XML

"Document Type Definition"
(DTD)

Schemă XML

Document XML “Well-Formed”

- *XML Well-Formed* permite folosirea tagurilor proprii.
- *XML Valid* se conformează unui anumit DTD.

XML “Well-Formed”

- Documentul începe cu o *declaraţie*, delimitată de <?xml … ?> .
- Exemplu de declaraţie:

```
<?xml version = ”1.0”
  standalone = ”yes” ?>
```

  - “standalone” = “fără DTD”
- Un document are un *tag root* ce înconjoară tag-uri imbricate.

Tag-uri

- Tag-urile sunt perechi de etichete:

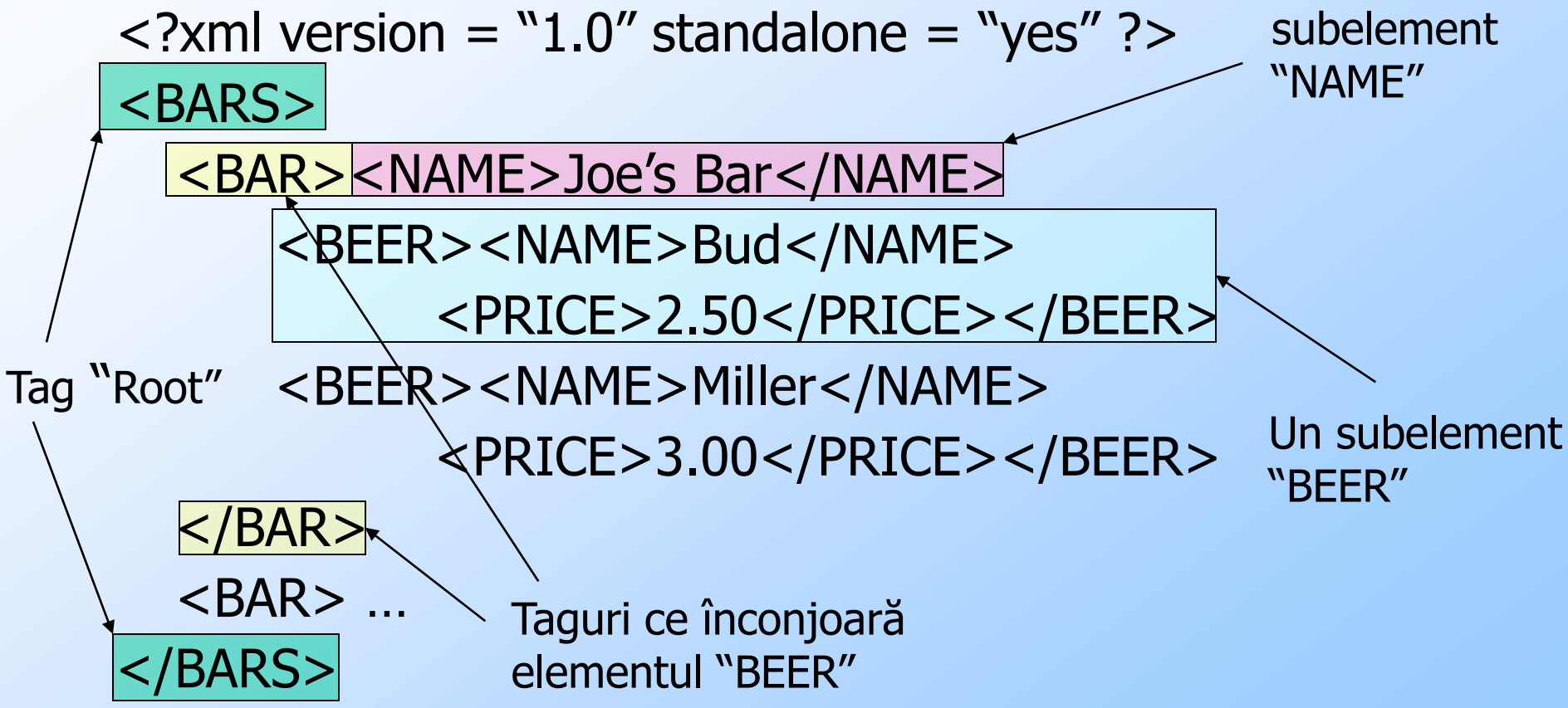
  <FOO> … </FOO>

- Sunt permise tag-uri singulare:

  <FOO/>

- Tag-urile pot fi imbricate în mod arbitrar.
- Tag-urile XML sunt “case-sensitive”.

Exemplu: XML "Well-Formed"



Structura DTD

<!DOCTYPE <tag root> [

  <!ELEMENT <nume> (<componente>)>

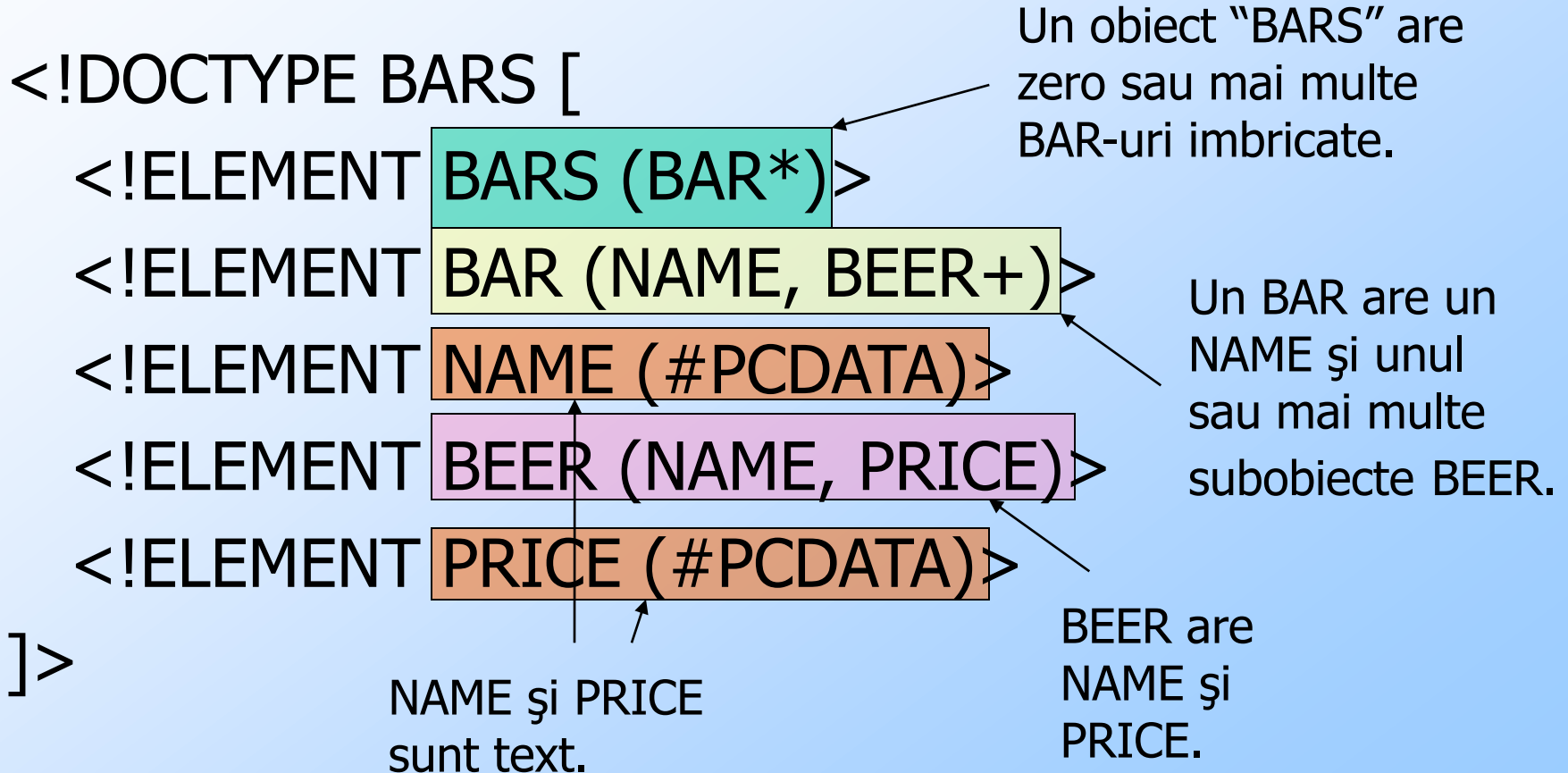
  . . . alte elemente . . .

]>

Elemente DTD

- Descrierea unui element constă din nume (tag) şi o descriere între paranteze a tag-urilor imbricate.
  - Include succesiunea subtag-urilor şi multiplicitatea lor.
- Frunzele (elemente text) au #PCDATA (*Parsed Character DATA*) în loc de tag-uri imbricate.

Exemplu: DTD


<!DOCTYPE BARS [
<!ELEMENT BARS (BAR*)>
<!ELEMENT BAR (NAME, BEER+)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT BEER (NAME, PRICE)>
<!ELEMENT PRICE (#PCDATA)>
]>
Un obiect "BARS" are zero sau mai multe BAR-uri imbricate.
Un BAR are un NAME şi unul sau mai multe subobiecte BEER.
BEER are NAME şi PRICE.
NAME şi PRICE sunt text.

Descrieri Element

- Subtag-urile trebuie să apară în ordinea prezentată.
- Un tag poate fi urmat de un simbol ce indică multiplicitatea.
  - * = zero sau mai multe.
  - + = unu sau mai multe.
  - ? = zero sau unu.
- Simbolul | poate conecta secvenţe alternative de tag-uri.

Exemplu: Descriere Element

- Un nume are un titlu opţional (de exemplu, “Prof.”), un prenume şi numele de familie, în această ordine, sau poate fi o adresă IP:

```
<!ELEMENT NUME (
  (TITLU?, PRENUME, NUME) | ADRIP
)>
```

Folosirea DTD-uri

1. Se specifică standalone = “no”.
2. Şi fie:
   a) Se include DTD ca prefaţă a documentului XML, sau
   b) În continuarea DOCTYPE şi a <tag-ului root> se specifică SYSTEM şi calea (“path”) către fişierul ce conţine DTD.

Exemplu: (a)

```
<?xml version = "1.0" standalone = "no" ?>
<!DOCTYPE BARS [
    <!ELEMENT BARS (BAR*)>
    <!ELEMENT BAR (NAME, BEER+)>
    <!ELEMENT NAME (#PCDATA)>
    <!ELEMENT BEER (NAME, PRICE)>
    <!ELEMENT PRICE (#PCDATA)>
]>
<BARS>
    <BAR><NAME>Joe's Bar</NAME>
        <BEER><NAME>Bud</NAME> <PRICE>2.50</PRICE></BEER>
        <BEER><NAME>Miller</NAME> <PRICE>3.00</PRICE></BEER>
    </BAR>
    <BAR> …
</BARS>
```

DTD

Documentul XML

Exemplu: (b)

- Presupunem că DTD pentru BARS se găseşte în fişierul bar.dtd.

```
<?xml version = "1.0" standalone = "no" ?>
<!DOCTYPE BARS SYSTEM "bar.dtd">
<BARS>
    <BAR><NAME>Joe's Bar</NAME>
        <BEER><NAME>Bud</NAME>
            <PRICE>2.50</PRICE></BEER>
        <BEER><NAME>Miller</NAME>
            <PRICE>3.00</PRICE></BEER>
    </BAR>
    <BAR> …
</BARS>
```

De aici se ştie că DTD se găseşte în fişierul bar.dtd

Atribute

- Tag-urile XML pot avea *atribute*.
- Într-un DTD,

`<!ATTLIST` *E* `...>`

declară atributele elementului *E*, împreună cu tipurile de date.

Exemplu: Atribute

- Bar-urile pot avea un atribut "`kind`", un şir de caractere ce descrie barul.

```
<!ELEMENT BAR (NAME BEER*)>
<!ATTLIST BAR kind CDATA
    #IMPLIED>
```

tipul şir de caractere; fără tag-uri

Atributul este opţional opus: #REQUIRED

Exemplu: Folosire Atribut

- Într-un document ce acceptă tag-uri "BAR", poate exista:

```
<BAR kind = "sushi">
  <NAME>Homma's</NAME>
  <BEER><NAME>Sapporo</NAME>
        <PRICE>5.00</PRICE></BEER>
  ...
</BAR>
```

ID-uri şi IDREF-uri

- Atributele pot fi pointeri de la un obiect la altul.
  - Comparabil cu HTML: NAME = ”foo” şi HREF = ”#foo”.
- Permite structurii unui document XML să fie un graf, în loc de arbore.

Crearea ID-urilor

- Presupunem un element *E* cu un atribut *A* de tip ID.
- Atunci când se foloseşte tag-ul <*E*> într-un document XML, atributul *A* primeşte o valoare unică.
- Exemplu:

```
<E  A = ”xyz”>
```

Crearea IDREF-urilor

- Pentru a permite elementelor de tip *F* să facă referire la un alt element cu un atribut ID, *F* primeşte un atribut de tip IDREF.
- Sau, atributul are tipul IDREFS, astfel încât elementul *F* poate face referire la oricâte alte elemente.

Exemplu: ID-uri şi IDREF-uri

- Un DTD nou pentru BARS include atât subelemente BAR cât şi BEER.
- BARS şi BEERS au atribute ID `name`.
- BARS au subelemente SELLS, ce constau din un număr (preţul unei beri) şi IDREF-ul `theBeer` ce conduce la acea marcă de bere.
- BEERS au atributul `soldBy`, ce este un IDREFS ce conduce la toate barurile ce vând berea respectivă.

DTD

Elementele BAR au numele ca un atribut ID şi au unul sau mai multe subelemente SELLS.

Elementele SELLS au un număr (preţul) şi o referinţă la beer.

```
<!DOCTYPE BARS [
   <!ELEMENT BARS (BAR*, BEER*)>
   <!ELEMENT BAR (SELLS+)>
      <!ATTLIST BAR name ID #REQUIRED>
   <!ELEMENT SELLS (#PCDATA)>
      <!ATTLIST SELLS theBeer IDREF #REQUIRED>
   <!ELEMENT BEER EMPTY>
      <!ATTLIST BEER name ID #REQUIRED>
      <!ATTLIST BEER soldBy IDREFS #IMPLIED>
]>
```

Se va explica în continuare

Elementele BEER au un atribut ID denumit name, şi un atribut soldBy ce este un set de nume de BAR-uri.

Exemplu: Document

```
<BARS>
   <BAR name = ”JoesBar”>
        <SELLS theBeer = ”Bud”>2.50</SELLS>
        <SELLS theBeer = ”Miller”>3.00</SELLS>
   </BAR> …
   <BEER name = ”Bud” soldBy = ”JoesBar
        SuesBar …” /> …
   </BARS>
```

Elemente “Empty”

- Un element poate fi descris prin atribute.
  - În exemplul anterior BEER.
- Un alt exemplu: elementele SELLS ar fi putut avea atributul `price` în locul valorii ce reprezintă “price”.

Exemplu: Element “Empty”

- În DTD se declară:

```
<!ELEMENT SELLS EMPTY>
   <!ATTLIST SELLS theBeer IDREF #REQUIRED>
   <!ATTLIST SELLS price CDATA #REQUIRED>
```

- Exemplu de utilizare:

```
<SELLS theBeer = ”Bud” price = ”2.50” />
```

De notat excepţia la regula “tag-uri pereche”

Schemă XML

- Oferă o cale mai puternică pentru a descrie structura documentelor XML.
- Declaraţiile Schemă-XML sunt ele însele documente XML .
  - Ele descriu “elemente” iar descrierea se face de asemenea cu “elemente”.

Structura unui Document Schemă-XML

```
<? xml version = … ?>
<xs:schema xmlns:xs =
  ”http://www.w3.org/2001/XMLschema”>
                    . . .
</xs:schema>
```

Folosirea ”xs” la definirea elementelor schemei se referă la tag-uri din acest namespace.

Se defineşte ”xs” ca fiind *namespace* şi este descris prin URL-ul respectiv. Orice şir de caractere poate lua locul ”xs”.

Elementul xs:element

- Are atributele:
  - name = numele tag-ului elementului definit.
  - type = tipul elementului.
    - Poate fi un tip Schemă-XML , de exemplu, xs:string.
    - Sau numele tipului definit în document.

Exemplu: xs:element

```
<xs:element name = ”NUME”
     type = ”xs:string” />
```

- Descrie elemente ca de exemplu:

  <NUME>Pop Ion</NUME>

Tipuri Complexe

- Pentru a descrie elemente ce constau din subelemente, se foloseşte xs:complexType.
  - Atributul name precizează numele tipului.
- Un subelement tipic al unui tip complex este xs:sequence, ce conţine o secvenţă xs:element de subelemente.
  - Se folosesc atributele minOccurs şi maxOccurs pentru a controla numărul apariţiilor unui xs:element.

Exemplu: Tip pentru Beers

```
<xs:complexType name = "beerType">
  <xs:sequence>
    <xs:element name = "NAME"
      type = "xs:string"
      minOccurs = "1" maxOccurs = "1" />
    <xs:element name = "PRICE"
      type = "xs:float"
      minOccurs = "0" maxOccurs = "1" />
  </xs:sequence>
</xs:complexType>
```

Exact 1 apariţie

Asemănător cu ? la DTD

Un Element de Tip “beerType”

```
<xxx>
  <NAME>Bud</NAME>
  <PRICE>2.50</PRICE>
</xxx>
```

Deoarece nu se ştie numele elementului de tipul respectiv.

Exemplu: un Tip pentru Bars

```
<xs:complexType name = "barType">
  <xs:sequence>
    <xs:element name = "NAME"
      type = "xs:string"
      minOccurs = "1" maxOccurs = "1" />
    <xs:element name = "BEER"
      type = "beerType"
      minOccurs = "0" maxOccurs =
          "unbounded" />
  </xs:sequence>
</xs:complexType>
```

Asemănător cu * la DTD

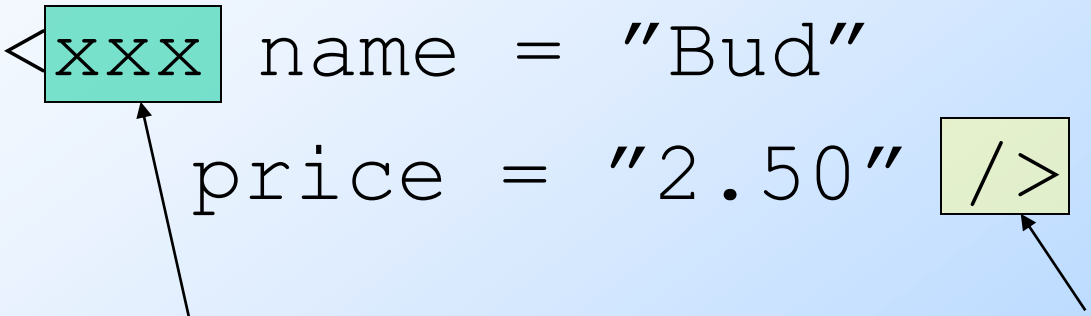
xs:attribute

- Elementele xs:attribute pot fi folosite în cadrul unui tip complex pentru a indica atributele elementelor tipului respectiv.
- Atribute xs:attribute:
  - name şi type ca şi la xs:element.
  - use = ”required” sau ”optional”.

Exemplu: xs:attribute

```
<xs:complexType name = ”beerType”>
  <xs:attribute name = ”name”
     type = ”xs:string”
     use = ”required” />
  <xs:attribute name = ”price”
     type = ”xs:float”
     use = ”optional” />
</xs:complexType>
```

Un Element de Tipul beerType



Tipuri Simple Restricţionate

- xs:simpleType poate descrie enumerări şi tipuri de bază 'plaje de valori'.
- name este un atribut
- xs:restriction este un subelement.

Restricţii

- Atributul base specifică tipul simplu ce va fi restricţionat, de exemplu, xs:integer.
- xs:{min, max}{Inclusive, Exclusive} sunt patru atribute ce specifică limita inferioară şi superioară a unei plaje de valori numerice.
- xs:enumeration este un subelement cu atributul value ce permite tipuri enumerare.

Exemplu: Attributul license pentru BAR

```
<xs:simpleType name = ”license”>
  <xs:restriction base = ”xs:string”>
    <xs:enumeration value = ”Full” />
    <xs:enumeration value = ”Beer only” />
    <xs:enumeration value = ”Sushi” />
  </xs:restriction>
</xs:simpleType>
```

Exemplu: Preţuri în Plaja [1,5)

```
<xs:simpleType name = ”price”>
  <xs:restriction
     base = ”xs:float”
     minInclusive = ”1.00”
     maxExclusive = ”5.00” />
</xs:simpleType>
```

Chei în Schema XML

- Un xs:element poate avea un subelement xs:key.
- Semnificaţia: pentru acest element, toate subelementele la care se ajunge cu un anumit *selector* de cale ("path") vor avea valori unice pentru o anumită combinaţie de *câmpuri*.
- Exemplu: pentru un element BAR, atributul name al elementului BEER este unic.

Exemplu: Cheie

```
<xs:element name = ”BAR” … >
    . . .
  <xs:key name = ”barKey”>
     <xs:selector xpath = ”BEER” />
     <xs:field xpath = ”@name” />
  </xs:key>
    . . .
</xs:element>
```

@ indică un atribut şi nu un tag.

XPath este un limbaj de interogare pentru XML. Singurul lucru ce este nevoie să fie cunoscut pentru moment este că o cale (“path”) este o secvenţă de tag-uri separate de /.

Chei Străine

- Un subelement xs:keyref pentru un xs:element precizează că pentru acest element, anumite valori (definite printr-un selector şi câmp(uri), ca şi la chei) trebuie să apară ca şi valori ale unei chei.

Exemplu: Cheie Străină

- Presupunem că s-a declarat că subelementul NAME al BAR este o cheie pentru BARS.
  - Numele cheii este barKey.
- Se doreşte să se declare că elementele DRINKER au subelementele FREQ. Un atribut bar al FREQ este cheie străină, ce referă către NAME al BAR.

Exemplu: Cheie Străină în Schema XML

```
<xs:element name = ”DRINKERS”
      . . .
  <xs:keyref name = ”barRef”
     refers = ”barKey”
     <xs:selector xpath =
          ”DRINKER/FREQ” />
     <xs:field xpath = ”@bar” />
  </xs:keyref>
</xs:element>
```