proximo" e "anterior" para o contexto obras por categoria -->
        <anchor type="next">
              <name>c_prox.gif</name>
              <destination target="contexto_obras_por_categoria"/>
        </anchor>
        <anchor type="previous">
              <name>c_ant.gif</name>
              <destination target="contexto_obras_por_categoria"/>
        </anchor>
        <!-- Links "proximo" e "anterior" para o contexto obras por arquiteto -->
        <anchor type="next">
              <name>c_prox.gif</name>
              <destination target="contexto_obras_por_arquit"/>
        </anchor>
        <anchor type="previous">
              <name>c_ant.gif</name>
              <destination target="contexto_obras_por_arquit"/>
        </anchor>
</context_class>
<!-- ================== Indice obras em ordem alfabetica ==================-->
<index id="indice_obras_alfabetico" name="obras_alfa_idx" type="static">
        <element_context context="contexto_obras_alfa"/>
        <dynamic_selector>
            <shown_attrib  name="nome_obra" navigational_class="no_obras"/>
            <selector_destination  target="contexto_obras_alfa"/>
        </dynamic_selector>
</index>
```

```
<!-- ======================== Indice obras por ano ======================-->
<index  id="index_obras_por_ano" name="obras_por_ano_idx" type="static">
      <element_context context="contexto_obras_por_ano"/>
      <shown_attrib  name="ano_construcao_obra" navigational_class="no_obras"/>
      <dynamic_selector>
            <shown_attrib  name="nome_obra" navigational_class="no_obras"/>
            <selector_destination  target="contexto_obras_por_ano"/>
      </dynamic_selector>
 </index>
<!-- ===================== Indice hierarquico obras por arquiteto ==============-->
<hierarch_index id="indice_hierarquico_obras_arquit" name="obras_por_arquit_H_idx"
                  type="static">
      <element_context context="contexto_obras_por_arquit"/>
      <level>
            <dynamic_selector>
                  <shown_attrib name="nome_arquit" navigational_class="no_arquitetos"/>
                  <selector_destination target="indice_hierarquico_obras_arquit"/>
            </dynamic_selector>
      </level>
      <level>
            <shown_attrib name="ano_construcao_obra" navigational_class="no_obras"/>
            <dynamic_selector>
                  <shown_attrib name="nome_obra" navigational_class="no_obras"/>
                  <selector_destination target="contexto_obras_por_arquit"/>
            </dynamic_selector>
      </level>
      <order id="ordenacao_nome_obra">
            <order_element name="nome_obra" navigational_class="id_obras" criteria="asc"/>
      </order>
</hierarch_index>
<!-- ================== Indice hierarquico obras por categoria ================-->
<hierarch_index id="indice_hierarquico_obras_categoria" name="obras_por_categoria_H_idx"
                  type="static">
      <element_context context="contexto_obras_por_categoria"/>
      <level>
            <shown_attrib name="nome_obra" navigational_class="no_obras"/>
            <dynamic_selector>
                  <shown_attrib name="categoria" navigational_class="no_obras"/>
                  <selector_destination target="indice_hierarquico_obras_categoria"/>
            </dynamic_selector>
      </level>
      <level>
            <dynamic_selector>
                  <shown_attrib name="nome_obra" navigational_class="no_obras"/>
                  <selector_destination target="contexto_obras_por_categoria"/>
            </dynamic_selector>
      </level>
      <order id="ordenacao_nome_obra2">
            <order_element name="nome_obra" navigational_class="id_obras" criteria="asc"/>
      </order>
</hierarch_index>
<!-- ================== Landmark obras em ordem alfabetica ================-->
<landmark target="indice_obras_alfa">Obras em Ordem Alfabetica</landmark>
```

Arquivo: rio_oohdmweb.xml

```
<!-- ============================================================
    |       Especificação do Mapeamento do Projeto para o ambiente OOHDM-Web      |
    |   * Mapeamento das Classes Conceituais e seus Atributos                      |
    |   * Mapeamento de Relacionamentos                                            |
    |   * Mapeamento das Classes em Contexto                                       |
    |   * Mapeamento dos Elementos de Interface                                    |
    ============================================================ -->
    <!-- ============ Mapeamento da classe conceitual obras para a tabela obras ==========-->
    <map_conceptual_class  conceptual_class="id_obras">
        <map_attribute id="map_chave_obra" name="chave" db_attribute="chave_obra"
                       primary_key="yes"/>
        <map_attribute id="map_nome_obra" name="nome_obra" db_attribute="nome_obra"
                       primary_key="yes"/>
        <map_attribute id="map_ano_construcao" name="ano_construcao_obra"
                       db_attribute="ano_construcao"/>
    </map_conceptual_class>
    <!-- ======= Mapeamento da classe conceitual arquitetos para a tabela arquitetos ========-->
    <map_conceptual_class conceptual_class="id_arquitetos">
        <map_attribute id="map_foto_arquit" name="foto_arquit"  db_attribute="foto"/>
        <map_attribute id="map_descricao_arquit" name="descricao_arquit"
                       db_attribute="descricao"/>
    </map_conceptual_class>
    <!-- ====== Mapeamento da classe conceitual construtoras para a tabela construtoras =====-->
    <map_conceptual_class conceptual_class="id_construtoras"/>
    <!-- ====== Mapeamento da classe em contexto obras_ano para a tabela obras_ano  ====== -->
    <map_context_class context_class="classe_contexto_obras_por_ano" db_class_ctx="obras_ano">
        <map_attribute id="map_local_obra" name="local_obra"
                       db_attribute="local_realizacao_obra"/>
    </map_context_class>
    <!--  ====== Mapeamento do relacionamento Arquiteto projeta Obra para a tabela
                  arquit_projeta_obra  ====== -->
    <map_relationship relationship="relacao_projeta" db_relationship="arquit_projeta_obra">
        <db_source_field>chv_arquit</db_source_field>
        <db_destination_field map_attribute="map_chave_obra">chv_obra</db_destination_field>
    </map_relationship>
    <!--  ====== Mapeamento do relacionamento Arquiteto projeta com Arquiteto para a tabela
                  arquit_projeta_arquit ====== -->
    <map_relationship relationship="relacao_projeta_com" db_relationship="arquit_projeta_arquit">
        <db_source_field>chv_arquit_orig</db_source_field>
        <db_destination_field>chv_arquit_dest</db_destination_field>
    </map_relationship>
    <!--  ==== Mapeamento do relacionamento Construtora constroi Obra para a tabela obras ====-->
    <map_relationship relationship="relacao_constroi" db_relationship="obras">
        <db_source_field>chv_construtora</db_source_field>
    </map_relationship>
    <!-- ==== Mapeamento dos objetos de interface para classes de navegacao e indices ====-->
    <interface_element target="no_obras">obra.html</interface_element>
    <interface_element target="no_arquitetos">arquiteto.html</interface_element>
    <interface_element target="indice_hierarquico_obras_arquit">ind_arq.html</interface_element>
    <interface_element target="indice_hierarquico_obras_categoria">ind_cat.html</interface_element>
```

Especificação OOHDM-ML do site "Coleção de CDs"

Arquivo: colecao_cds.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xml" href="../../oohdmXWeb/stylesheets/oohmd_translation.xsl"?>
<!-- ================================================================
    |   Documento XML para especificacao do site Colecao de CDs                    |
    |   Data: 12/03/2001                                                           |
    |   Autora: Adriana Pereira de Medeiros                                        |
    | ============================================================== -->
<!DOCTYPE OOHDM SYSTEM "../../oohdmXWeb/DTDs/OOHDM_v1_2.dtd"
[
  <!ENTITY modelo_conceitual   SYSTEM "cds_con_model.xml">
  <!ENTITY modelo_navegacao  SYSTEM "cds_nav_model.xml">
  <!ENTITY mapeamento_oohdmweb SYSTEM "cds_oohdmweb.xml">
  <!ENTITY % MS_ACCESS "INCLUDE">
  <!ENTITY % ORACLE  "IGNORE">
]>
<OOHDM name=" Colecao de CDs" version="1.0">
        <!--======= Projeto OOHDM da aplicacao ======= -->
        <OOHDM_BASE>
              <!--***** Definicao do Modelo Conceitual ***** -->
              <conceptual_model>
                    &modelo_conceitual;
              </conceptual_model>
              <!-- ***** Definicao do Modelo de Navegacao *****-->
              <navigational_model>
                    &modelo_navegacao;
              </navigational_model>
              <!--***** Definicao do projeto da Interface Abstrata *****-->
                    <interface_model/>
        </OOHDM_BASE>
        <!-- ===== Mapeamento dos modelos da aplicacao para o ambiente OOHDM-Web ===== -->
        <OOHDM_WEB>
              &mapeamento_oohdmweb;
        </OOHDM_WEB>
</OOHDM>
```

Arquivo: cds_con_model.xml

```
<!-- ================================================================
    |                        Especificação do Modelo Conceitual                    |
    |   * Classes Conceituais (atributos e perspectivas)                           |
    |   * Relacionamentos                                                          |
    |=============================================================-->
   <!-- ================== Classe Conceitual cds ================== -->
   <conceptual_class id="id_cds" name="cds">
         <attrib name="chave_cd" type="string" size="20"/>
```

```
        <attrib name="nome_cd" type="string" size="70"/>
        <attrib name="ano_gravacao_cd" type="string" size="4"/>
        <attrib name="foto_cd" type="image" size="30"/>
        <attrib name="num_copias" type="integer" size="30"/>
        <attrib name="local_gravacao_cd" type="string" size="50"/>
    </conceptual_class>
    <!-- ================== Classe Conceitual artistas ================== -->
    <conceptual_class id="id_artistas" name="artistas">
        <attrib name="chave_artista" type="string" size="20"/>
        <attrib name="nome_artista" type="string" size="50"/>
        <attrib name="descricao_artista" type="text">
            <perspective name="descricao_artista"  type="text" default="yes"/>
            <perspective name="foto_artista" type="image" size="50"/>
        </attrib>
        <attrib name="ano_nasc_artista" type="string" size="4"/>
    </conceptual_class>
    <!-- ==============  Relacionamento artista grava cd ==============  -->
    <relationship id="relacao_grava" name="artista_grava_cd"
        source_class="id_artistas" target_class="id_cds"
        source_cardinality="n" target_cardinality="n">
    </relationship>
```

Arquivo: cds_nav_model.xml

```
<!-- ================================================================
    |                  Especificação do Projeto de Navegação                                  |
    |    * Classes Navegacionais e seus atributos                                             |
    |    * Elos                                                                               |
    |    * Contextos de Navegação                                                             |
    |    * Classes em Contexto                                                                |
    |    * Estruturas de Acesso (índices)                                                     |
    | ================================================================ -->
    <!-- ================== Classe navegacional cds ================== -->
    <navigational_class id="no_cds" name="cds">
        <nav_attrib name="chave_cd" conceptual_class="id_cds"/>
        <nav_attrib name="nome_cd" conceptual_class="id_cds"/>
        <nav_attrib name="ano_gravacao_cd" conceptual_class="id_cds"/>
        <nav_attrib name="foto_cd" conceptual_class="id_cds"/>
    </navigational_class>
    <!-- ================== Classe navegacional artistas ================== -->
    <navigational_class id="no_artistas" name="artistas">
        <nav_attrib name="chave_artista" conceptual_class="id_artistas"/>
        <nav_attrib name="nome_artista" conceptual_class="id_artistas"/>
        <nav_attrib name="ano_nasc_artista" conceptual_class="id_artistas"/>
        <perspective_attrib name="descricao_artista" perspective="descricao_artista"
                            conceptual_attrib="descricao_artista"
                            conceptual_class="id_artistas"/>
        <perspective_attrib name="foto_artista" perspective="foto_artista"
                            conceptual_attrib="descricao_artista"
                            conceptual_class="id_artistas"/>
        <nav_attrib_index name="indice_cds" index="indice_cds_por_artista_H"/>
    </navigational_class>
    <!-- ================== Elo artista grava CD ====================== -->
    <link id="link_grava" name="artista_grava_cd"
        source_class="no_artistas" target_class="no_cds"
```

```
        source_cardinality="n" target_cardinality="n">
</link>
<!-- ================== Contexto CDs em ordem alfabetica ================== -->
<navigational_context id="contexto_cds_alfa" name="cds_alfa"
                element_class="no_cds" type="static"
                navigation_type="circular">
        <order id="ordenacao_cds" default="yes">
              <order_element name="nome_cd" navigational_class="no_cds" criteria="asc"/>
        </order>
</navigational_context>
<!-- ================== Contexto artistas em ordem alfabetica ================== -->
<navigational_context id="contexto_artistas_alfa" name="artistas_alfa"
                element_class="no_artistas" type="static"
                navigation_type="free">
        <order id="ordenacao_artistas" default="yes">
              <order_element name="nome_artista" navigational_class="no_artistas" criteria="asc"/>
        </order>
</navigational_context>
 <!-- ================ Contexto CDs gravados nos anos 1991 e 1992 ================ -->
<navigational_context id="contexto_cds_91_92" name="cds_91_92"
                element_class="no_cds" type="static"
                navigation_type="free">
        <selection>
              <OR>
                <equal>
                      <attribute name="ano_gravacao_cd" navigational_class="no_cds"/>
                      <value>1991</value>
                </equal>
                <equal>
                      <attribute name="ano_gravacao_cd" navigational_class="no_cds"/>
                      <value>1992</value>
                </equal>
              </OR>
        </selection>
        <order  id="ordenacao_cds_91_92_ano">
             <order_element name="ano_gravacao_cd" navigational_class="no_cds" criteria="desc"/>
        </order>
        <order  id="ordenacao_cds_91_92_nome" default="yes">
             <order_element name="nome_cd" navigational_class="no_cds" criteria="asc"/>
        </order>
</navigational_context>
 <!-- ========================= Contexto CDs por ano ========================= -->
 <navigational_context id="contexto_cds_por_ano" name="cds_por_ano"
                element_class="no_cds" type="static"
                navigation_type="sequential">
        <selection>
             <AND>
                <equal>
                      <attribute name="ano_gravacao_cd" navigational_class="no_cds"/>
                      <nav_parameter>ano</nav_parameter>
                </equal>
                <equal>
                      <attribute name="nome_cd" navigational_class="no_cds"/>
                      <nav_parameter>nome</nav_parameter>
                </equal>
             </AND>
        </selection>
```

```
        <order id="ordenacao_cds_por_ano">
            <order_element name="ano_gravacao_cd" navigational_class="no_cds" criteria="asc"/>
        </order>
</navigational_context>
 <!-- ========================= Contexto favoritos ======================== -->
 <navigational_context id="contexto_favoritos" name="favoritos"
                element_class="no_cds no_artistas" type="temporary"
                navigation_type="free">
        <!-- Instancias da classe navegacional CDs -->
        <instance name="instancia_1" navigational_class="no_cds">
            <instance_attrib name="chave_cd">1</instance_attrib>
        </instance>
        <instance name="instancia_2" navigational_class="no_cds">
            <instance_attrib name="chave_cd">2</instance_attrib>
        </instance>
        <instance name="instancia_5" navigational_class="no_cds">
            <instance_attrib name="chave_cd">5</instance_attrib>
        </instance>
       <!-- Instancias da classe navegacional artistas -->
        <instance name="instancia_billy" navigational_class="no_artistas">
            <instance_attrib name="chave_artista">billy</instance_attrib>
        </instance>
        <instance name="instancia_Skank" navigational_class="no_artistas">
            <instance_attrib name="nome_artista">Skank</instance_attrib>
        </instance>
        <!-- Ordenacao para os CDs -->
        <order id="ordenacao_cds_favoritos">
            <order_element name="nome_cd" navigational_class="no_cds" criteria="asc"/>
        </order>
        <!-- Ordenacao para os artistas -->
        <order id="ordenacao_artistas_favoritos">
            <order_element name="nome_artista" navigational_class="no_artistas" criteria="asc"/>
        </order>
</navigational_context>
 <!-- ====================== Contexto CDs do artista Bryan ================== -->
 <navigational_context id="contexto_cds_bryan" name="cds_bryan"
                element_class="no_cds" type="static"
                navigation_type="free">
        <selection>
            <AND>
                <equal>
                    <attribute name="ano_gravacao_cd" navigational_class="no_cds"/>
                    <nav_parameter>ano</nav_parameter>
                </equal>
                <equal>
                    <attribute name="chave_artista" navigational_class="no_artistas"
                                link="link_grava"/>
                    <value>bryan</value>
                </equal>
            </AND>
        </selection>
        <order  id="ordenacao_cds_bryan" default="yes">
            <order_element name="nome_cd" navigational_class="no_cds" criteria="asc"/>
        </order>
</navigational_context>
```

```
<!-- ================ Contexto CDs dos artistas nascidos em 1960 ================ -->
<navigational_context id="contexto_cds_artistas_60" name="cds_artistas_60"
                element_class="no_cds" type="static"
                navigation_type="index">
        <selection>
              <AND>
                <equal>
                    <attribute name="ano_gravacao_cd" navigational_class="no_cds"/>
                    <nav_parameter>ano</nav_parameter>
                </equal>
                <equal>
                    <attribute name="ano_nasc_artista" navigational_class="no_artistas"
                                link="link_grava"/>
                    <value>1960</value>
                </equal>
              </AND>
        </selection>
</navigational_context>
<!-- ======================== Contexto CDs por artista ================ -->
<navigational_context id="contexto_cds_por_artista" name="cds_por_artista"
                element_class="no_cds" type="static"
                navigation_type="free">
        <selection>
              <equal>
                   <attribute name="chave_artista" navigational_class="no_artistas" link="link_grava"/>
                   <nav_parameter>chave_artista</nav_parameter>
              </equal>
        </selection>
        <order  id="ordenacao_cds_por_artista" default="yes">
              <order_element name="nome_cd" navigational_class="no_cds" criteria="asc"/>
        </order>
</navigational_context>
<!-- ======================== Contexto CDs por consulta ================== -->
<navigational_context id="contexto_cds_por_consulta" name="cds_por_consulta"
                element_class="no_cds" type="by_query"
                navigation_type="free"/>
<!-- ====================== Classe em contexto CDs por ano ================ -->
<context_class id="classe_contexto_cds_por_ano" base_class="no_cds"
                  contexts="contexto_cds_por_ano">
        <nav_attrib  name="local_gravacao_cd" conceptual_class="id_cds"/>
</context_class>
 <!-- =================== Indice CDs em ordem alfabetica ================-->
 <index id="indice_cds_alfabetico" name="cds_alfa_idx" type="static">
        <element_context context="contexto_cds_alfa"/>
        <dynamic_selector>
              <shown_attrib  name="nome_cd" navigational_class="no_cds"/>
              <selector_destination  target="contexto_cds_alfa"/>
        </dynamic_selector>
        <dynamic_selector>
              <shown_attrib  name="ano_gravacao_cd" navigational_class="no_cds"/>
              <selector_destination  target="contexto_cds_por_ano"/>
        </dynamic_selector>
</index>
<!-- ================ Indice CDs gravados nos anos 1991 e 1992 ================-->
<index id="indice_cds_91_92" name="cds_91_92_idx" type="static">
        <element_context context="contexto_cds_91_92"/>
```

```
        <shown_attrib  name="ano_gravacao_cd" navigational_class="no_cds"/>
        <dynamic_selector>
              <shown_attrib  name="nome_cd" navigational_class="no_cds"/>
              <selector_destination  target="contexto_cds_91_92"/>
    </dynamic_selector>
</index>
<!-- ================ Indice CDs gravados no ano de 1997 =================-->
<index id="indice_cds_97" name="cds_97_idx" type="static" element_class="no_cds">
        <selection>
              <equal>
                    <attribute name="ano_gravacao_cd" navigational_class="no_cds"/>
                    <value>1997</value>
              </equal>
        </selection>
        <dynamic_selector>
              <shown_attrib  name="nome_cd" navigational_class="no_cds"/>
              <selector_destination  target="contexto_cds_por_ano"/>
        </dynamic_selector>
        <static_selector id="cds_1997">
              <name>CDs_1997</name>
              <selector_destination  target="contexto_cds_por_ano"/>
        </static_selector>
        <order id="ordenacao_cds1997_nome">
              <order_element name="nome_cd" navigational_class="no_cds" criteria="asc"/>
        </order>
        <order id="ordenacao_cds1997_ano">
             <order_element name="ano_gravacao_cd" navigational_class="no_cds" criteria="desc"/>
        </order>
</index>
 <!-- ============== Indice CDs gravados por artistas nascidos em 1960 ===========-->
 <index id="indice_cds_artistas_60" name="cds_artistas_60_idx" type="static">
        <element_context context="contexto_cds_artistas_60"/>
        <shown_attrib  name="ano_gravacao_cd" navigational_class="no_cds"/>
        <dynamic_selector>
              <shown_attrib  name="nome_cd" navigational_class="no_cds"/>
              <selector_destination  target="contexto_cds_artistas_60"/>
        </dynamic_selector>
        <dynamic_selector>
              <shown_attrib  name="nome_artista" navigational_class="no_artistas"/>
              <selector_destination  target="contexto_artistas_alfa"/>
        </dynamic_selector>
</index>
<!-- ======================== Indice CDs do artista Bryan ==================-->
<index  id="indice_bryan" name="cds_bryan_idx" type="static">
        <element_context context="contexto_cds_bryan"/>
        <shown_attrib  name="ano_gravacao_cd" navigational_class="no_cds"/>
        <dynamic_selector>
              <shown_attrib  name="nome_cd" navigational_class="no_cds"/>
              <selector_destination  target="contexto_cds_bryan"/>
        </dynamic_selector>
 </index>
 <!-- ======================== Indice CDs e artistas favoritos ==================-->
 <index  id="indice_favoritos" name="favoritos_idx" type="static">
        <element_context context="contexto_favoritos"/>
        <dynamic_selector>
              <shown_attrib  name="nome_cd" navigational_class="no_cds"/>
```

```
            <selector_destination  target="contexto_favoritos"/>
        </dynamic_selector>
        <dynamic_selector>
            <shown_attrib  name="nome_artista" navigational_class="no_artistas"/>
            <selector_destination  target="contexto_favoritos"/>
        </dynamic_selector>
</index>
<!-- ======================== Indice CDs e artistas favoritos ==================-->
<index  id="indice_cds_por_ano" name="cds_por_ano_idx" type="static">
        <element_context context="contexto_cds_por_ano"/>
        <shown_attrib  name="ano_gravacao_cd" navigational_class="no_cds"/>
        <dynamic_selector>
            <shown_attrib  name="nome_cd" navigational_class="no_cds"/>
            <selector_destination  target="contexto_cds_por_ano"/>
        </dynamic_selector>
</index>
<!-- ======================== Indice CDs por artista ==================-->
<index id="indice_cds_por_artista_S" name="cds_por_artista_S_idx" type="static">
        <element_context context="contexto_cds_por_artista"/>
        <shown_attrib  name="ano_gravacao_cd" navigational_class="no_cds"/>
        <dynamic_selector>
            <shown_attrib  name="nome_artista" navigational_class="no_artistas"/>
            <selector_destination  target="contexto_artistas_alfa"/>
        </dynamic_selector>
        <dynamic_selector>
            <shown_attrib  name="nome_cd" navigational_class="no_cds"/>
            <selector_destination  target="contexto_cds_por_artista"/>
        </dynamic_selector>
        <order id="ordenacao_cds_por_artista_S">
            <order_element name="nome_cd" navigational_class="no_cds" criteria="asc"/>
        </order>
</index>
<!-- ====================== Indice hierarquico cds por arquiteto ===============-->
<hierarch_index id="indice_cds_por_artista_H" name="cds_por_artista_H_idx" type="static">
        <element_context context="contexto_cds_por_artista"/>
        <level>
            <dynamic_selector>
                <shown_attrib name="nome_artista" navigational_class="no_artistas"/>
                <selector_destination target="indice_cds_por_artista_H"/>
            </dynamic_selector>
        </level>
        <level>
            <shown_attrib name="ano_gravacao_cd" navigational_class="no_cds"/>
            <dynamic_selector>
                <shown_attrib name="nome_cd" navigational_class="no_cds"/>
                <selector_destination target="contexto_cds_por_artista"/>
            </dynamic_selector>
        </level>
</hierarch_index>
<!-- ====================== Indice CDs por consulta ====================-->
<index id="indice_cds_por_consulta" name="cds_por_consulta_idx" type="dynamic">
        <element_context context="contexto_cds_por_consulta"/>
        <dynamic_selector>
            <shown_attrib name="nome_cd" navigational_class="no_cds"/>
            <selector_destination target="contexto_cds_por_consulta"/>
        </dynamic_selector></index>
```

Arquivo: cds_oohdmweb.xml

```
<!-- ============================================================
   |            Especificação do Mapeamento do Projeto para o ambiente OOHDM-Web  |
   |    * Mapeamento das Classes Conceituais e seus Atributos                      |
   |    * Mapeamento de Relacionamentos                                            |
   |    * Mapeamento das Classes em Contexto                                       |
   |    * Mapeamento dos Elementos de Interface                                    |
   ============================================================ -->
    <!-- ========== Mapeamento da classe conceitual cds para a tabela cds ========== -->
    <map_conceptual_class  conceptual_class="id_cds">
          <map_attribute id="map_ano_gravacao" name="ano_gravacao_cd"
                         db_attribute="ano_gravacao"/>
   </map_conceptual_class>
    <!-- ======== Mapeamento da classe conceitual artistas para a tabela artistas ======== -->
   <map_conceptual_class conceptual_class="id_artistas">
          <map_attribute id="map_foto_artista" name="foto_artista" db_attribute="foto"/>
          <map_attribute id="map_descricao_artista" name="descricao_artista"
                         db_attribute="descricao"/>
   </map_conceptual_class>
    <!-- ======== Mapeamento da classe em contexto cds_ano para a tabela cds_ano ========-->
   <map_context_class context_class="classe_contexto_cds_por_ano" db_class_ctx="cds_ano">
          <map_attribute id="map_local_gravacao_cd" name="local_gravacao_cd"
                         db_attribute="local_gravacao"/>
   </map_context_class>
    <!-- === Mapeamento do relacionamento Artista grava CD para a tabela  artista_grava_cd === -->
   <map_relationship relationship="relacao_grava" db_relationship="artista_grava_cd">
          <db_source_field>chv_artista</db_source_field>
          <db_destination_field>chv_cd</db_destination_field>
   </map_relationship>
    <!-- === Mapeamento dos objetos de interface para classes de navegacao e indices === -->
   <interface_element target="no_cds cds_1997">cd.html</interface_element>
   <interface_element target="no_artistas">artista.html</interface_element>
   <interface_element target="indice_cds_por_artista_H">cds_artista_idx.html</interface_element>
```

Referências

[Antonacci 2000] Antonacci, M.J. "NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia com Sincronização Temporal e Espacial." Dissertação de Mestrado, Departamento de Informática PUC-Rio, 2000.

[Atzeni 1997] Atzeni, P.; Mecca, G.; Merialdo, P. "To Weave the Web", In Proc. 23rd Conference on Very Large Database, Athens (Greece), Aug. 1997, pp. 206–215.

[Atzeni 1998] Atzeni, P.; Mecca, G.; Merialdo, P. "Design and Maintenance of Data Intensive Web Sites", Proc. Int. Conf. on Extending Database Technology, EDBT98, Valencia (Spain), March 1998, pp. 436–450.

[Atzeni 1998] Atzeni, P.; Mecca, G.; Merialdo, P.; Masci, A.; Sindoni, G. "The Araneus Web-Base Management System", Proc. Int. Conf. Sigmod'98, Exhibits Program, Seattle, June 1998, pp. 544-546.

[Baresi 2000] Baresi, L.; Garzotto, F.; Paolini, P.; Valenti, S. "HDM2000: The HDM Hypertext Design Model Revisited", Tech. Report, Politecnico di Milano, Jan. 2000.

[Bochicchio 1999] Bochicchio, M. A.; Paiano, R.; Paolini, P. "JWeb: an HDM Environment for fast development of Web Applications", In Proccedings of IEEE Multimedia Computing and Systems 1999 (ICMCS'99), Vol. 2, pp. 809-813.

[Bosak 1999] Bosak, J; Bray, T. "XML and the Second-Generation Web", In Scientific American Volume 280, Number 5 (May 1999), pages 89-93.

[Bray 1998] Bray, T. "The Annoted XML 1.0 Specification" (Disponível em: http://www.xml.com/xml/pub/axml/axmlintro.html

[Ceri 2000] Ceri, S.; Fraternali, P.; Bongio, A. "Web Modeling Language (WebML): a modeling language for designing Web sites", WWW9 Conference, Amsterdam, May 2000.

[Fernandez 1998] Fernandez, M. F.; Florescu, D.; Levy, A. Y.; Suciu, D. "Catching the boat with strudel: Experiences with a web-site management system", Proc. Int. Conf. Sigmod'98, Seattle, June 1998, pp. 414-425.

[Fraternali 1998] Fraternali, P.; Paolini P. "A Conceptual Model and a Tool Environment for Developing More Scalable and Dynamic Web Applications", In Proccedings on Extending Database Technology, EDBT98, Valencia (Spain), March 1998, pp. 421–435.

[Fraternali 1999] Fraternali, P. "Tools and Approaches for Developing Data-Intensive Web Applications: a Survey", in ACM Computing Surveys 31(3), 227-263, 1999.

[Garzotto 1993] Garzotto, F.; Paolini, P.; Schwabe, D. HDM – "A Model-Based Approach to Hypertext Application Design". ACM Transactions

on Information Systems 11, 1 (January 1993), 1-26.

[Garzotto 2000] Garzotto, F.; Paolini, P.; Baresi, L. "Supporting Reusable Web Design with HDM-Edit", Tech. Report, Politecnico di Milano, 2000.

[Hester 1997] Hester, A.M.; Borges, R.C.; Ierusalimschy, R. "CGILua: A Multi-Paradigmatic Tool for Creating Dynamic WWW Pages", Proceedings of the XI Brazilian Software Engineering Symposium (SBES'97) p. 347-360, Fortaleza, Brasil, 1997 (disponível em http://www.tecgraf.puc-rio.br/~anna/cgilua/cgilua.ps.gz)

[Ierusalimschy 1996] Ierusalimschy, R.; Figueiredo, L. H.; Celes, W. "Lua - an extensible extension language", Software: Practice & Experience 26 #6 (1996) 635-652. (disponível em http://www.tecgraf.puc-rio.br/lua/).

[Isakowitz 1995] Isakowitz, T.; Stohr, E.; Balasubramanian, P. RMM: A methodology for structuring hypermedia design. Commun. ACM 38, 8 (August 1995), 34-44.

[ISO 1986] ISO (International Organization for Standardization). ISO 8879:1986(E), "Information processing – Text and Office Systems – Standard Generalized Markup Language (SGML)". First Edition – Geneva, 1986-10-15.

[Jacobson 1999] Jacobson, I.; Booch, G.; Rumbaugh, J. "The Unified Software Development Process", Addison-Wesley, 1999.

[Kay 2000] Kay, M. "Saxon package 5.5.1" (disponível em http:users.iclway.co.uk/mhkay/saxon/saxon5-5-1/index.html)

[Lange 1994] Lange, D. "An Object-Oriented Design Method for Hypermedia Information Systems". 1994, 366-375.

[Medeiros 2001] Medeiros, A. P.; Schwabe, D. "Especificação OOHDM-ML". Relatório Técnico – Departamento de Informática da PUC-Rio, Rio de Janeiro, 2001.

[Megginson 2000] Megginson Technologies. SAX 2.0: The Simple API for XML (disponível em www.megginson.com/SAX/)

[Moura 1999] Moura, I.C.R. "Um ambiente para o Suporte ao Projeto e Implementação de Sistemas de Informação baseados na WWW", Dissertação de Mestrado, Departamento de Informática PUC-Rio, 1999.

[OMG 2000] XML Metadata Interchange (XMI) Specification - Object Management Group (disponível em http://www.omg.org/technology/documents/formal/xml_metadata_interchange.htm

[Pizzol 1998] Pizzol, A. M. "Um Framework para a Implementação na WWW de Aplicações Hipermídia modeladas com o OOHDM", Dissertação de Mestrado, Departamento de Informática PUC-Rio, 1998.

[RMS 1998] Rodrigues, R. F.; Muchaluat-Saade, D. C.; Soares, L. F. G.; "Composite Nodes, Contextual Links and Graphical Structural Views on the WWW", Journal of Brazilian Computer Society, special issue on World–Wide Web, Vol. 5, No. 2, November 1998.

[Rossi 1996] Schwabe, D.; Rossi, G. "Um método orientado a objetos para o projeto de Aplicações Hipermídia", Tese de Doutorado, Departamento de Informática PUC-Rio, 1996.

[Rossi 1999] Rossi, G.; Schwabe, D.; Lyardet, F.; "Improving Web Information Systems with Navigational Patterns", The International Journal of Computer and Telecommunications Networking, Elsevier, Maio, 1999 pp. 589-600.(ISBN 0-444-50264-5) (Proc. WWW8).

[Schwabe 1998] Schwabe, D.; Rossi, G. "An Object-Oriented Approach to Web-based Application Design". Theory and Practice of Object Systems (TAPOS) (October 1998), 207-225.

[Soares 2000] Soares, L. F. G. "Modelo de Contextos Aninhados Versão 2.3". Relatório Técnico do Laboratório Telemídia, Departamento de Informática da PUC-Rio, Rio de Janeiro, 2000.

[UML 1997] Booch, G.; Rumbaugh, J.; Jacobson, I. "The Unified Modeling Language for Object-Oriented Development" (http://www.rational.com/uml)

[Vilain 1999] Vilain, P.; Schwabe, D. Notação da Metodologia OOHDM. Versão 1.1 (Abril 1999).

[W3C] World Wide Web Consortium (W3C) – Official site: http://www.w3.org

[W3C 1998] Extensible Markup Language (XML) 1.0 - W3C Recommendation 10-February-1998. (Disponível em: http://www.w3.org/TR/1998/REC-xml-19980210)

[W3C 1999] Extensible Stylesheet Language Transformations (XSLT) 1.0 - W3C Recommendation 16-November-1999. (Disponível em: http://www.w3.org/TR/1999/REC-xslt-19991116)

[W3C 2000a] Extensible Stylesheet Language (XSL) 1.0 – W3C Candidate Recommendation 21-November-2000. (Disponível em: http://www.w3.org/TR/xsl)

[W3C 2000b] Document Object Model (DOM) Level 2 - W3C Recommendation 13-November-2000. (Disponível em: http://www.w3.org/DOM/DOMTR)

[W3C 2001] XML Schema Part 0: Primer - W3C Recommendation 02-May-2001. (Disponível em: http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/)