3
Especificação do mapeamento navegacional SHDM

O mapeamento navegacional é o processo que cria a visão navegacional da aplicação a partir dos dados do modelo conceitual. Neste processo os dados das instâncias conceituais são mapeados em um grafo RDF de instâncias navegacionais, sobre o qual são realizadas as consultas para recuperação de dados dos contextos e índices.

Este grafo é criado pela combinação dos resultados dos mapeamentos de classes navegacionais, atributos simples, atributos como listas e elos. Seus dados são descritos em função de classes e propriedades definidas no *namespace* padrão do esquema navegacional `http://www.tecweb.inf.puc-rio.br/shdm/nSchema#`, ao qual o método associa o prefixo reservado *nSchema* para ser utilizado diretamente nas declarações de consultas de contextos e índices.

Implementações do método podem adotar duas estratégias distintas em relação à criação do grafo de instâncias navegacionais. O grafo pode ser gerado automaticamente em uma etapa de pré-processamento, utilizando-se a ontologia conceitual, as instâncias conceituais e a especificação do mapeamento navegacional. Outra alternativa é não gerar o grafo previamente e, em tempo de execução, traduzir as consultas dos contextos e índices em consultas diretamente sobre instâncias conceituais.

Os mapeamentos que geram o grafo de instâncias navegacionais estão detalhados no decorrer deste capítulo. Como um exemplo desse processo, a Figura 12(b) apresenta o grafo de instâncias navegacionais resultante do mapeamento das instâncias conceituais da Figura 11(b). A Figura 12(a) representa o esquema de classes navegacionais especificado sobre o esquema conceitual da Figura 11(a) segundo o qual as instâncias conceituais foram definidas.

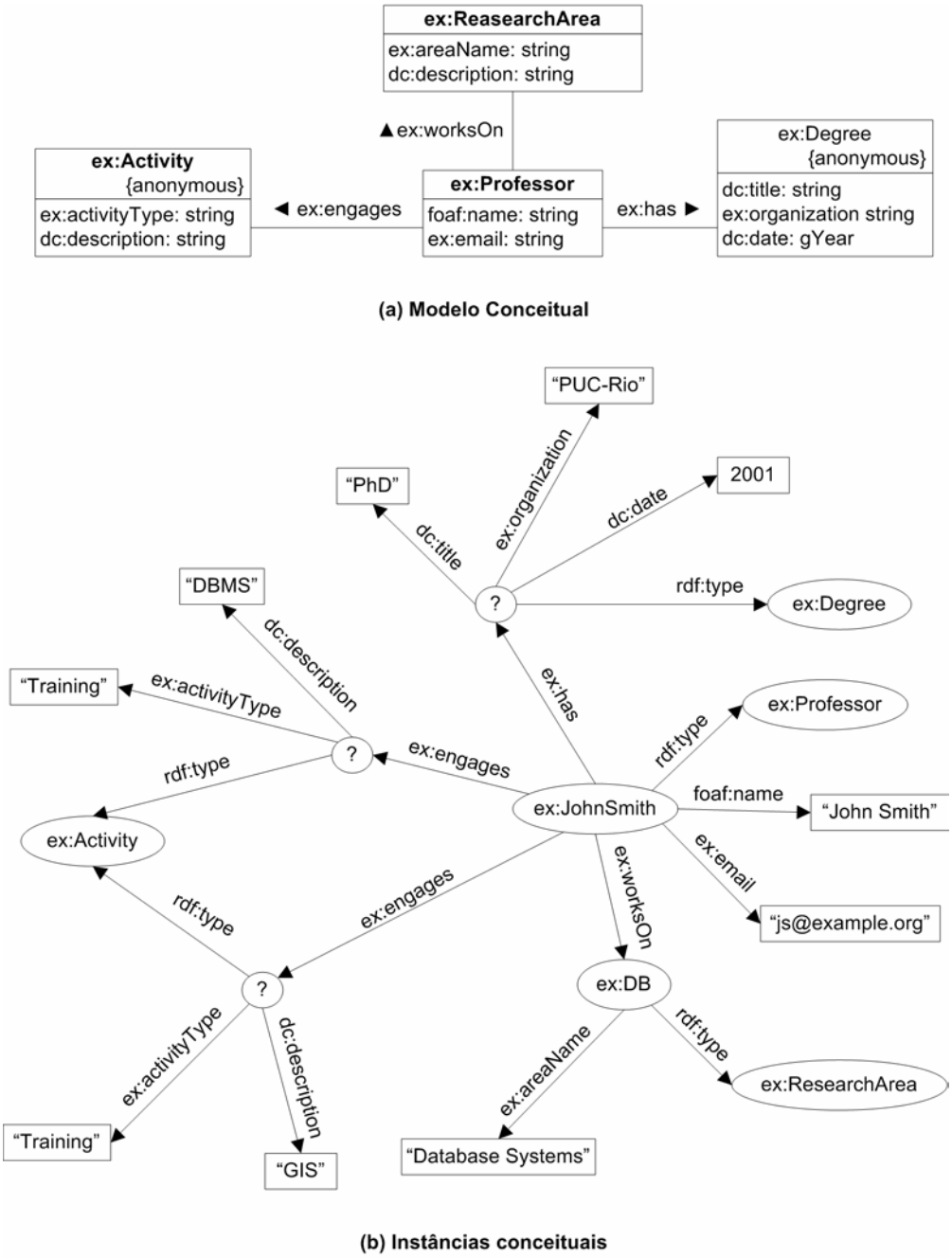

Figura 11 - Exemplo de um grafo RDF de instâncias conceituais

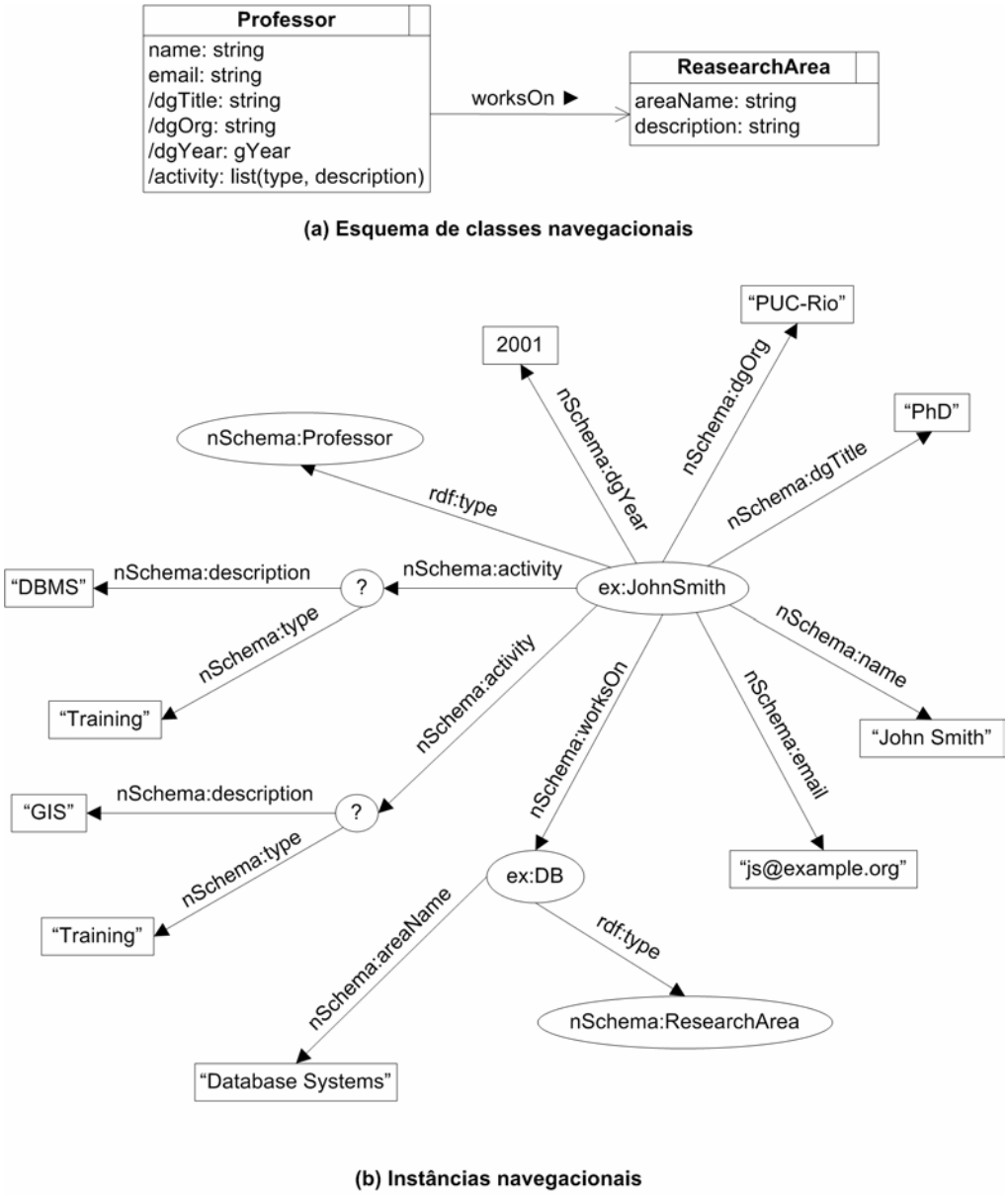

Figura 12 - Exemplo de um grafo RDF de instâncias navegacionais

A especificação do mapeamento navegacional é expressa em RDF utilizando-se classes e propriedades de um vocabulário específico do método. Todos os elementos deste vocabulário são descritos no apêndice "*Vocabulário SHDM"*, e estão definidos no *namespace* `http://www.tecweb.inf.puc-rio.br/2004/06/shdm-schema#`, para o qual utilizamos por convenção o prefixo *shdm*.

3.1
Mapeamento de classes navegacionais

Como foi visto no capítulo anterior, uma classe navegacional é definida através da declaração de um mapeamento de sua classe conceitual base. Considerando isso, o mapeamento de uma classe navegacional é declarado como uma instância de **NavClass** e apresenta obrigatoriamente uma, e somente uma ocorrência da propriedade **baseclass,** cujo valor corresponde à sua classe conceitual base.

Exemplo[9]:

Declaração do mapeamento da classe navegacional *Pessoa*, oriunda da classe conceitual *ex:Pessoa*. Esta é a forma mais simples possível para declaração do mapeamento de uma classe navegacional.

```
:Pessoa
      a shdm:NavClass ;
      shdm:baseclass ex:Pessoa .
```

Quadro 5 - Exemplo de declaração de mapeamento de classe navegacional

3.1.1
Mapeamento de instâncias

O mapeamento de uma classe navegacional estabelece primeiro quais recursos, instâncias da classe base, serão tratados como instâncias da classe navegacional. Este mapeamento também deve garantir que todo recurso mapeado como instância de uma classe navegacional também seja uma instância da classe navegacional especial *nSchema:Node*, ou em outras palavras, que toda classe navegacional seja uma subclasse de *nSchema:Node*. Em um mapeamento como o declarado anteriormente no Quadro 5, todas as instâncias da classe base são consideradas instâncias da classe navegacional. No caso desse exemplo o

[9] Os exemplos de declarações do mapeamento navegacional apresentados ao longo do trabalho estão expressos na sintaxe N3 para representação de dados RDF. Consulte o apêndice *"Notação N3"* para mais detalhes.

mapeamento das instâncias da classe conceitual *ex:Pessoa* em instâncias da classe navegacional *Pessoa* corresponde à seguinte regra[10]:

```
(?x rdf:type ex:Pessoa) -> (?x rdf:type nSchema:Pessoa)
  (?x rdf:type nSchema:Node)
```

Este é o modelo básico de mapeamento de instâncias das classes navegacionais, mas a versão do método SHDM apresentada neste trabalho introduz a capacidade de filtragem de instâncias. Através da declaração de uma regra de filtragem opcional, apenas as instâncias da classe base que satisfaçam as condições estabelecidas nessa regra são mapeadas como instâncias da classe navegacional. Neste caso a regra de mapeamento segue o modelo abaixo, no qual `FILTER` corresponde às condições da regra de filtragem, `BASECLASS` à classe conceitual base e `NAVCLASS` à classe navegacional definida no *namespace* do esquema navegacional.

```
(?x rdf:type BASECLASS) (FILTER) -> (?x rdf:type NAVCLASS)
  (?x rdf:type nSchema:Node)
```

Considere como exemplo do uso deste recurso o caso de uma aplicação para uma loja, cujo modelo conceitual contém produtos (instâncias de *ex:Product*) em diferentes categorias (identificadas pela propriedade *ex:category*), mas que se deseja no modelo navegacional tratar apenas dos produtos relacionados à pesca. Com o recurso de filtragem de instâncias no mapeamento navegacional, podemos definir uma classe navegacional *FishingGear* oriunda *ex:Product* mapeando apenas as instâncias conceituais com o valor *"fishing"* para propriedade *ex:category*. Sem este recurso, a única forma de se obter um efeito semelhante seria através da declaração de uma classe navegacional baseada em *ex:Product*, do mapeamento de *ex:category* como um atributo desta classe, e finalmente a declaração de todos contextos e índices de produtos incluindo apenas os nós cujo valor do atributo derivado de *ex:category* fosse igual a *"fishing"*. A alternativa de declarar uma subclasse no modelo conceitual não é desejável, pois introduz uma

[10] Ao longo do trabalho, os exemplos e modelos das regras de mapeamento estão expressos na sintaxe da linguagem de inferência da API Jena 2 como descrita em Reynolds (2004).

mudança no modelo conceitual devido a um requisito navegacional, violando o principio de separação de interesses ("*concerns*").

A regra de filtragem é declarada através da propriedade **withRule**, cujo valor é a declaração da regra como uma instância de **MappingRule**. O recurso correspondente à regra de filtragem pode ser anônimo, e deve apresentar como valor da propriedade **ruleExpression** a condição de filtragem, como valor da propriedade **ruleVariable** o nome da variável que representa os recursos mapeados, e como valor da propriedade **ruleLanguage** a identificação da linguagem na qual a condição está expressa. Qualquer linguagem pode ser utilizada para expressão da condição. Cabe a cada implementação do método decidir que ação tomar quando não for possível interpretar a linguagem utilizada. Em nossos exemplos utilizamos a sintaxe da linguagem de inferência da API Jena 2[11] como descrita em Reynolds (2004).

Exemplo:

Declaração do mapeamento da classe navegacional *FishingGear*, oriunda da classe conceitual *ex:Product*. Este mapeamento utiliza um filtro estabelecendo que apenas as instâncias de *ex:Product* com um determinado valor para a propriedade *ex:category* devem ser mapeadas como instâncias de *FishingGear*.

```
:FishingGear
      a shdm:NavClass ;
      shdm:baseclass ex:Product ;
      shdm:withRule [
            a shdm:MappingRule ;
            shdm:ruleExpression "(?x
                  <http://www.example.org/category>
                  <http://www.example.org/Fishing>)" ;
            shdm:ruleVariable "?x" ;
            shdm:ruleLanguage "JENA2"
      ] .
```

Quadro 6 - Exemplo de declaração de mapeamento de classe navegacional com regra de filtragem de instâncias

[11] - http://jena.sourceforge.net/

3.1.2
Definição de atributos

A segunda parte da especificação do mapeamento de uma classe navegacional é a definição de seus atributos. A definição de cada atributo é declarada como o valor da propriedade **attribute** na especificação do mapeamento de uma classe navegacional. Os atributos de uma classe navegacional são classificados em quatro tipos (simples, lista, âncora, índice) dependendo da natureza do valor que apresentam; a cada tipo corresponde uma forma de declaração diferente. Estes tipos e suas declarações são discutidos a seguir.

3.1.2.1.
Atributos simples

O valor de um atributo simples é um literal, instância de um dos tipos de *XML Schema*. Este valor é oriundo de uma propriedade que representa um atributo da classe conceitual base ou de uma classe conceitual associada, e por isso é mapeado em função de um caminho de propriedades conceituais, que leva do recurso mapeado como instância da classe navegacional até o valor do atributo. A regra para este mapeamento segue o modelo abaixo, definido para um caminho com duas propriedades, mas que pode ser simplificado ou expandido no caso de caminhos com apenas uma, ou inúmeras propriedades respectivamente. Neste modelo *BASECLASS* corresponde à classe conceitual base da classe navegacional para a qual o atributo está sendo definido, *PROP1* e *PROP2* representam as propriedades conceituais que formam o caminho, e *ATRIB* corresponde à propriedade, definida no *namespace* do esquema navegacional, que corresponde ao atributo definido.

```
(?x rdf:type BASECLASS) (?x PROP1 ?y) (?y PROP2 ?z)
-> (?x ATRIB ?z)
```

Como o valor de todo atributo simples é tratado como um literal, se o caminho declarado levar a um recurso ao invés de um literal, o valor do atributo será o URI de tal recurso (um literal do tipo *xsd:anyURI*). Se o caminho definido mapear mais de um valor, apenas um dos valores mapeados é retornado quando o atributo é requisitado. O valor retornado é selecionado aleatoriamente. Se a

intenção for recuperar todos os valores, o atributo deve ser mapeado como uma lista (vide 3.1.2.2).

O mapeamento de um atributo simples é declarado como uma instância de **MappedAttribute**. O nome do atributo é declarado como o valor da propriedade **attributeName**, e o caminho do mapeamento como o valor da propriedade **fromPath**. Este caminho é especificado como uma seqüência RDF (*rdf:Seq*), na qual cada elemento corresponde a uma propriedade conceitual.

Exemplo:

Considere um modelo conceitual onde o fato de um professor possuir uma titulação é representado por um recurso, instância de *ex:Professor*, apresentando a propriedade *ex:possui* cujo valor é um recurso, instância de *ex:Titulacao*, que por sua vez apresenta uma propriedade *ex:titulo* correspondente ao valor dessa titulação. A seguinte declaração define o mapeamento de uma classe navegacional *Professor*, oriunda de *ex:Professor*, onde a descrição da titulação de um professor é mapeada para o atributo *titulacao* através do caminho formado pelas propriedades *ex:possui* e *ex:titulo*.

```
:Professor
      a shdm:NavClass ;
      shdm:baseclass ex:Professor ;
      shdm:attribute [
            a shdm:MappedAttribute ;
            shdm:attributeName "titulacao" ;
            shdm:fromPath [
                  a rdf:Seq ;
                  rdf:li ex:possui , ex:titulo ;
            ]
      ] .
```

Quadro 7 - Exemplo de declaração de mapeamento de atributo simples

A experiência de modelagem com o método OOHDM mostrou que na maioria dos casos, uma classe navegacional mapeia todos (ou quase todos) os atributos de sua classe conceitual base. Considerando isso e a possibilidade de utilizar informações da ontologia conceitual, foi acrescentado ao método SHDM o recurso de mapeamento automático de atributos simples. Em termos práticos, é como se para cada propriedade da classe conceitual base, fosse declarado

implicitamente um mapeamento de atributo simples. Este recurso ainda inclui a possibilidade de especificação de um filtro de exclusão, que corresponde a uma lista das propriedades para as quais não se deseje o mapeamento automático. Desta forma, quando uma classe navegacional apresenta todos, ou a maioria dos atributos da classe conceitual base, o mapeamento automático pode ser ativado com a filtragem dos atributos indesejados. Já no caso da classe navegacional não apresentar nenhum, ou poucos atributos da classe conceitual base, o mapeamento automático pode ser desativado e a declaração explícita de mapeamento de cada um dos atributos simples utilizada.

O critério empregado na definição deste mapeamento automático varia de acordo com a linguagem na qual a ontologia conceitual está definida. Nas ontologias definidas em OWL são mapeadas todas as *datatype properties* que não constem do filtro de exclusão, e para as quais a classe conceitual base tenha sido declarada como o tipo do domínio. Em ontologias definidas em RDFS, como não é possível fazer a distinção entre propriedades que representam atributos (o valor é um literal) das que representam relacionamentos (o valor é um recurso), são mapeadas indistintamente todas as propriedades que não constem do filtro de exclusão, e para as quais a classe conceitual base tenha sido declarada como o tipo do domínio. Neste caso, para garantir o resultado adequado do mapeamento automático, as propriedades que representam relacionamentos devem ser adicionadas ao filtro de exclusão na especificação do mapeamento.

A ativação do mapeamento automático é declarada através da propriedade **autoMap,** cujo valor deve ser *true* para definição automática do mapeamento e *false* caso contrário. Na ausência de declaração dessa propriedade, *true* é assumido como valor padrão. Os atributos a serem excluídos do mapeamento automático são declarados através da propriedade **autoMapExclude**, cujo valor é uma instância de *rdf:Bag*, onde cada elemento corresponde a uma propriedade conceitual.

Exemplo:

Declaração de mapeamento de uma classe navegacional *Aluno*, que especifica explicitamente o mapeamento automático de atributos de sua classe conceitual base.

```
:Aluno
      a shdm:NavClass ;
      shdm:baseclass ex:Aluno ;
      shdm:autoMap true .
```

Quadro 8 - Exemplo de declaração do mapeamento de classe navegacional com mapeamento automático de atributos simples

Declaração de mapeamento de uma classe navegacional *Professor*, que especifica explicitamente o mapeamento automático filtrando a propriedade conceitual *ex:categoria*.

```
:Professor
      a shdm:NavClass ;
      shdm:baseclass ex:Professor ;
      shdm:autoMap true ;
      shdm:autoMapExclude [
            a rdf:Bag ;
            rdf:li ex:categoria
      ] .
```

Quadro 9 - Exemplo de declaração de mapeamento de uma classe navegacional com filtro de exclusão para mapeamento automático de atributos simples (no exemplo, *ex:categoria*)

3.1.2.2. Listas

Além de um único valor simples, uma classe navegacional pode apresentar atributos cujo valor é uma lista contendo diversos valores simples. Essa lista é formada por uma série de elementos, cada um com inúmeros atributos simples, e todos com o mesmo formato (mesmos atributos).

De forma análoga ao mapeamento de um atributo simples, cada atributo dos elementos de uma lista é mapeado sobre um caminho de propriedades conceituais que leva do recurso mapeado como instância da classe navegacional até o valor do atributo. Estes caminhos são combinados formando a definição de uma árvore, cuja raiz é o recurso e as folhas os valores dos atributos dos elementos. Cada elemento da lista corresponde então a uma ocorrência dessa árvore no grafo de instâncias conceituais.

Considere, por exemplo, um CD como descrito no modelo conceitual mostrado na Figura 13(a).

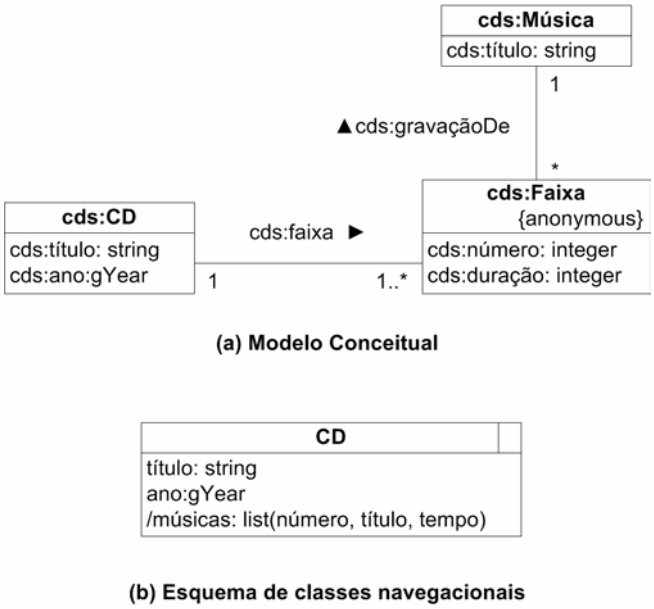

Figura 13 - Exemplo de mapeamento de um atributo como uma lista

A Figura 13(b) mostra um esquema de classes navegacionais onde os atributos *cds:número, cds:duração* da classe conceitual *cds:Faixa* e *cds:título* da classe conceitual *cds:Músicas* foram mapeados respectivamente como os atributos *número, tempo* e *título* da lista correspondente ao valor do atributo *músicas* da classe navegacional *CD*.

A Figura 16 ilustra como a identificação no grafo com dados de uma instância conceitual mostrado na Figura 14, de ocorrências da árvore definida na Figura 15, resulta na seguinte lista:

número	**título**	**Tempo**
3	“Crosstown Traffic”	146
16	“Voodoo Child (Slight Return)”	313

Tabela 4 - Lista com dados das músicas de um CD

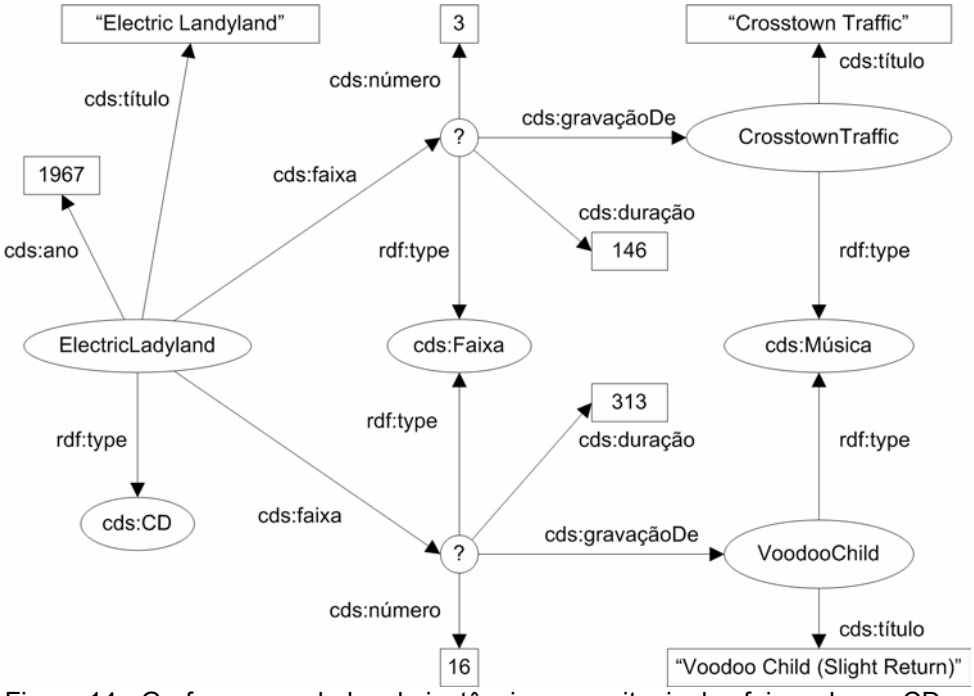

Figura 14 - Grafo com os dados de instâncias conceituais das faixas de um CD

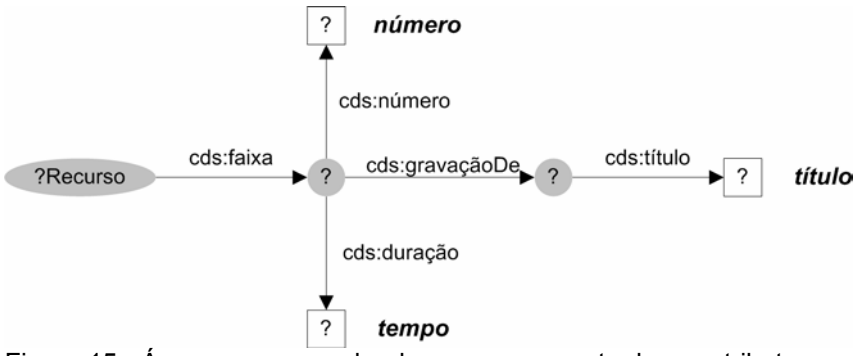

Figura 15 - Árvore correspondendo ao mapeamento de um atributo como uma lista

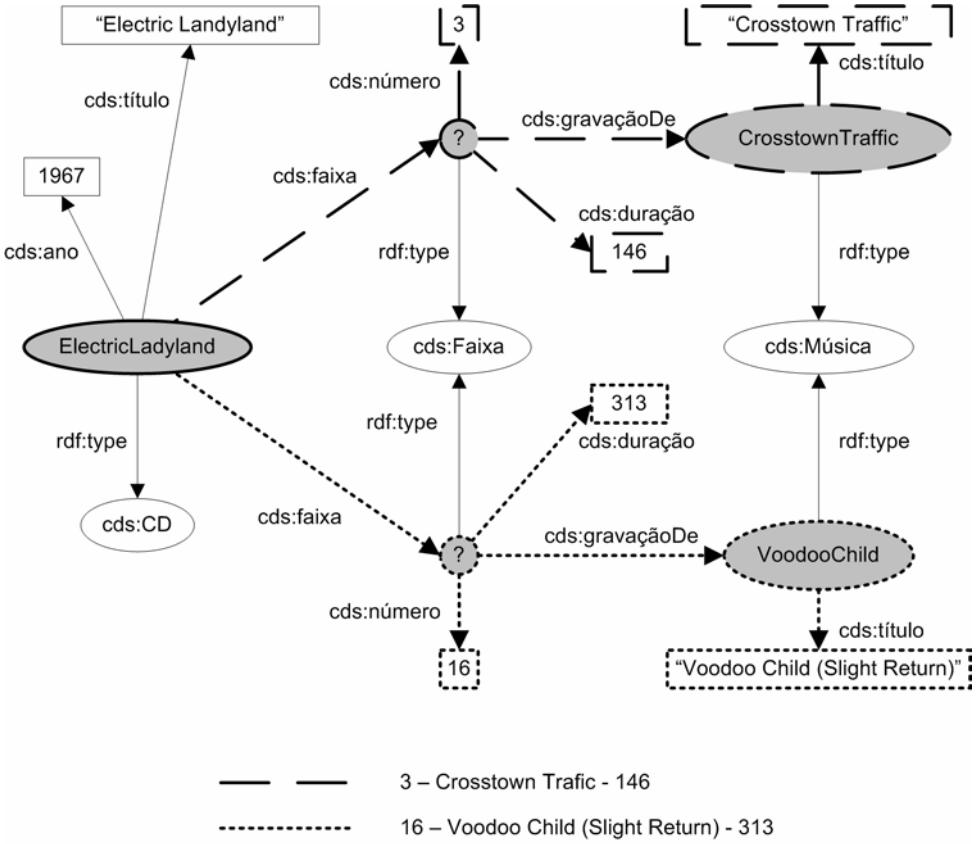

Figura 16 - Identificação dos dados de uma lista em um grafo de instâncias conceituais

No caso do exemplo das figuras 14, 15, e 16, o mapeamento da lista é definido de acordo a seguinte regra:

```
(?r rdf:type cds:CD) (?r cds:faixa ?f) (?f cds:número ?n) (?f
cds:duração ?d) (?f cds:gravaçãoDe ?m) (?m cds:título ?t) ->
(?r nSchema:músicas ?k) (?k nSchema:número ?n) (?k
nSchema:tempo ?d) (?k nSchema:título ?t)
```

Por sua vez, a aplicação dessa regra resulta no grafo de instâncias navegacionais mostrado na figura a seguir.

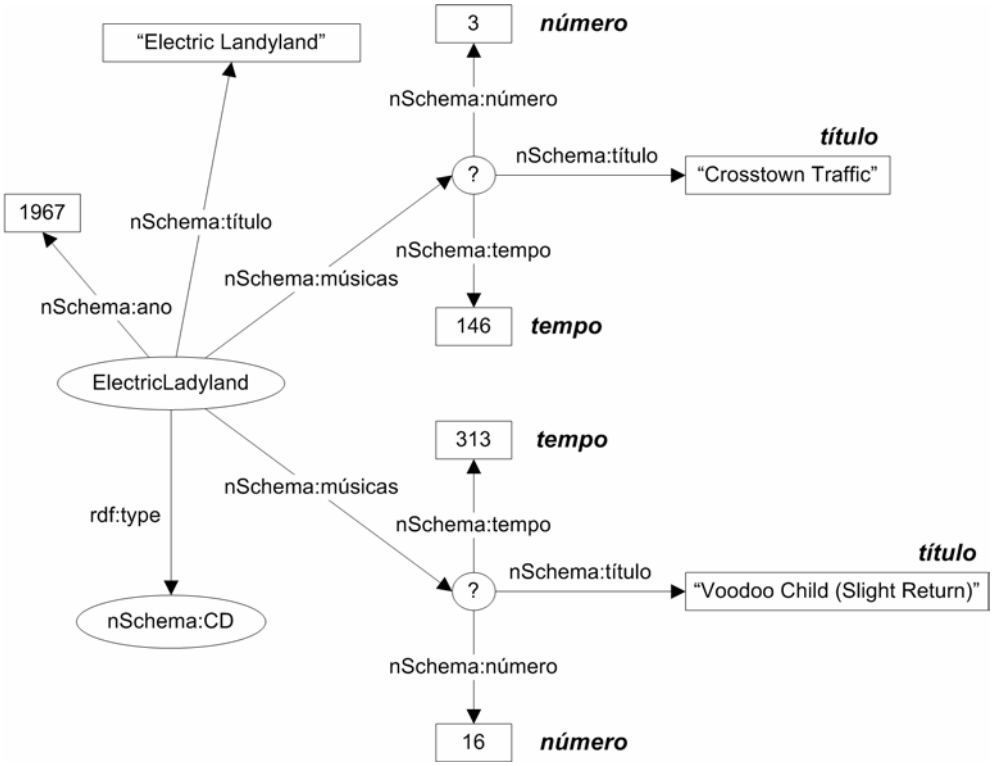

Figura 17 - Grafo de instâncias navegacionais resultante do mapeamento de uma lista

O mapeamento de uma lista é declarado como uma instância de **ListAttribute**, que apresenta o nome do atributo com valor da propriedade **attributeName**, e uma ocorrência da propriedade **attribute** correspondendo à declaração de cada atributo da lista. Os atributos da lista são declarados como instâncias de **MappedAttribute**, da mesma forma que no mapeamento de um atributo simples.

Exemplo:

Declaração do mapeamento de uma classe navegacional *CD*, contendo a especificação do atributo *músicas* como uma lista com três atributos, *número, título* e *tempo*.

```
:CD
      a shdm:NavClass ;
      shdm:baseclass cds:CD ;
      shdm:attribute [
            a shdm:ListAttribute ;
            shdm:attributeName "músicas" ;
            shdm:attribute [
                  a shdm:MappedAttribute ;
                  shdm:attributeName "número" ;
                  shdm:fromPath [
                        a rdf:Seq ;
                        rdf:li cds:faixa , cds:número ;
                  ]
            ] , [
                  a shdm:MappedAttribute ;
                  shdm:attributeName "tempo" ;
                  shdm:fromPath [
                        a rdf:Seq ;
                        rdf:li cds:faixa , cds:duração ;
                  ]
            ] , [
                  a shdm:MappedAttribute ;
                  shdm:attributeName "título" ;
                  shdm:fromPath [
                        a rdf:Seq ;
                        rdf:li cds:faixa ,
                              cds:gravaçãoDe ,
                              cds:título ;
                  ]
            ]
      ] .
```

Quadro 10 - Exemplo de declaração de atributo de uma classe navegacional como uma lista

Assim como ocorre no caso de contextos e índices, os elementos de uma lista podem ser ordenados de acordo com um critério de ordenação baseado no valor de seus atributos.

Todo critério de ordenação é formado por uma seqüência de termos que especificam o nome dos atributos avaliados e a ordem (crescente ou decrescente) de cada um deles. Na ausência da declaração da ordem, o atributo deve ser avaliado por padrão na ordem crescente. Para exemplificar a aplicação de um critério de ordenação, considere a Tabela 5 que mostra o *titulo* e o *ano* de um CD.

título	ano
Sgt Pepper's Lonely Hearts Club Band	1967
Are You Experienced	1967
Electric Ladyland	1968

Tabela 5 - Lista de CDs sem ordenação

Considere agora os seguintes critérios de ordenação: por *titulo* em ordem crescente; por *ano* em ordem decrescente e então por *titulo* em ordem crescente. A aplicação desses dois critérios resulta na ordenação conforme mostrado na Tabela 6 e Tabela 7 respectivamente.

título	ano
Are You Experienced	1967
Electric Ladyland	1968
Sgt Pepper's Lonely Hearts Club Band	1967

Tabela 6 - Lista de CDs ordenados por título

título	ano
Electric Ladyland	1968
Are You Experienced	1967
Sgt Pepper's Lonely Hearts Club Band	1967

Tabela 7 - Lista de CDs ordenados por ano e título

O critério de ordenação de uma lista é definido através da propriedade **ordering**, cujo valor é uma instância de **OrderingCriteria**. Os termos do critério de ordenação são declarados através da definição de uma seqüência RDF. Nesta seqüência cada elemento é uma instância de **OrderingCriteriaTerm**, com as propriedades **attributeName** e **order** correspondendo respectivamente ao atributo

avaliado e à direção de avaliação. Os valores possíveis para **order** são “ASC” ou “DESC” para direção de avaliação crescente ou decrescente respectivamente.

Exemplo:

Declaração do mapeamento de uma classe navegacional *Professor*, contendo a especificação do atributo *atuacao* como uma lista com dois atributos, *tipo* e *detalhe*, ordenada pelos valores de *tipo*.

```
:Professor
      a shdm:NavClass ;
      shdm:baseclass ex:Professor ;
      shdm:attribute [
            a shdm:ListAttribute ;
            shdm:attributeName "atuacao" ;
            shdm:attribute [
                  a shdm:MappedAttribute ;
                  shdm:attributeName "tipo" ;
                  shdm:fromPath [
                        a rdf:Seq ;
                        rdf:li ex:atua , ex:tipo ;
                  ]
            ] , [
                  a shdm:MappedAttribute ;
                  shdm:attributeName "detalhe" ;
                  shdm:fromPath [
                        a rdf:Seq ;
                        rdf:li ex:atua , ex:detalhe ;
                  ]
            ] ;
            shdm:ordering [
                  a shdm:OrderingCriteria ;
                  shdm:orderBy [
                        a rdf:Seq ;
                        rdf:li [
                              a shdm:OrderingCriteriaTerm ;
                              shdm:attributeName "tipo" ;
                              shdm:order "ASC"
                        ]
                  ]
            ]
      ] .
```

Quadro 11 - Exemplo de declaração de atributo de uma classe navegacional como uma lista com ordenação

3.1.2.3. Atributos como âncoras

A definição de um atributo de uma classe navegacional como uma âncora se aplica em dois casos distintos:

- Para criar um elo entre instâncias da classe e um índice qualquer.
- Para representar um elo originado na classe.

Em ambos os casos o atributo é definido como uma instância de **AnchorAttribute**, seu nome é declarado através da propriedade **attributeName** e seu valor especificado pela propriedade **anchorValue**, cujo valor é uma instância de **Anchor**. Na declaração da âncora a propriedade **label** é utilizada para especificar o texto da etiqueta da âncora, a propriedade **target** especifica o contexto ou o índice para o qual a âncora aponta. Os parâmetros passados pela âncora são declarados através de várias ocorrências da propriedade **parameter**, cujo valor deve ser uma instância de **ParameterBinding**, onde o nome do parâmetro é declarado através da propriedade **parameterName**. O valor passado no parâmetro é declarado através da propriedade **parameterValue**. Para uma âncora passar no parâmetro o valor de um dos atributos simples do nó em que ela aparece, o valor desta propriedade deve ser a propriedade do esquema navegacional correspondente a esse atributo simples. Para passar o próprio nó com valor do parâmetro, utiliza-se a palavra reservada *_THIS* como valor de **parameterValue**. Finalmente, para passar um valor arbitrário qualquer, declara-se o mesmo diretamente como valor da propriedade.

No caso do uso da declaração de uma âncora para criação de elos entre instâncias da classe navegacional e um índice, o valor de **label** deve ser o texto da etiqueta e o valor de **target** deve ser obrigatoriamente um recurso correspondendo à declaração de um índice.

Na especificação da âncora para representação de um elo, a declaração do atributo deve apresentar uma ocorrência da propriedade **fromLink,** cujo valor é a referência para o elo em questão. O valor de **target** deve ser um contexto do qual se saiba que o nó referenciado pelo elo participa. Finalmente, de acordo com o

valor que se queira exibir como etiqueta da âncora, o valor de **label** pode ser uma das alternativas apresentadas na tabela a seguir.

Valor da propriedade label	Valor exibido como etiqueta da âncora
Um literal	O literal declarado com valor de **label**
Propriedade correspondente a um atributo simples do objeto que é o destino do elo	O valor do atributo correspondente

Tabela 8 - Alternativas para declaração da etiqueta de um atributo âncora definido sobre um elo

Assim como um atributo simples, um atributo cujo valor é uma âncora retorna apenas um valor. Se o recurso correspondente à instância da classe navegacional apresentar mais de uma ocorrência da propriedade que representa o elo sobre o qual a âncora está definida, apenas uma das ocorrências será utilizada para gerar a âncora. A determinação de qual destas ocorrências será usada é aleatória. Para apresentar todas as ocorrências do elo, deve-se especificar o valor do atributo como um índice.

Exemplo:

Definição do mapeamento de uma classe navegacional aluno, contendo a declaração de um atributo *orientador* como uma âncora sobre o elo *eOrientadoPor*, que exibe como etiqueta o nome do orientador.

```
:Aluno
      a shdm:NavClass ;
      shdm:baseclass ex:Aluno ;
      shdm:attribute [
            a shdm:AnchorAttribute ;
            shdm:attributeName "orientador" ;
            shdm:fromLink nSchema:eOrientadoPor;
            shdm:anchorValue [
                  a shdm:Anchor ;
                  shdm:label nSchema:nome ;
                  shdm:target :ctxProfessorAlfa
            ]
      ] .
```

Quadro 12 - Exemplo de declaração de atributo de classe navegacional como âncora a partir de um elo

Exemplo:

Definição do mapeamento de uma classe navegacional *Artista*, contendo a declaração de um atributo *CDs* como uma âncora para o índice *idxCDPorArtista*. A âncora definida exibe como etiqueta o texto "CDs desse artista" e passa como valor do parâmetro *@artista* a própria instância da classe sendo definida.

```
:Artista
      a shdm:NavClass ;
      shdm:baseclass ex:Artista ;
      shdm:attribute [
            a shdm:AnchorAttribute ;
            shdm:attributeName "CDs" ;
            shdm:anchorValue [
                  a shdm:Anchor ;
                  shdm:label "CDs desse artista" ;
                  shdm:target :idxCDPorArtista ;
                  shdm:parameter [
                        a shdm:ParameterBinding ;
                        shdm:parameterName "@artista";
                        shdm:parameterValue "_THIS"
                  ]
            ]
      ] .
```

Quadro 13 - Exemplo de declaração de atributo de classe navegacional como âncora

3.1.2.4. Atributos como índices

Ao invés de apresentar uma âncora para um índice o método SHDM permite que uma classe navegacional apresente um índice diretamente como o valor de um atributo. Para isso a definição do atributo é declarada como uma instância de **IndexAttribute**, apresentando como valor da propriedade **indexValue** uma referência à definição do índice que será o valor do atributo. Os parâmetros passados para o índice (se for o caso) têm seus valores declarados de forma análoga à declaração dos parâmetros passados em uma âncora.

Exemplo:

Declaração do mapeamento de uma classe navegacional *Professor,* contendo a especificação de um atributo *orientandos*, cujo valor é definido como o índice *idxAlunosPorProfessor*, que recebe como valor do parâmetro *@prof* a própria instância da classe.

```
:Professor
      a shdm:NavClass ;
      shdm:baseclass ex:Professor ;
      shdm:attribute [
            a shdm:IndexAttribute ;
            shdm:attributeName "orientandos" ;
            shdm:indexValue :idxAlunosPorProfessor ;
            shdm:parameter [
                  a shdm:ParameterBinding ;
                  shdm:parameterName "@prof";
                  shdm:parameterValue "_THIS"
            ]
      ] .
```

Quadro 14 - Exemplo de declaração de mapeamento de índice como atributo de uma classe navegacional

3.1.3 Generalização

Hierarquias de classes navegacionais são construídas através da declaração de que o mapeamento de uma classe estende o mapeamento de uma outra classe. Isto é feito pelo uso da propriedade **extends**, cujo valor é o recurso correspondente à declaração de mapeamento da superclasse.

Exemplo:

Declarações dos mapeamentos das classes navegacionais *Professor* e *Pessoa*, estabelecendo que a primeira é uma subclasse da segunda.

```
:Pessoa
      a shdm:NavClass ;
      shdm:baseclass ex:Pessoa .


:Professor
      a shdm:NavClass ;
      shdm:baseclass ex:Professor ;
      shdm:extends :Pessoa .
```

Quadro 15 - Exemplo da declaração de mapeamento de classe navegacional com especialização

3.2
Mapeamento de elos

Um elo é mapeado com base na definição dos tipos de sua origem e destino, e na especificação do caminho de propriedades conceituais que liga os recursos mapeados como instâncias desses tipos. A regra de mapeamento de um elo segue o modelo da regra apresentada abaixo para o mapeamento de um caminho com duas propriedades. Nesse modelo *BASE_O* representa a classe base da classe navegacional que é o tipo da origem do elo, *BASE_D* representa a classe base da classe navegacional que é o tipo do destino do elo, *PROP1* e *PROP2* representam as propriedades conceituais que formam o caminho, e *ELO* corresponde à propriedade, definida no *namespace* do esquema navegacional, que corresponde ao elo definido.

```
(?x rdf:type BASE_O) (?x PROP1 ?y) (?y PROP2 ?z)
(?z rdf:type BASE_D) -> (?x ELO ?z)
```

O mapeamento de um elo é declarado como uma instância de **Link**. Os tipos da origem e destino do elo são declarados respectivamente pelas propriedades **origin** e **destination**, cujos valores devem ser referências para o mapeamento das classes navegacionais correspondentes. A propriedade **fromPath** é utilizada para especificar o caminho de relacionamentos conceituais que será mapeado e a propriedade **invert** para indicar se esse caminho deve ser mapeado no sentido inverso.

Exemplo:

Declaração do mapeamento de um elo *orienta*, ligando instâncias das classes navegacionais *Professor* e *Aluno*, mapeado sobre caminho formado pela propriedade conceitual *ex:orienta.*

```
:orienta
      a shdm:Link ;
      shdm:origin :Professor ;
      shdm:destination :Aluno ;
      shdm:fromPath [
            a rdf:Seq ;
            rdf:li ex:orienta
      ] .
```

Quadro 16 - Exemplo de declaração de mapeamento de elo

Exemplo:

Declaração do mapeamento de um elo *gravadoPor*, ligando as classes navegacionais *CD* e *Artista*, e definido como mapeamento inverso do caminho formado pela propriedade conceitual *ex:grava*.

```
:gravadoPor
      a shdm:Link ;
      shdm:origin :CD ;
      shdm:destination :Artista ;
      shdm:fromPath [
            a rdf:Seq ;
            rdf:li ex:grava
      ] ;
      shdm:invert true .
```

Quadro 17 - Exemplo de declaração de mapeamento inverso de elo

3.3 Classes em contexto

Uma classe em contexto é declarada como uma instância de **InContextClass**, apresentando uma, e somente uma ocorrência das propriedades **decorates** e **inContext,** para declarar respectivamente a classe navegacional decorada, através de uma referência ao seu mapeamento, e o contexto no qual a

classe em contexto deve ser utilizada. Os atributos adicionais da classe em contexto são declarados através do uso da propriedade **attribute**, exatamente da mesma forma em que é empregada nas declarações de mapeamentos de classes navegacionais.

Exemplo:

Declaração de uma classe em contexto *ProfessorCompleto*, que decora uma classe navegacional *Professor,* no contexto *ctxProfessorCompleto*, acrescentando um atributo simples categoria.

```
:ProfessorCompleto
      a shdm:InContextClass ;
      shdm:decorates :Professor ;
      shdm:inContext :ctxProfessorCompleto ;
      shdm:attribute [
            a shdm:MappedAttribute ;
            shdm:attributeName "categoria" ;
            shdm:fromPath [
                  a rdf:Seq ;
                  rdf:li ex:categoria ;
      ] .
```

Quadro 18 - Exemplo de declaração de classe em contexto

3.4 Contextos navegacionais

Todo contexto é declarado como uma instância de **Context**.

O critério de seleção dos elementos de um contexto é definido através da declaração de um conjunto de regras de seleção. Cada regra seleciona nós de um determinado tipo e que satisfaçam uma certa condição (a especificação desta condição é opcional), e o contexto resulta da união dos nós selecionados por todas as regras que compõem o seu critério de seleção.

Uma regra de seleção é declarada como uma instância de **ElementsSelection** e atribuída como valor da propriedade **selects** na declaração do contexto. O recurso correspondente à regra de seleção pode ser anônimo. O tipo dos nós selecionados pela regra é declarado através da propriedade **elementType**. Opcionalmente, utilizam-se as propriedades **condition**, **elementVariable** e **queryLanguage,** para declarar respectivamente, a condição

adicional da regra de seleção, o nome da variável com a qual os nós selecionados são referenciados na condição adicional e a identificação da linguagem de consulta na qual a mesma está expressa. De forma análoga à expressão da condição na regra de filtragem do mapeamento de classes navegacionais, a condição das seleções de elementos nas definições de contextos e estruturas de acesso pode ser expressa em qualquer linguagem de consulta para RDF. Novamente, cabe a cada implementação do método decidir que ação tomar quando não for possível interpretar a linguagem utilizada. Em nossos exemplos foi utilizada a sintaxe da linguagem RDQL.

Exemplo:

Declaração do contexto de todos os professores. Este contexto utiliza apenas uma regra de seleção, sem condição adicional.

```
:ctxProfesssorAlfa
      a shdm:Context ;
      shdm:selects [
            a shdm:ElementsSelection ;
            shdm:elementType nSchema:Professor
      ] .
```

Quadro 19 - Exemplo de declaração de um contexto com uma regra de seleção sem condição adicional

Exemplo:

Declaração do contexto de CDs ou músicas com os dizeres "*Electric Ladyland*" no título. Este contexto é especificado pela declaração de duas regras de seleção diferentes. Uma que recupera nós do tipo *CD* com uma condição adicional sobre o valor do atributo *titulo*, e outra que emprega uma condição semelhante para nós do tipo *Musica*.

```
:ctxElectricLadyland
      a shdm:Context ;
      shdm:selects [
            a shdm:ElementsSelection ;
            shdm:elementType nSchema:CD ;
            shdm:elementVariable "?cd" ;
            shdm:condition "(?cd nSchema:titulo ?tit)
                           AND ?tit LIKE 'Electric
                           Ladyland'" ;
            shdm:queryLanguage "RDQL"
      ] , [
            a shdm:ElementsSelection ;
            shdm:elementType nSchema:Musica ;
            shdm:elementVariable "?musica" ;
            shdm:condition "(?musica nSchema:titulo ?tit)
                           AND ?tit LIKE 'Electric
                           Ladyland'" ;
            shdm:queryLanguage "RDQL"
      ] .
```

Quadro 20 - Exemplo de declaração de um contexto com duas regras de seleção

A parametrização de um contexto permite a definição de grupos de contextos. Esta parametrização implica em utilizar no critério de seleção dos elementos do contexto, valores que são passados através de parâmetros no momento da recuperação do contexto. A definição funciona como se fossem declarados contextos para cada um dos possíveis valores dos parâmetros utilizados. Seguindo este raciocínio, a definição de um grupo de contextos de CDs por artistas, por exemplo, equivale a declaração dos contextos de CDs do Jimi Hendrix, CDs dos Beatles, CDs do Chico Buarque, CDs do Raul Seixas, etc.

Os parâmetros de um contexto são declarados através da propriedade **parameter**, cujo valor é a definição de um parâmetro como uma instância de **ParameterDefinition**. Toda definição de parâmetro de contexto apresenta as propriedades **parameterName**, **parameterType** e **bindsTo**, correspondendo respectivamente ao nome do parâmetro, o seu tipo e a propriedade do nó à qual o mesmo se aplica.

Exemplo:

Declaração do contexto de CDs gravados por um artista.

```
:ctxCDsPorArtista
      a shdm:Context ;
      shdm:parameter [
            a shdm:ParameterDefinition ;
            shdm:parameterName @artista ;
            shdm:parameterType nSchema:Artista ;
            shdm:bindsTo nSchema:grava ;
      ] ;
      shdm:selects [
            a shdm:ElementsSelection ;
            shdm:elementType nSchema:CD ;
            shdm:elementVariable "?cd" ;
            shdm:condition "(@artista nSchema:grava ?cd)" ;
            shdm:queryLanguage "RDQL"
      ] .
```

Quadro 21 - Exemplo de declaração de um contexto parametrizado

Como o critério de seleção dos elementos de contextos facetados é definido em tempo de execução, a declaração desse tipo de contexto dispensa a definição de regras de seleção ou parâmetros. No lugar destas é utilizada apenas uma referência ao grupo de facetas sobre o qual contexto está baseado. Sendo assim, a declaração de um contexto facetado apresenta uma, e somente uma ocorrência da propriedade **onFacets**, cujo valor é uma instância de **FacetsGroup** correspondendo à especificação de um grupo de facetas. Os detalhes da declaração de facetas e grupos de facetas são apresentados mais adiante na seção *3.6-Facetas e grupos de facetas*.

Exemplo:

Exemplo de declaração de um contexto facetado de obras de arte.

```
:ctxObrasDeArteFaceto
      a shdm:Context ;
      shdm:onFacets :facetasDeObrasDeArte .

:facetasDeObrasDeArte
      a shdm:FacetsGroup .
```

Quadro 22 - Exemplo de declaração de um contexto facetado

Por outro lado, tanto contextos facetados quanto contextos que empregam um critério de seleção podem apresentar um padrão de navegação interna. A

declaração dos tipos de navegação que compõe o padrão de navegação interna de um contexto é feita através da propriedade **navigation**, cujos valores possíveis são "FULL" para navegação livre, "CIRC" para navegação circular, "SEQ" para navegação seqüencial e "IDX" para navegação por índice. Caso a navegação por índice seja empregada o índice de contexto associado deve ser declarado através da propriedade **contextIndex**.

Exemplo:

Declaração de um contexto de professores com navegação circular.

```
:ctxProfesssorAlfa
      a shdm:Context ;
      shdm:selects [
            a shdm:ElementsSelection ;
            shdm:elementType nSchema:Professor
      ] ;
      shdm:navigation "CIRC" .
```

Quadro 23 - Exemplo de declaração de um contexto com navegação circular

Exemplo:

Declaração do contexto de CDs gravados por um artista com navegação seqüencial e por índice.

```
:ctxCDsPorArtista
      a shdm:Context ;
      shdm:parameter [
            a shdm:ParameterDefinition ;
            shdm:parameterName @artista ;
            shdm:parameterType nSchema:Artista ;
            shdm:bindsTo nSchema:grava ;
      ] ;
      shdm:selects [
            a shdm:ElementsSelection ;
            shdm:elementType nSchema:CD ;
            shdm:elementVariable "?cd" ;
            shdm:condition "(@artista nSchema:grava ?cd)" ;
            shdm:queryLanguage "RDQL"
      ]
      shdm:navigation "SEQ" , "IDX" ;
      shdm:contextIndex :idxCDsPorArtista .
```

Quadro 24 - Exemplo de declaração de um contexto com navegação seqüencial e por índice

Quando um nó participa de mais de um contexto, ele representa um ponto de interseção entre os contextos de que participa. Isso cria a possibilidade que quando se esteja acessando um nó em um contexto, seja possível a qualquer momento passar a seguir a navegação interna de outro contexto do qual esse nó faça parte. Mas para que essa possibilidade possa ser explorada na prática, é necessária a declaração explícita das interseções de contextos válidas. Isso é feito através da declaração de instâncias de **ContextsIntersection**, que podem ser recursos anônimos, e que devem apresentar uma ocorrência da propriedade **contexts** cujo valor é um *rdf:Bag* com referências a todos os contextos para os quais esse tipo de navegação será permitida.

Exemplo:

Declaração de interseção entre os contextos de CDs por artistas e de todos os CDs (em ordem alfabética, por exemplo). O exemplo, desta declaração indica que será possível, ao se navegar em um dado CD no contexto “CDs por Artista”, seguir tanto para o próximo CD deste contexto quanto para o próximo “CD em ordem alfabética”.

```
[
      a shdm:ContextsIntersection ;
      shdm:contexts [
            a rdf:Bag ;
            rdf:li :ctxCDsPorArtista ;
            rdf:li :ctxCDs
      ]
] .
```

Quadro 25 - Exemplo de declaração de interseção de contextos

O critério de ordenação padrão de um contexto é definido através da propriedade **defaultOrdering**. Vale ressaltar que um contexto pode apresentar apenas um critério de ordenação padrão. Critérios de ordenação adicionais são definidos através de sucessivas ocorrências da propriedade **ordering**.

Assim como ocorre com atributos mapeados como listas o critério de ordenação é declarado como uma instância de **OrderingCriteria**. Mas nesse caso o recurso correspondente ao critério de ordenação não pode ser um recurso anônimo, de forma a permitir a identificação sem ambigüidade do critério selecionado pelo usuário. Além disso, a propriedade **rdfs:label** deve ser utilizada para especificar um nome descritivo para apresentação na interface da aplicação.

Exemplo:

Declaração de um contexto de professores com um critério de ordenação padrão, que especifica a ordenação segundo o nome dos professores selecionados.

```
:ctxProfesssor
      a shdm:Context ;
      shdm:selects [
            a shdm:ElementsSelection ;
            shdm:elementType nSchema:Professor
      ] ;
shdm:navigation "CIRC" ;
defaultOrdering :ordenacaoPorNome .


: ordenacaoPorNome
      a shdm:OrderingCriteria ;
      rdfs:label "Por nome" ;
      shdm:orderBy [
            a rdf:Seq ;
            rdf:li [
                  a shdm:OrderingCriteriaTerm ;
                  shdm:attributeName "nome" ;
                  shdm:order "ASC"
            ]
      ] .
```

Quadro 26 - Exemplo de declaração de um contexto com critério de ordenação padrão

3.5 Estruturas de acesso

Estruturas de acesso são declaradas como instâncias de **Index** e têm a ordenação de seus elementos especificada da mesma forma que nas declarações de contextos navegacionais. Além disso, a definição de toda estrutura de acesso deve incluir a especificação dos atributos de suas entradas. Um atributo de uma entrada é declarado como o valor da propriedade **attribute**, podendo ser uma âncora, definida com uma instância de **AnchorAttribute**, ou um valor simples especificado como uma instância de **SimpleAttribute**. A seguir, são apresentados os detalhes pertinentes às declarações de cada um dos tipos de índices.

3.5.1 Índices derivados de consulta

A consulta do índice é definida pela propriedade **query**, cujo valor é uma instância de **IndexQuery**. As variáveis da consulta são declaradas com múltiplas ocorrências da propriedade **variable**. A condição de seleção de dados é definida como o valor da propriedade **condition**, e a linguagem na qual essa condição está expressa é definida pela propriedade **queryLanguage**.

Exemplo:

Declaração de um índice de alunos derivado de consulta, no qual cada entrada exibe dois atributos, o primeiro sendo uma âncora com o nome do aluno, o segundo um atributo simples com o seu e-mail.

```
:idxAlunos
      a shdm:Index ;
      shdm:query [
            a shdm:IndexQuery ;
            shdm:variable "?aluno" , "?nome", "?email" ;
            shdm:condition "(?aluno rdf:type nSchema:Aluno)
                        (?aluno nSchema:nome ?nome)
                        (?aluno nSchema:email ?email)" ;
            shdm:queryLanguage "RDQL"
      ] ;
      shdm:attribute [
            a shdm:AnchorAttribute ;
            shdm:attributeName "nome" ;
            shdm:anchorValue [
                  a shdm:Anchor ;
                  shdm:label "?nome" ;
                  shdm:target :ctxAlunoAlfa ;
                  shdm:parameter [
                        a shdm:parameterBinding ;
                        shdm:parameterName "_ID" ;
                        shdm:parameterValue "?aluno"
                  ]
            ]
      ] , [
            a shdm:SimpleAttribute ;
            shdm:attributeName "email" ;
            shdm:simpleValue "?email"
      ] .
```

Quadro 27 - Exemplo de declaração de um índice derivado de consulta com dois atributos

Assim como os contextos que empregam um critério de seleção, um índice derivado de consulta pode receber parâmetros cujos valores são utilizados na recuperação dos dados do índice. A declaração desses parâmetros segue o mesmo modelo utilizado para declaração de parâmetros de contextos.

Exemplo:

Declaração de um índice de alunos por professor que recebe a identificação de um professor como parâmetro, utilizando-a para gerar as entradas do índice com os dados dos alunos desse professor.

```
:idxAlunosPorProfessor
      a shdm:Index ;
      shdm:parameter [
            a shdm:ParameterDefinition ;
            shdm:parameterName "@prof" ;
            shdm:parameterType nSchema:Professor ;
            bindsTo nSchema:orienta
      ] ;
      shdm:query [
            a shdm:IndexQuery ;
            shdm:variable "?aluno", "?nome" ;
            shdm:condition "(?aluno rdf:type nSchema:Aluno)
                        (?aluno nSchema:nome ?nome)
                        (@prof nSchema:orienta ?aluno)" ;
            shdm:queryLanguage "RDQL"
      ] ;
      shdm:attribute [
            a shdm:AnchorAttribute ;
            shdm:attributeName "nome" ;
            shdm:anchorValue [
                  a shdm:Anchor ;
                  shdm:label "?nome" ;
                  shdm:target :ctxAlunoPorProfessor ;
                  shdm:parameter [
                        a shdm:ParameterBinding ;
                        shdm:parameterName "_ID" ;
                        shdm:parameterValue "?aluno"
                  ]
            ]
      ] .
```

Quadro 28 - Exemplo de declaração de um índice derivado de consulta parametrizado

3.5.2 Índices derivados de contexto

A declaração de um índice derivado de contexto deve apresentar uma, e somente uma ocorrência da propriedade **fromContext**, correspondendo à definição do contexto sobre qual o índice está sendo definido.

Os parâmetros do índice são os mesmos de seu contexto e por isso não precisam ser declarados. Além disso, como esses parâmetros são repassados para o contexto implicitamente, a definição do atributo correspondente à âncora para

um nó nesse contexto dispensa as declarações para passagem de parâmetros, assim como especificação do destino da âncora, que é assumido como sendo o contexto em questão.

Para exibir dados dos nós do contexto nas entradas do índice, basta utilizar as propriedades correspondentes aos seus atributos como os valores das propriedades **label** ou **simpleValue** nas especificações das âncoras e atributos simples do índice.

Exemplo:

Declaração de um índice de professores, derivado de um contexto de professores, e que apresenta em cada uma de suas entradas uma âncora para um professor nesse contexto. A âncora em questão exibe como etiqueta o nome do respectivo professor.

```
:idxProfessores
      a shdm:Index ;
      shdm:fromContext :ctxProfessorAlfa ;
      shdm:attribute [
            a shdm:AnchorAttribute ;
            shdm:attributeName "nome" ;
            shdm:anchorValue [
                  a shdm:Anchor ;
                  shdm:label nSchema:nome
            ]
      ] , [
            a shdm:SimpleAttribute ;
            shdm:attributeName "email" ;
            shdm:simpleValue nSchema:email
      ] .
```

Quadro 29 - Exemplo de declaração de um índice derivado de contexto

3.5.3 Índices arbitrários

O conteúdo de um índice arbitrário é especificado através da propriedade **entries**, cujo valor deve ser uma instância de *rdf:Bag* contendo instâncias de **IndexEntry** representando as entradas do índice. Os recursos correspondentes às entradas não podem ser anônimos. Os atributos de cada entrada são declarados da mesma maneira como para os tipos de índice previamente definidos. A criação de

mais de um nível de entradas é feita através do uso da propriedade **entries** dentro da declaração de uma entrada, permitindo dessa forma a declaração de suas entradas subordinadas.

Exemplo:

Declaração de um índice correspondendo ao menu principal de uma aplicação do *website* de um departamento acadêmico. O menu apresenta três entradas, uma delas contendo uma âncora para um índice de alunos, outra apresentando uma âncora para um índice de professores e outra com uma âncora para um índice de áreas de pesquisa.

```
:idxMenuPrincipal
      a shdm:Index ;
      shdm:entries [
            a rdf:Bag ;
            rdf:li :entryAlunos , :entryProfessores,
              :entryAreas
      ] .

:entryAlunos
      a shdm:IndexEntry ;
      shdm:attribute [
            a shdm:AnchorAttribute ;
            shdm:attributeName "linkAlunos" ;
            shdm:anchorValue [
                  a shdm:Anchor ;
                  shdm:label "Alunos" ;
                  shdm:target :idxAlunos
            ]
      ] .

:entryProfessores
      a shdm:IndexEntry ;
      shdm:attribute [
            a shdm:AnchorAttribute ;
            shdm:attributeName "linkProfessores" ;
            shdm:anchorValue [
                  a shdm:Anchor ;
                  shdm:label "Professores" ;
                  shdm:target :idxProfessores
            ]
      ] .

:entryAreas
      a shdm:IndexEntry ;
      shdm:attribute [
            a shdm:AnchorAttribute ;
            shdm:attributeName "linkAreas" ;
            shdm:anchorValue [
                  a shdm:Anchor ;
                  shdm:label "Áreas de pesquisa" ;
                  shdm:target :idxAreasDePesquisa
            ]
      ] .
```

Quadro 30 - Exemplo de declaração de um índice arbitrário

3.5.4 Índices facetados

A especificação de um índice facetado requer a definição do grupo de facetas associado ao índice, o que é feito através da propriedade **onFacets**. Nas declarações de âncoras que apontem para contextos baseados no mesmo grupo de facetas que o índice, não há necessidade de especificação dos parâmetros da âncora. E, assim como ocorre em índices derivados de contexto, os atributos de

um índice facetado podem ser definidos em função dos dados dos nós selecionados pelas facetas do índice.

Exemplo:

Declaração de um índice facetado de obras de arte, que apresenta em cada uma de suas entradas uma âncora para uma obra de arte em um contexto facetado, baseado no mesmo grupo de facetas. A âncora em questão exibe como etiqueta o título da respectiva obra.

```
:idxObrasDeArteFacetado
      a shdm:Index ;
      shdm:onFacets :facetasDeObrasDeArte ;
      shdm:attribute [
            a shdm:AnchorAttribute ;
            shdm:attributeName "tituloObra" ;
            shdm:anchorValue [
                  a shdm:Anchor ;
                  shdm:label nSchema:titulo ;
                  shdm:target :ctxObrasDeArteFacetado
            ]
      ] .
```

Quadro 31 - Exemplo de declaração de um índice facetado

3.6
Facetas e grupos de facetas

Como visto anteriormente, uma faceta é definida em função de um atributo simples ou elo de uma classe navegacional, e um grupo de facetas reúne facetas definidas para uma mesma classe de objetos.

Um grupo de facetas é declarado com uma instância de **FacetsGroup**. Nesta declaração a propriedade **isComposedBy** é utilizada para definição de uma lista com as facetas que fazem parte do grupo. Esta lista é declarada como uma instância de *rdf:Bag*, onde cada um dos elementos corresponde a uma faceta, declarada como uma instância de **Facet**. Todas as facetas em um mesmo grupo devem ser definidas em função da mesma classe navegacional.

Exemplo:

Exemplo de declaração de um grupo de facetas *facetasDeObrasDeArte* contendo quatro facetas, *facetaRegiao*, *facetaEstilo*, e *facetaAutor*, *facetaTipo*.

```
:facetasDeObrasDeArte
      a shdm:FacetsGroup ;
      shdm:isComposedBy [
            a rdf:Bag ;
            rdf:li :facetaRegiao ;
            rdf:li :facetaEstilo ;
            rdf:li :facetaAutor ;
            rdf:li :facetaTipo
      ] .
```

Quadro 32 - Exemplo de declaração de um grupo de facetas

As combinações de valores de facetas que são inválidas dentro de um grupo são declaradas através de ocorrências da propriedade **invalidCombination.** O valor desta propriedade deve ser uma instância de *rdf:Bag*, cujos elementos devem ser recursos que apresentem as propriedades **invalidFacet** e **withValue**, correspondendo respectivamente a uma faceta e o valor da mesma que é inválido na combinação.

Exemplo:

Especificação de um grupo de facetas, declarando como inválida a combinação dos valores “Egypt” e “Barroque”, respectivamente nas facetas de região e estilo de uma obra de arte.

```
:facetasDeObrasDeArte
      a shdm:FacetsGroup ;
      shdm:isComposedBy [
            a rdf:Bag ;
            rdf:li :facetaRegiao ;
            rdf:li :facetaEstilo ;
            rdf:li :facetaAutor ;
            rdf:li :facetaTipo
      ] ;
      shdm:invalidCombination [
            a rdf:Bag ;
            rdf:li [
                  shdm:invalidFacet :facetaRegiao ;
                  shdm:withValue "Egypt"
            ] ;
            rdf:li [
                  shdm:invalidFacet :facetaEstilo ;
                  shdm:withValue "Barroque"
            ]
      ] .
```

Quadro 33 - Exemplo de declaração de combinações inválidas de valores de facetas

Uma faceta é definida como uma instância de **Facet**. A classe navegacional para a qual a faceta está sendo definida é especificada através da propriedade **onClass**. A definição do atributo ou elo utilizado na classificação da faceta é feita com a declaração da propriedade **onProperty**, cujo valor deve ser a propriedade correspondente ao atributo ou elo.

Quando os valores da faceta não são declarados explicitamente, são utilizados todos os valores definidos nas instâncias navegacionais como valores da propriedade sobre a qual a faceta foi definida. No caso de facetas definidas sobre elos, para se evitar a exibição do URI do recurso que é o valor da propriedade correspondente ao elo, pode ser declarado a exibição do valor de uma propriedade associada a esse recurso e que corresponda a um atributo do nó apontado pelo elo. Isto é feito com uma declaração da propriedade **displayProperty**.

Exemplo:

Definição de uma faceta de estilos de obras de arte, especificada sobre o atributo *estilo*. Os valores dessa faceta serão todos os valores que aparecem nas instâncias de *ObraDeArte*.

```
:facetaEstilo
      a shdm:Facet ;
      shdm:onClass nSchema:ObraDeArte ;
      shdm:onProperty nSchema:estilo .
```

Quadro 34 - Exemplo de declaração de uma faceta sobre um atributo de uma classe navegacional

Exemplo:

Exemplo de declaração de uma faceta de autores de uma obra de arte, definida sobre o elo *criadaPor* que liga instâncias de *ObraDeArte* aos recursos que representam os seus criadores. Esta declaração especifica que o nome dos autores deve ser utilizado na apresentação dos valores da faceta.

```
:facetaAutor
      a shdm:Facet;
      shdm:onClass nSchema:ObraDeArte ;
      shdm:onProperty nSchema:criadaPor ;
      shdm:displayPropery nSchema:nome .
```

Quadro 35 - Exemplo de declaração de uma faceta sobre um elo de uma classe navegacional

A declaração dos valores de uma faceta é feita através da propriedade **facetValues**, cujo valor é uma composição de valores declarada como uma instância de **FacetValuesComposite**. Essa composição de valores apresenta como valor da propriedade **isComposedBy**, uma seqüência RDF correspondendo a uma lista com os valores da faceta. Para criação de uma hierarquia de valores, basta definir algum valor dessa lista como sendo uma composição de valores.

Exemplo:

Esta declaração especifica uma faceta de regiões para obras de arte, definindo uma hierarquia de valores sobre o país de origem da obra. Esta declaração corresponde ao exemplo de uso de uma hierarquia de valores de facetas ilustrada na Figura 10(a).

```
:facetaRegiao
      a shdm:Facet ;
      shdm:onClass shdm:ObraDeArte ;
      shdm:onProperty shdm:pais ;
      shdm:facetValues [
            a shdm:FacetValuesComposite ;
            shdm:isComposedBy [
                  a rdf:Seq ;
                  rdf:li :Africa , :Asia , :Europe
            ]
      ] .

:Africa
      a shdm:FacetValuesComposite ;
      rdfs:label "Africa" ;
      shdm:isComposedBy [
            a rdf:Seq ;
            rdf:li "Egypt" , "Kenya" , "Morocco"
      ] .

:Asia
      a shdm:FacetValuesComposite ;
      rdfs:label "Asia" ;
      shdm:isComposedBy [
            a rdf:Seq ;
            rdf:li "China" , "India" , "Japan"
      ] .

:Europe
      rdfs:label "Europe" ;
      a shdm:FacetValuesComposite ;
      shdm:isComposedBy [
            a rdf:Seq ;
            rdf:li "France" , "Germany" , "Italy" , "Spain"
      ] .
```

Quadro 36 - Exemplo de declaração de uma faceta com a especificação de uma hierarquia de valores

Para definir uma faceta em função da hierarquia de subclasses da classe navegacional para qual a faceta está sendo definida, basta definir a propriedade *rdf:type* como sendo o valor da propriedade **onProperty** na declaração da faceta. Neste tipo de declaração a propriedade **facetValues** não pode ser utilizada para especificar valores da faceta. A hierarquia de classes é recuperada diretamente do esquema de classes navegacionais implícito.

Exemplo:

Exemplo de declaração de uma faceta que apresenta como valores, a hierarquia de subclasses da classe *Publication*.

```
: facetaTipo
      a shdm:Facet ;
      shdm:onClass nSchema:ObraDeArte ;
      shdm:onProperty rdf:type .
```

Quadro 37 - Exemplo de declaração de uma faceta em função de uma hierarquia de classes

3.7 Landmarks

Um *landmark* é declarado como uma instância de **Landmark** apresentando obrigatoriamente uma, e somente uma ocorrência da propriedade **landmarkAnchor**, cujo valor corresponde à declaração da âncora correspondente.

Exemplo:

Declaração de um *landmark* que define uma âncora para a estrutura de acesso utilizada como menu principal da aplicação.

```
:lmkMenuPrincipal
      a shdm:Landmark ;
      shdm:landmarkAnchor [
            a shdm:Anchor ;
            shdm:label "Menu pricipal" ;
            shdm:target :idxMenuPrincipal
      ] .
```

Quadro 38 - Exemplo de declaração de um *landmark*

3.8 Operações

A assinatura de uma operação é definida em função da especificação dos valores que devem ser fornecidos para sua execução (parâmetros de entrada) e dos valores retornados como resultado dessa execução. Uma operação é especificada como uma instância de **Operation** e seus parâmetros declarados como instâncias de **ParameterDefinition**, que define o nome (**parameterName**) e o tipo (**parameterType**) do parâmetro. Os parâmetros de entrada são definidos através de ocorrências da propriedade **input** e os parâmetros de saída através de

ocorrências da propriedade **output**. Também pode ser incluída uma descrição textual do propósito da operação utilizando-se a propriedade *rdfs:comment*.

Exemplo:

Declaração de uma operação para adicionar um produto a um carrinho de compras em uma aplicação de loja virtual. A operação recebe como parâmetro de entrada o produto que será adicionado, retornando a quantidade deste produto que já encontra no carrinho.

```
:colocarNoCarrinho
      a shdm:Operation ;
      shdm:input [
            a shdm:ParameterDefinition ;
            shdm:parameterName "produto" ;
            shdm:parameterType nSchema:Produto
      ] ;
      shdm:output [
            a shdm:ParameterDefinition ;
            shdm:parameterName "qtd" ;
            shdm:parameterType xsd:integer
      ] ;
      rdfs:comment "Adiciona um produto ao carrinho de
                    compras " .
```

Quadro 39 - Exemplo de declaração de assinatura de uma operação