Bases de dados semi-estruturadas, XML

Fernando Lobo

Base de Dados, Universidade do Algarve

Modelo de informação semi-estruturado

- Esquema está implícito nos dados.
- Ao invés do modelo relacional, não é obrigatório a existência de um esquema para a BD.
- Muito em voga nos últimos anos com XML.

Modelo de informação semi-estruturado

- Os dados são auto-descritivos.
- Podemos forçar a existência de um esquema se assim o desejarmos.
- É mais flexível que o modelo relacional.
- As linguagens para manipular os dados não são tão eficientes como no modelo relacional.

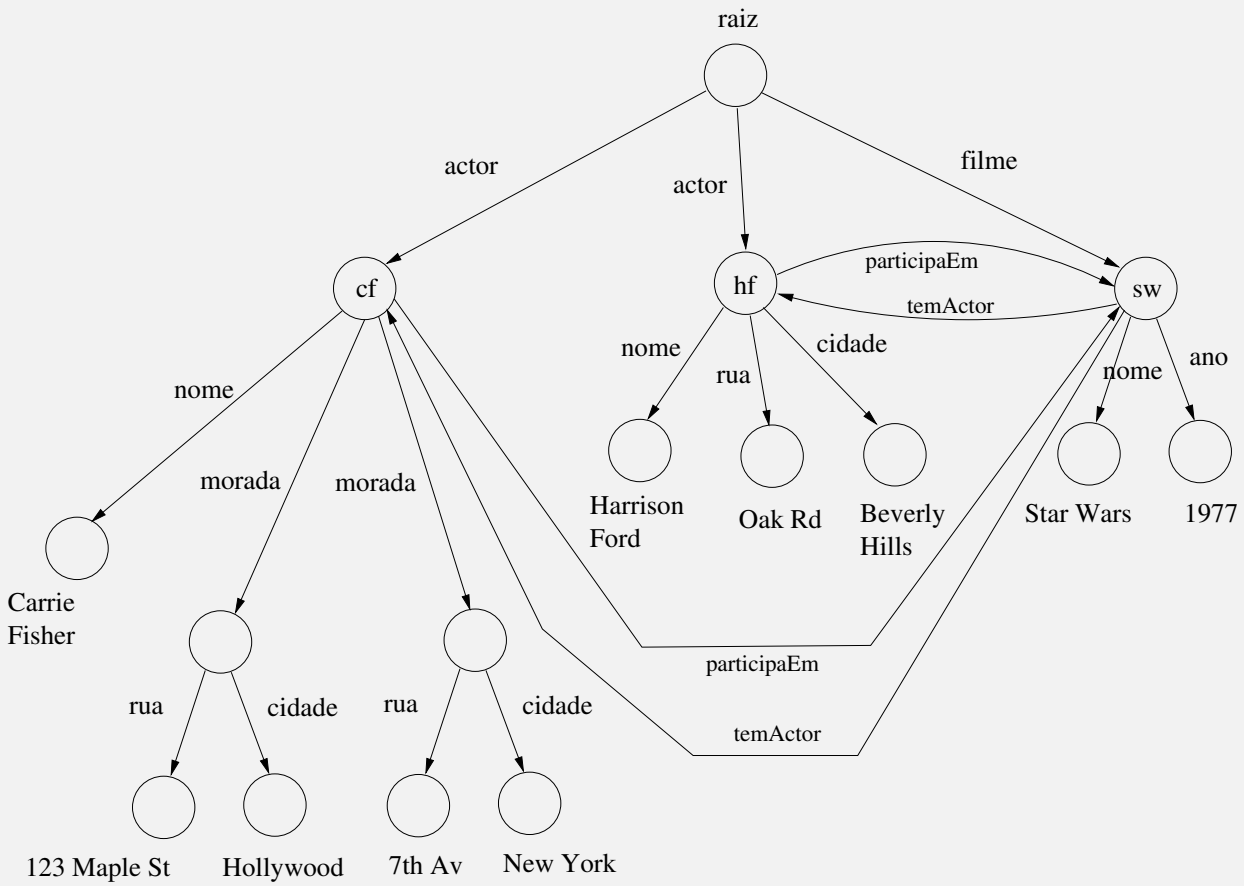
Representação

- A BD é um grafo, uma colecção de nós e arcos entre nós.
- Um nó pode ser um nó interior ou uma folha.
- As folhas têm informação associada (inteiros, strings, datas, etc.)
- Os nós interiores têm 1 ou mais arcos para outros nós.

Nós interiores

- Cada arco tem uma anotação que indica como o nó de origem se relaciona com o nó destino.
- Existe um nó especial, a raíz, que não tem arcos a apontar para ele, e que representa a BD inteira.
- Todo o nó pode ser alcançado através de um caminho a partir da raíz.

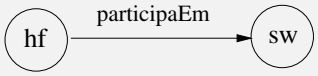
Exemplo: BD de filmes



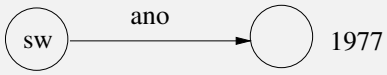
Etiquetas em nó interior

Uma anotação L num arco do nó N para o nó M pode significar 2 coisas:

- N e M podem ser objectos que estão relacionados através de uma associação de nome L.



- N pode ser um objecto e M é um dos atributos do objecto. Neste caso L é o nome do atributo.



XML — eXtensible Markup Language

- XML é um standard que nos permite armazenar e manipular informação semi-estruturada.
- Muito bom para integrar informação oriunda de bases de dados distintas e para partilhar informação na web.

XML

- XML é uma linguagem de anotação.
- Utiliza etiquetas (tags, em inglês) para descrever informação.
- Podemos definir os tags que quisermos.

XML: tags e elementos

- Tags:

```
<ano> </ano>
```

- Um elemento é um par de "matching tags"e tudo o que está aí incluído.
- Um documento XML tem um só elemento raíz, que corresponde à BD inteira.

XML bem formado

- Não há esquema pré-definido.
- Encadeamento de tags tem de ser respeitado, tal como os parêntesis de uma expressão aritmética.

Exemplo

```
<?xml version="1.0" standalone="yes"?>
<Info_Filmes>
   <Actor>
      <Nome>Carrie Fisher</Nome>
      <Morada>
         <Rua>123 Maple St</Rua>
         <Cidade>Hollywood</Cidade>
      </Morada>
      <Morada>
         <Rua>7th Ave</Rua>
         <Cidade>New York</Cidade>
      </Morada>
   </Actor>
```

Exemplo (cont.)

```
  <Actor>
      <Nome>Harrison Ford</Nome>
      <Rua>Oak Rd</Rua>
      <Cidade>Beverly Hills</Cidade>
   </Actor>
   <Filme>
      <Nome>Star Wars</Nome>
      <Ano>1977</Ano>
   </Filme>
</Info_Filmes>
```


Atributos

- Os elementos podem ter atributos. É um modo alternativo de representar uma folha no modelo semi-estruturado.

Exemplo,

```
<Filme ano="1977"><Nome>Star Wars</Nome></Filme>
```

Ou,

```
<Filme nome="Star Wars" ano="1977"></Filme>
```

Ou ainda,

```
<Filme nome="Star Wars" ano="1977" />
```

Atributos para associar elementos

- Podemos usar atributos para representar associações entre elementos.
- Tornam-se uma espécie de apontadores.

```
<?xml version="1.0" standalone="yes"?>
<Info_Filmes>
   <Actor actorID="cf" participaEm="sw">
      <Nome>Carrie Fisher</Nome>
      <Morada>
         <Rua>123 Maple St</Rua>
         <Cidade>Hollywood</Cidade>
      </Morada>
      <Morada>
         <Rua>7th Ave</Rua>
         <Cidade>New York</Cidade>
      </Morada>
   </Actor>
```



```
   <Actor actorID="hf" participaEm="sw">
      <Nome>Harrison Ford</Nome>
      <Rua>Oak Rd</Rua>
      <Cidade>Beverly Hills</Cidade>
   </Actor>
   <Filme filmeID="sw" actores="cf hf">
      <Nome>Star Wars</Nome>
      <Ano>1977</Ano>
   </Filme>
</Info_Filmes>
```

XML válido

- Requer um esquema.
- Pode ser definido usando um DTD (Document Type Definition)
- Um DTD é uma espécie de gramática que define regras para o documento.
- Uma alternativa ao DTD é usar XML Schema (que não vamos ver nesta disciplina.)

DTDs

```
<!DOCTYPE <tag raiz> [
   <!ELEMENT <nome> ( <componentes> )
   <mais elementos>
]>
```

- A descrição de um elemento consiste num nome (tag), e na descrição dos seus elementos constituintes.
  - inclui ordem e multiplicidade.
- As folhas são elementos de texto e têm `#PCDATA` (Parsed Character Data)

DTD para filmes

```
<!DOCTYPE Info_Filmes [
   <!ELEMENT Info_Filmes (Actor*, Filme*)>
   <!ELEMENT Actor (Nome, (Morada+ | (Rua, Cidade)))>
   <!ELEMENT Morada (Rua, Cidade)>
   <!ELEMENT Rua (#PCDATA)>
   <!ELEMENT Cidade (#PCDATA)>
   <!ELEMENT Filme (Nome, Ano)>
   <!ELEMENT Nome (#PCDATA)>
   <!ELEMENT Ano (#PCDATA)>
]>
```


Definição de elementos

- Os elementos devem aparecer pela ordem especificada.
- No final do tag coloca-se um símbolo que indica a multiplicidade.
  - * $\rightarrow$ zero ou mais.
  - $+ \rightarrow$ um ou mais.
  - ? $\rightarrow$ zero ou um.
- O símbolo | tem o significado de "ou" (permite especificar uma sequência alternativa de tags).

Definição de atributos

```
<!ATTLIST nome-elemento nome-atributo tipo >
```

- O tipo mais comum é `CDATA` (uma string)
- Identificador é do tipo `ID`
- Referência é do tipo `IDREF`, lista de referências é do tipo `IDREFS`.
- Atributos podem ser obrigatórios (`#REQUIRED`) ou opcionais (`#IMPLIED`)

DTD para filmes, com ID's e IDREF's

```
<!DOCTYPE Info_Filmes [
   <!ELEMENT Info_Filmes (Actor*, Filme*)>
   <!ELEMENT Actor (Nome, Morada+)>
       <!ATTLIST Actor
         actorID ID #REQUIRED
         participaEm IDREFS #IMPLIED
       >
   <!ELEMENT Nome (#PCDATA)>
   <!ELEMENT Morada (Rua, Cidade)>
   <!ELEMENT Rua (#PCDATA)>
   <!ELEMENT Cidade (#PCDATA)>
   <!ELEMENT Filme (Nome, Ano)>
       <!ATTLIST Filme
         filmeID ID #REQUIRED
         actores IDREFS #IMPLIED
       >
   <!ELEMENT Ano (#PCDATA)>
]>
```

Utilização do DTD

- No documento XML colocar STANDALONE = "no", e depois,
  - Incluir o DTD no preâmbulo do documento XML, ou
  - Colocar DOCTYPE <elemento raiz> SYSTEM "path para o ficheiro que contem o DTD".

Exemplo 1

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE Info_Filmes [
   <!ELEMENT Info_Filmes (Actor*, Filme*)>
   <!ELEMENT Actor (Nome, Morada+)>
   ...
]>
<Info_Filmes>
   <Actor actorID="cf" participaEm="sw">
      <Nome>Carrie Fisher</Nome>
      <Morada>
      ...
      </Morada>
   </Actor>
   ...
</Info_Filmes>
```

Exemplo 2, DTD em ficheiro externo

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE Info_Filmes SYSTEM "filmes-ID.dtd">
<Info_Filmes>
   <Actor actorID="cf" participaEm="sw">
      <Nome>Carrie Fisher</Nome>
      <Morada>
      ...
      </Morada>
   </Actor>
   ...
</Info_Filmes>
```


Como validar

- Existem programas que permitem validar um documento XML
- Um desses programas chama-se `xmllint` e corre na linha de comandos.
- Exemplo de utilização,

```
xmllint --dtdvalid filmesID.dtd --noout filmesID.xml
```

Linguagens de programação para XML

Duas linguagens importantes para manipular XML,

- XPath
- XQuery

XPath

- Permite extrair informação de um documento XML usando expressões que denotam caminhos a partir do raíz do grafo.
- Exemplos,
  - Retorna os actores

    ```
    Info_Filmes/Actor
    ```

  - Retorna os nomes dos actores

    ```
    Info_Filmes/Actor/Nome
    ```

Atributos

- Para extrair o valor de um atributo temos de usar @nome-do-atributo.
- Exemplo,
  - Retorna os IDs dos actores

```
Info_Filmes/Actor/@actorID
```


Condições

- Nome dos actores que moram em Hollywood

```
/Info_Filmes/Actor[Morada/Cidade = "Hollywood"]/Nome
```

- Nome dos filmes feitos depois de 1980

```
/Info_Filmes/Filme[Ano > 1980]/Nome
```

XQuery

- XQuery é uma generalização do XPath
- É uma linguagem funcional baseada em expressões
- Tem algumas semelhanças com SQL

Expressões FLWR (flower)

- F $\rightarrow$ for ...
- L $\rightarrow$ let ...
- W $\rightarrow$ where ...
- R $\rightarrow$ return ...

Expressões FLWR (flower)

Query tem,

- zero ou mais clausulas `for` e `let`, intercaladas em qualquer ordem
- seguido de uma clausula opcional `where`
- seguido de uma clausula `return`

Exemplo

- Nome dos actores

```
let $info := doc("filmes-ID.xml")
for $a in $info//Actor
return $a/Nome
```

- variáveis começam com $
- := → sinal de atribuição
- return não tem o significado habitual como em C ou Java. Pode ser chamado várias vezes. O resultado vai sendo concatenado sempre que a clausula where é verdadeira.

Mais exemplos

- Nome dos actores que moram em Hollywood

```
let $info := doc("filmes-ID.xml")
for $a in $info//Actor
where $a/Morada/Cidade = "Hollywood"
return $a/Nome
```

- Nome de filmes feitos depois de 1980

```
let $info := doc("filmes-ID.xml")
for $f in $info//Filme
where $f/Ano > 1980
return $f/Nome
```


Mais exemplos

- Número de actores

```
let $seq := (
   let $info := doc("filmes-ID.xml")
   for $a in $info//Actor
   return $a/Nome
)
return count($seq)
```

Mais exemplos

- Dentro de tags, substituir variáveis pelo respectivo valor

```
let $seq := (
   let $info := doc("filmes-ID.xml")
   for $a in $info//Actor
   return $a/Nome
)
return <actores>{$seq}</actores>
```


Mais exemplos

- Nome dos filmes em que participou o Harrison Ford

```
let $info := doc("filmes-ID.xml")
let $actores := $info//Actor
let $filmesID_hf := (
   for $a in $actores
   where $a/Nome = "Harrison Ford"
   return $a/data(@participaEm)
)
let $filmes := $info//Filme
for $f in $filmes
where contains($filmesID_hf, $f/@filmeID)
return $f/Nome
```

Mais exemplos

- Retornar pares de actores com a mesma morada (equivalente a um Join)

```
let $info := doc("filmes-ID.xml")
let $actores := $info//Actor
for $a1 in $actores, $a2 in $actores
where $a1/Morada/Rua = $a2/Morada/Rua
  and $a1/Morada/Cidade = $a2/Morada/Cidade
  and $a1/Nome < $a2/Nome
return <par>{$a1/Nome}{$a2/Nome}</par>
```


Informação adicional

- Há muito mais para além disto.
- Um bom local para explorar é ver os tutoriais de XPath e XQuery em `http://www.w3schools.com/`
- Podem testar os exemplos que dei usando um processador de XML, como o Saxon. (Kernow é uma aplicação gráfica que usa Saxon, pode ser decarregada em `http://kernowforsaxon.sourceforge.net/`)
- Existem API's para usar XQuery em programas Java.