

# **XML PT.2**

Tecnologie Web e Laboratorio A.A. 2020-21 - Università degli Studi di Udine



Andrea Urgolo - [andrea.urgolo@uniud.it](mailto:andrea.urgolo@uniud.it)

# LEZIONE PRECEDENTE

- Introduzione
- Elementi
- Attributi
- Entità
- Altre Componenti
- Ben Formatezza
- Document Type Definition

# INDICE

- Document Type Definition (cont.)
  - Dichiarazione entità
  - Dichiarazione entità parametriche
  - Esempi
- Validazione
- ID, IDREF(S)
- XPath
  - Percorso di locazione
  - Operatori e Funzioni
  - Esempi

# EDITOR

Usate l'editor che preferite:

- **Brackets** <http://brackets.io>
- **Visual Studio Code**  
<https://code.visualstudio.com>
- **Atom** <https://atom.io>

Editor online:

- <https://jsbin.com>
- <https://jsfiddle.net>
- <https://codesandbox.io>
- <https://codeanywhere.com>
- ecc.

# MATERIALE

## Documentazione ufficiale

- Raccomandazione del W3C:  
Extensible Markup Language (XML) 1.0  
(Fifth Edition)  
<https://www.w3.org/TR/xml>

## Siti con materiale didattico:

- <https://w3schools.com/xml>
- <http://users.dimi.uniud.it/~massimo.franceschet/caffe-xml>

## Libro di testo suggerito:



XML in a Nutshell, 3rd Edition by  
Elliotte Rusty Harold, W. Scott  
Means, Pub.: O'Reilly Media, Inc.  
ISBN: 9780596007645

# DOCUMENT TYPE DEFINITION (CONT.)

XML – Tecnologie Web  
e Laboratorio

# ESERCIZIO: SCRIVERE IL DTD

```
<playlist>
  <track id="t01" genere="rock">
    <title>Time</title>
    <artist>Pink Floyd</artist>
    <album>The Dark Side of the Moon</album>
    <file src="01-time.mp3">
      <encoding>MP3</encoding><bitrate>160kbps</bitrate>
    </file>
  </track>
  ...
</playlist>
```

# ESERCIZIO: POSSIBILE SOLUZIONE

```
<!ELEMENT playlist (track+)>
<!ELEMENT track (title, artist+, album?, file)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT artist (#PCDATA)>
<!ELEMENT album (#PCDATA)>
<!ELEMENT file (encoding, bitrate))>
<!ELEMENT encoding (#PCDATA)>
<!ELEMENT bitrate (#PCDATA)>
<!ATTLIST track id ID #IMPLIED
               genre ( rock | pop | jazz | blues | other) 'other'>
<!ATTLIST file src CDATA #REQUIRED>
```



# DICHIARAZIONE DI ENTITÀ

**`<!ENTITY nome_entità contenuto_entità>`**

Es.:

```
<!ENTITY XML "Extensible Markup Language">
<!ENTITY footer "<author>Andrea Urgolo</author>
<date>17 November 2020</date>">
```

È possibile definire un'entità con contenuto esterno

**`<!ENTITY nome_entità SYSTEM "URL">`**

Es.:

```
<!ENTITY footer SYSTEM "footer.xml">
```

# DICHIARAZIONE DI ENTITÀ PARAMETRICHE

L'entità **parametrica** rappresenta un riferimento ad una porzione di DTD

**<!ENTITY % *nome\_entità contenuto\_entità*>**

Es.:

```
<!ENTITY % contenuto "titolo, (paragrafo | immagine) *">  
<!ELEMENT capitolo (%contenuto;, sezione*)>  
<!ELEMENT sezione (%contenuto;)>
```

Per favorire la modularizzazione e il riuso del DTD, è possibile definire un'entità parametrica con contenuto esterno

**<!ENTITY % *nome\_entità* SYSTEM "*URL*">**

Es.:

```
<!ENTITY % body SYSTEM "body.dtd">  
  
%body;
```

# ESERCIZIO

Scrivere il DTD per un documento XML che rappresenti un generico libro di testo

# ESERCIZIO (POSSIBILE SOLUZIONE)

```
<!ELEMENT libro      (autore+,titolo,capitolo+)>
<!ENTITY % contenuto "titolo, (paragrafo|figura)*">
<!ELEMENT capitolo   (%contenuto;,sezione*)>
<!ELEMENT sezione  (%contenuto;)>
<!ELEMENT autore     (#PCDATA)>
<!ELEMENT titolo     (#PCDATA)>
<!ELEMENT paragrafo  (#PCDATA|riferimento)*>
<!ELEMENT figura     (#PCDATA)>
<!ELEMENT riferimento EMPTY>
<!ATTLIST libro isbn ID #IMPLIED>
<!ATTLIST riferimento risorsa IDREF #REQUIRED>
```

# VALIDAZIONE

XML – Tecnologie Web  
e Laboratorio

# COLLEGARE UN DOCUMENTO XML AL DTD

Per collegare un documento XML a un DTD, inserire nel documento XML (sulla riga successiva all'eventuale dichiarazione XML) una **document type declaration**:

```
<!DOCTYPE el_radice DTD>
```

dove **DTD** può essere

1. **interno**: incluso nel documento stesso tra [ e ]
2. **esterno**: un riferimento alla risorsa in cui il DTD è contenuto: **SYSTEM dtd\_URL**
3. **esterno pubblico**: un riferimento **PUBLIC dtd\_URN** (ad es. "-//Uniuud//Address Book/EN"), seguito da un riferimento alla risorsa (**dtd\_URL**)
4. sia **interno**, che **esterno**: parte del DTD tra [], e parte in un file esterno (indicato tramite **SYSTEM** o **PUBLIC**)

Es.: 

```
<!DOCTYPE el_radice SYSTEM "URL" [DTD]>
```

# ESEMPIO DTD INTERNO

```
<?xml version="1.0"?>
<!DOCTYPE address-book [
  <!ELEMENT address-book (entry+)>
  <!ELEMENT entry (name,address?,tel+,email*)>
  <!ELEMENT name (#PCDATA)>

  ...
]>

<address-book>
  <entry>
    ...

    <email address="jdoe@email.com"/>
  </entry>
</address-book>
```

# ESEMPIO DTD ESTERNO

```
<?xml version="1.0"?>
<!DOCTYPE address-book SYSTEM "address-book.dtd">
<address-book>
  <entry>
    <name><fname>John</fname><lname>Doe</lname></name>
    <address>
      <street>34 Fountain Square Plaza</street>
      <region>OH</region><postal-code>45202</postal-code>
      <locality>Cincinnati</locality><country>US</country>
    </address>
    <tel preferred="true">513-555-8889</tel>
    <tel>513-555-7098</tel>
    <email address="jdoe@email.com" />
  </entry>
</address-book>
```



# ESEMPIO DTD ESTERNO PUBBLICO (XHTML)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html version="-//W3C//DTD XHTML 1.1//EN"
    xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/1999/xhtml
        http://www.w3.org/Markup/SCHEMA/xhtml11.xsd">
    <head>
        <title>Virtual Library</title>
    </head>
    <body>
        <p>
            Moved to <a href="http://example.org/">example.org</a>.
        </p>
    </body>
</html>
```

# DOCUMENTO XML VALIDO

- Include una **document type declaration** che identifica il DTD da soddisfare
  - Rispetta quanto viene dichiarato nel DTD
  - Principio di validità: **tutto ciò che non è permesso è vietato**
    - il DTD elenca tutti gli elementi, attributi ed entità che possono essere usati nel documento e loro contesto d'impiego
    - non si possono usare elementi, attributi ed entità non dichiarati nel DTD
  - Validatori:
    - <https://www.xmlvalidation.com>
    - [https://www.truugo.com/xml\\_validator](https://www.truugo.com/xml_validator)
    - xmllint: <http://xmlsoft.org/downloads.html>
      - parser XML chiamabile da riga di comando
- es.: `xmllint --valid --noout documento.xml`

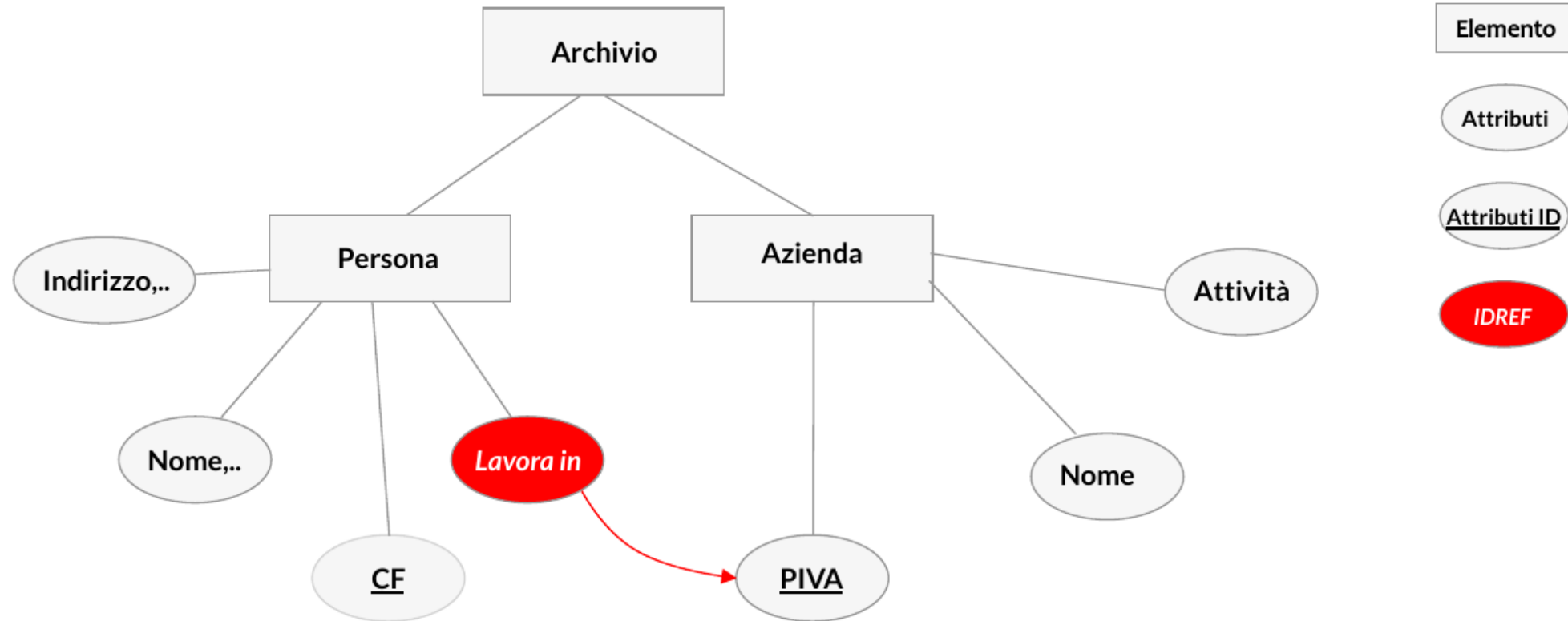
# ID, IDREF(S)

XML – Tecnologie Web  
e Laboratorio

# UTILIZZO IDREF → ID

- Gli attributi di tipo **IDREF** servono per collegare un elemento del documento XML ad un altro elemento. Il contenuto di un **IDREF** è il valore di un attributo **ID** di qualche altro elemento.
- Ad esempio: desideriamo strutturare le informazioni per costruire un archivio in cui vi siano un **elenco di persone** ed un **elenco di aziende**. Vogliamo inoltre sapere **in quale azienda lavora ciascuna persona** aggiungendo un **riferimento** tra la persona e l'azienda.

# ESEMPIO IDREF → ID



# ESEMPIO IDREF → ID: CODICE

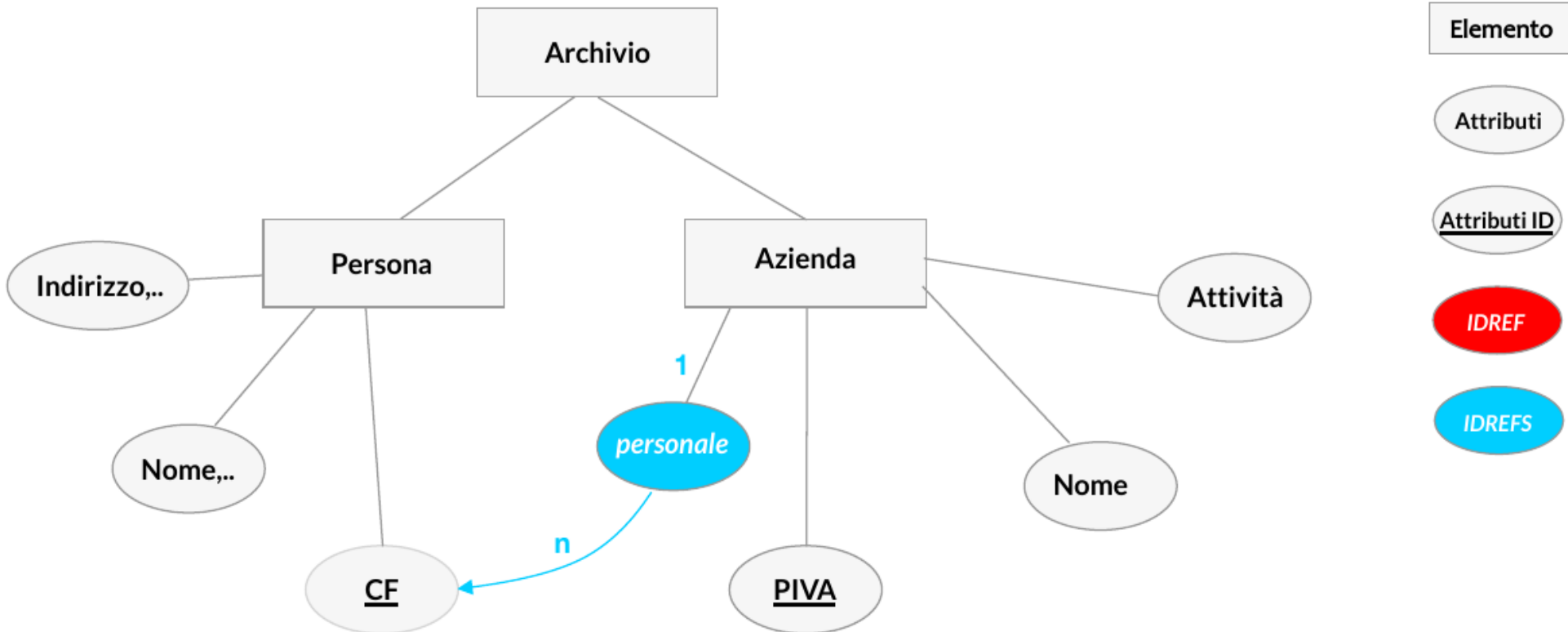
```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE archivio [
  <!ELEMENT archivio (azienda*,persona*)>
  <!ELEMENT azienda (#PCDATA)>
  <!ELEMENT persona (#PCDATA)>
  <!--ATTLIST azienda piva ID #REQUIRED-->
  <!--ATTLIST persona cf ID #REQUIRED-->
  <!--ATTLIST persona lavora_in IDREF #REQUIRED-->
]

<archivio>
  <azienda piva=" 02363620309">ACME S.R.L.</azienda>
  <azienda piva="74646478484">WebWebWeb</azienda>
  <persona cf="cfmr" lavora_in=" 02363620309">Mario Rossi</persona>
  <persona cf="cfef" lavora_in="74646478484">Elisa Bianchi</persona>
  <persona cf="cfpp" lavora_in="74646478484">Pinco Pallo</persona>
</archivio>
```

# ESEMPIO IDREFS → ID(S)

Per ciascuna azienda consideriamo un attributo “**personale**” di tipo **IDREFS** che ci serve come **riferimento a tutti i dipendenti** che lavorano in quella azienda.



# ESEMPIO IDREFS → ID(S): CODICE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE archivio [
  <!ELEMENT archivio (azienda*,persona*)>
  <!ELEMENT azienda (#PCDATA)>
  <!ELEMENT persona (#PCDATA)>
  <!ATTLIST azienda piva ID #REQUIRED>
  <!!ATTLIST persona cf ID #REQUIRED>
  <!!ATTLIST azienda personale IDREFS #REQUIRED>
]>
<archivio>
  <azienda piva=" 02363620309" personale="cfmr cfeb">ACME
  S.R.L.</azienda>
  <azienda piva=" 74646478484" personale="cfpp">WebWebWeb</azienda>
  <persona cf="cfmr">Mario Rossi</persona>
  <persona cf="cfeb">Elisa Bianchi</persona>
  <persona cf="cfpp">Pinco Pallo</persona>
</archivio>
```



# ESERCIZIO

Si pensi ad un sito relativo ad una community di appassionati di musica. Si strutturi un archivio dati che contenga dei **brani musicali** e un elenco di **utenti**. Per ciascun utente si vuole tenere traccia dei **riferimenti** ai brani preferiti.

# ESERCIZIO: POSSIBILE SOLUZIONE

```
<?xml version="1.0" ?>
<!DOCTYPE music-site [
  <!ELEMENT music-site (track*, user*)>
  <!ELEMENT user (name,surname)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT surname (#PCDATA)>
  <!ELEMENT track (title, artist+, album?)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT artist (#PCDATA)>
  <!ELEMENT album (#PCDATA)>
  <!ATTLIST track id ID #REQUIRED>
  <!ATTLIST user favorite-tracks IDREFS #REQUIRED>
]>
<music-site>
  <track id="t1"><title>Track 1</title><artist>Artist 1</artist></track>
  <track id="t2"><title>Track 2</title><artist>Artist 2</artist></track>
  <track id="t3"><title>Track 3</title><artist>Artist 3</artist></track>
  <track id="t4"><title>Track 4</title><artist>Artist 4</artist></track>
  <track id="t5"><title>Track 5</title><artist>Artist 5</artist></track>
  <user favorite-tracks="t2 t5"><name>User</name><surname>One</surname></user>
  <user favorite-tracks="t1 t3
t5"><name>User</name><surname>Two</surname></user>
</music-site>
```

# XPATH

XML – Tecnologie Web  
e Laboratorio

# XPATH

- **XPath** è un linguaggio di interrogazione che consente di selezionare dei nodi o computare dei valori dal contenuto di un documento XML
- La prima versione è stata pubblicata dal World Wide Web Consortium (W3C) nel 1999
- Sfruttato da varie tecnologie XML, linguaggi, API, librerie e applicazioni
  - **XSLT**: contiene la parte XPath e ne aggiunge un'altra per effettuare delle trasformazioni dei dati selezionati (es. XML -> HTML, XML -> PDF)
  - **XQuery**: è il linguaggio più espressivo, si basa su XPath ma aggiunge un vero e proprio linguaggio per effettuare interrogazioni in stile SQL
  - **XML Schema**: simile a DTD, ma più complesso; viene utilizzato per esprimere un insieme (schema) di regole alle quali un documento XML deve conformarsi per essere considerato "valido" in accordo con tale schema
  - **XPointer**: per puntare ad degli elementi XML particolari presenti nel documento XML linkato
  - ...

# COME FUNZIONA XPATH

XPath ci permette di fare due cose:

1. Definire un percorso nell'albero (**location path**), ossia come navigarlo

**/step1/step2/step3/....**

2. Definire dei criteri (**conditions**) per decidere quali dati prendere

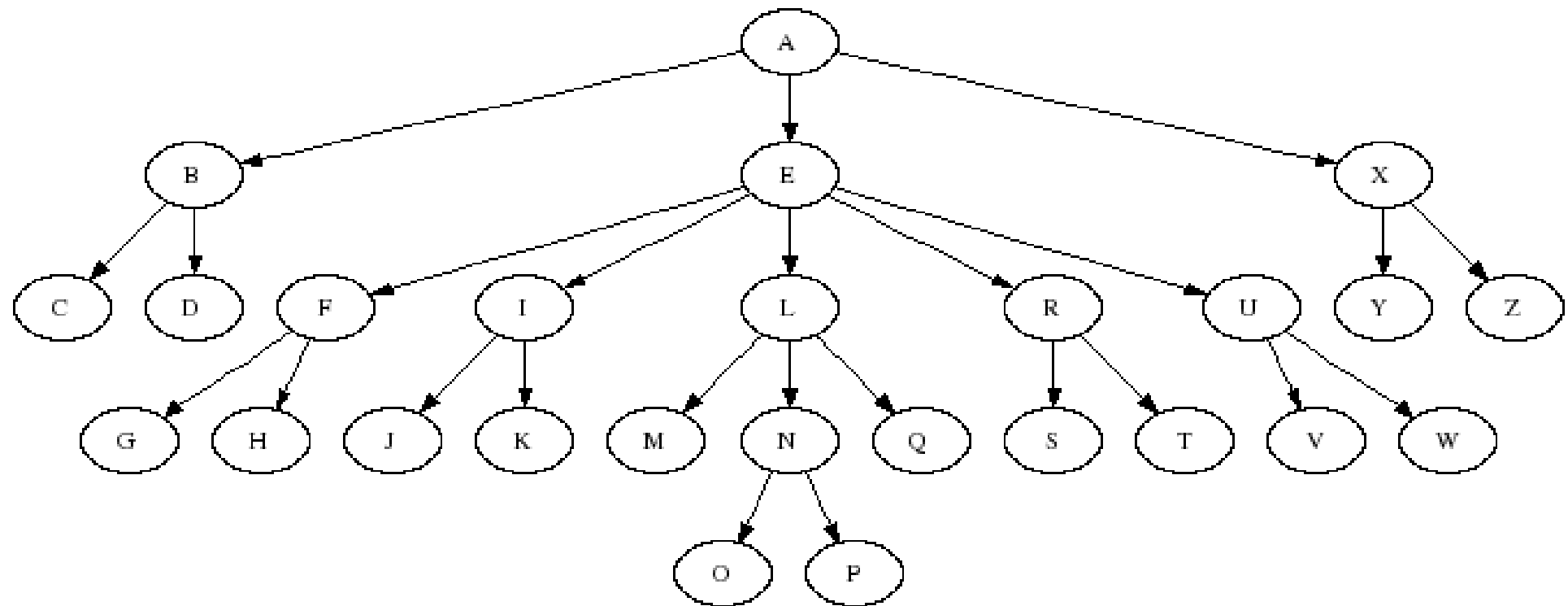
**step := asse::test[conditions]**

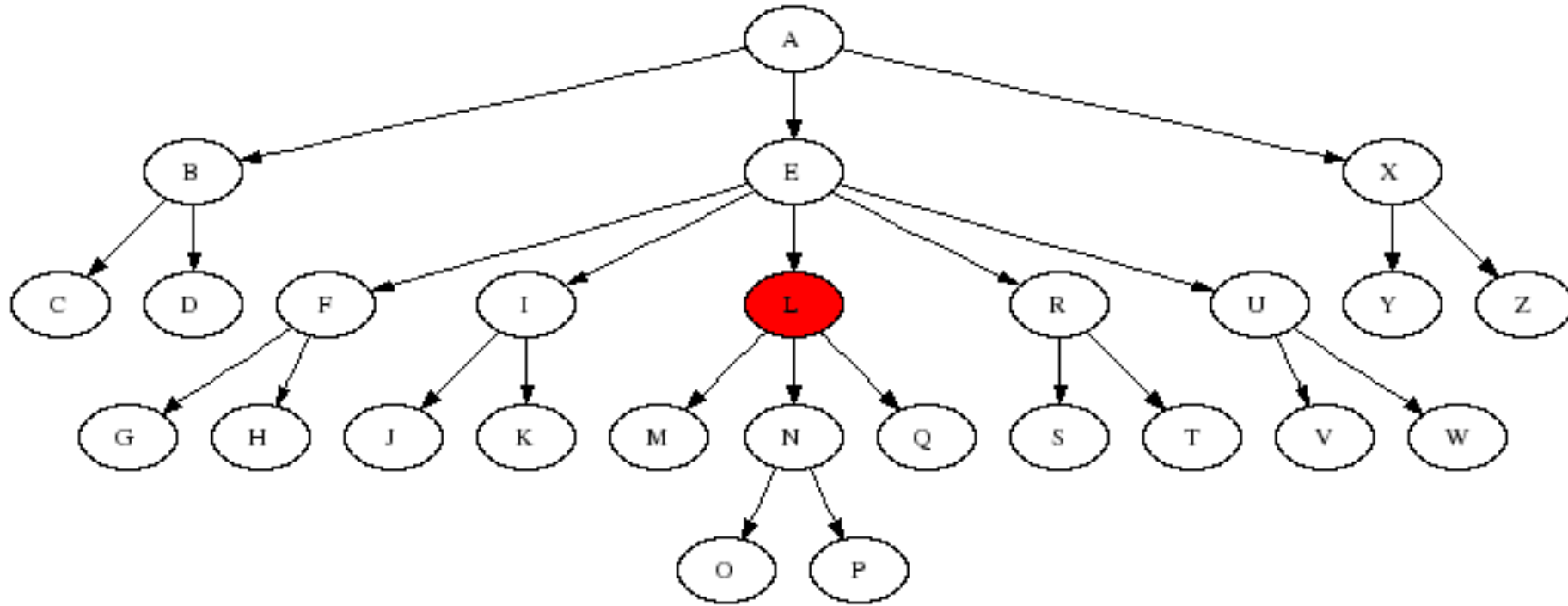
Esempio di condizione: `@price < 50`

# PERCORSI DI LOCALIZZAZIONE

- Ogni espressione XPath (o query XPath) viene valutata in uno più nodi dell'albero del documento XML (chiamati **nodi di contesto**) e restituisce un oggetto di uno dei quattro tipi: booleano, numero, stringa, node set (insieme di nodi).
- L'espressione XPath più importante è chiamata **percorso di localizzazione** (location path) e restituisce sempre un node set. Un percorso di localizzazione può essere composto da più **step**. Ogni passaggio ha la forma **asse::test[conditions]**. La parte **[conditions]** è facoltativa.

# MODELLO DEI DATI

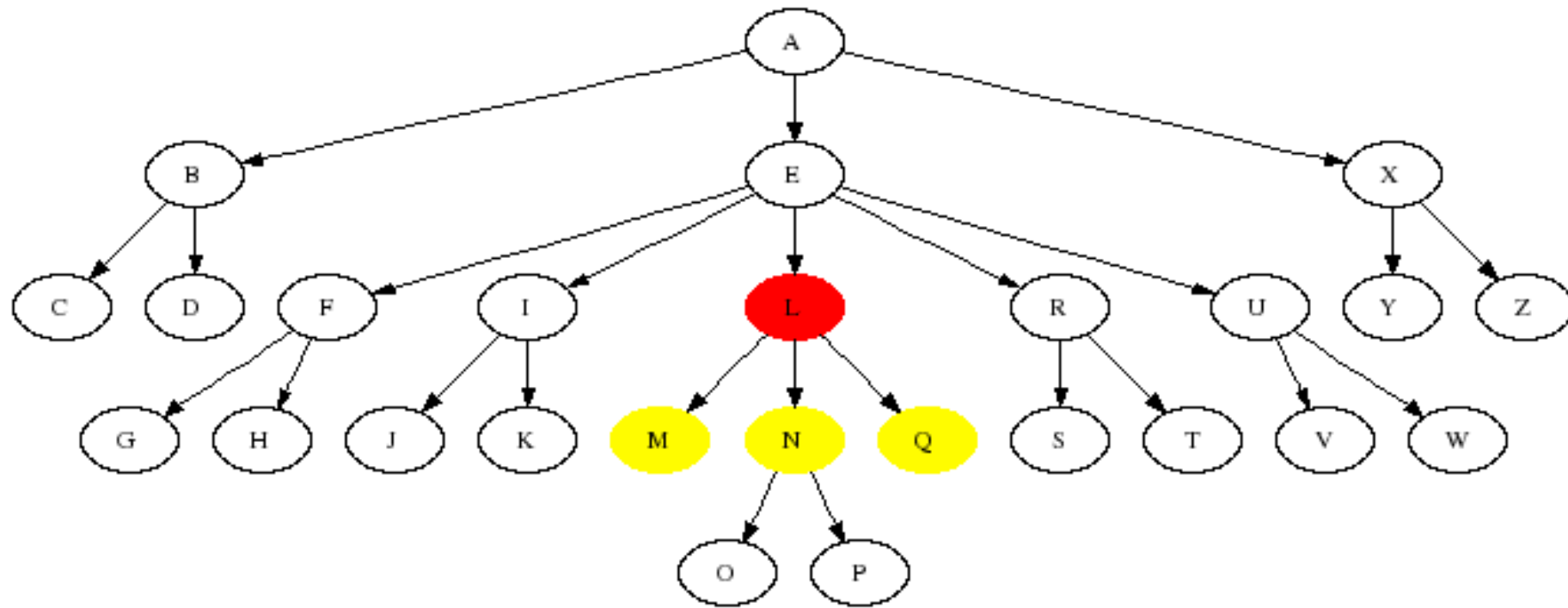




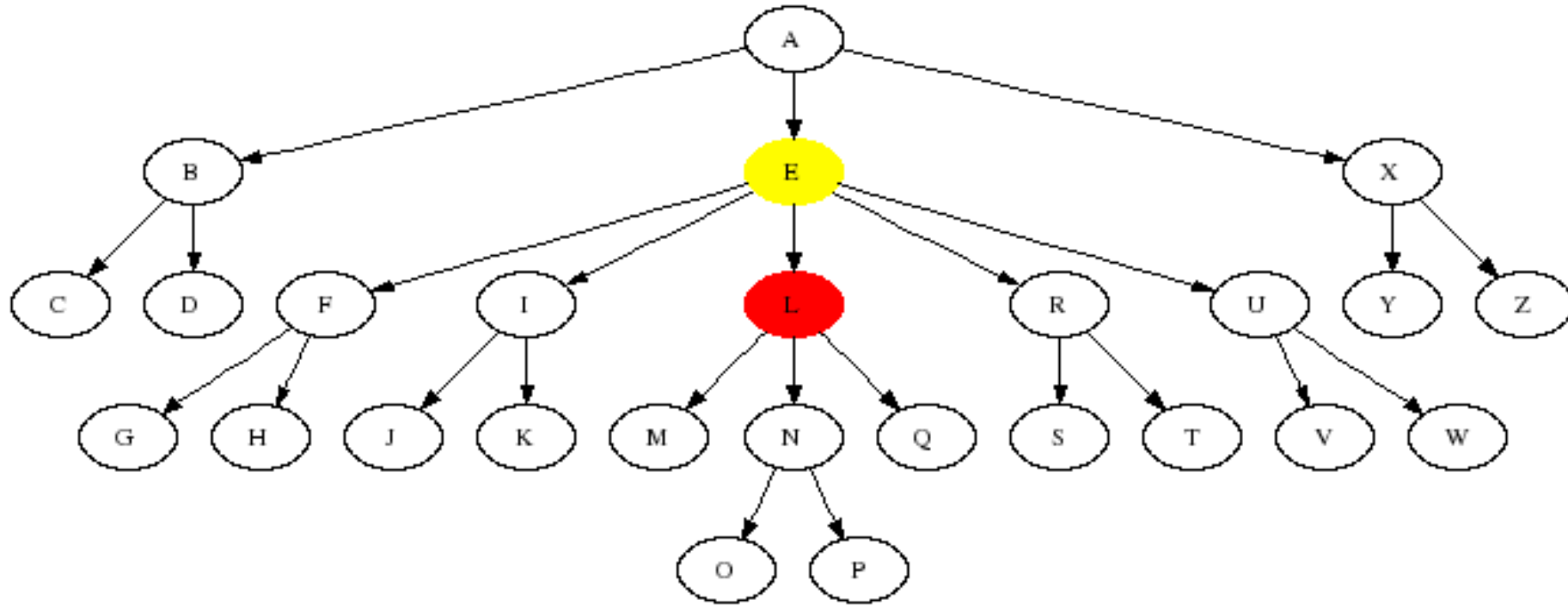
**ASSE self:\***

self = nodo di contesto (rosso)

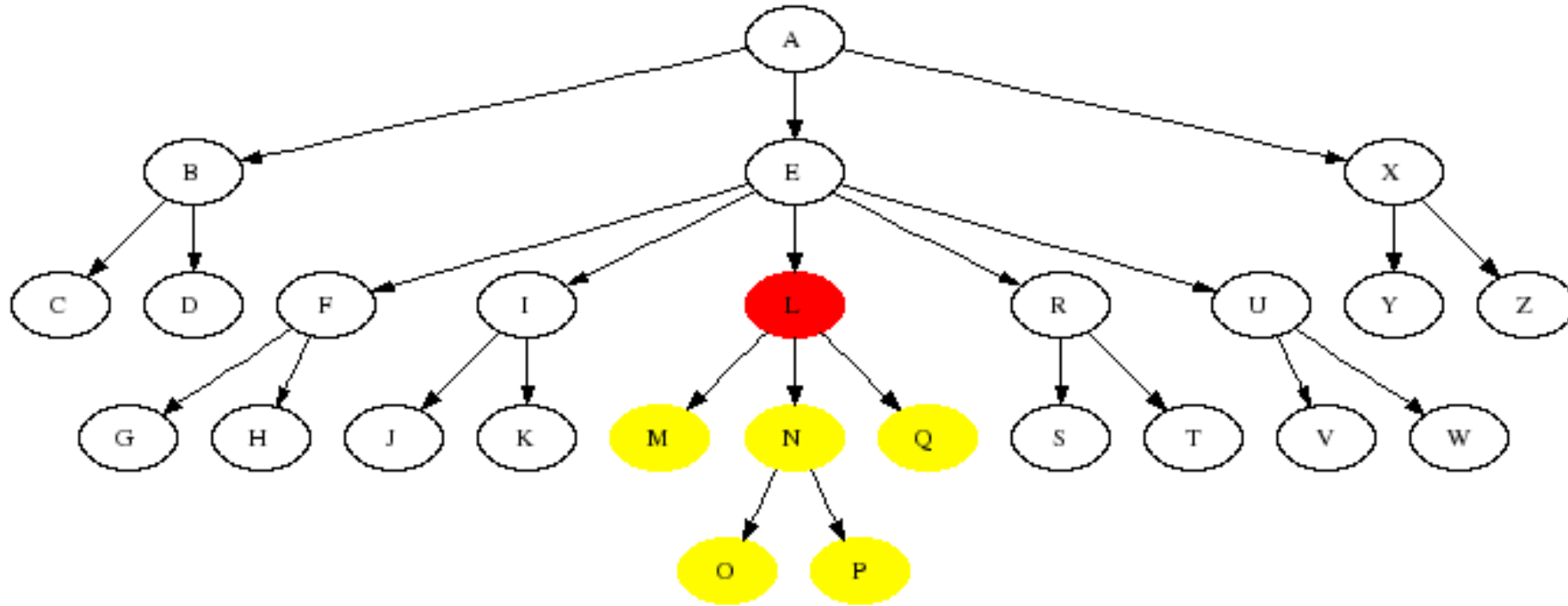




**ASSE child::\***

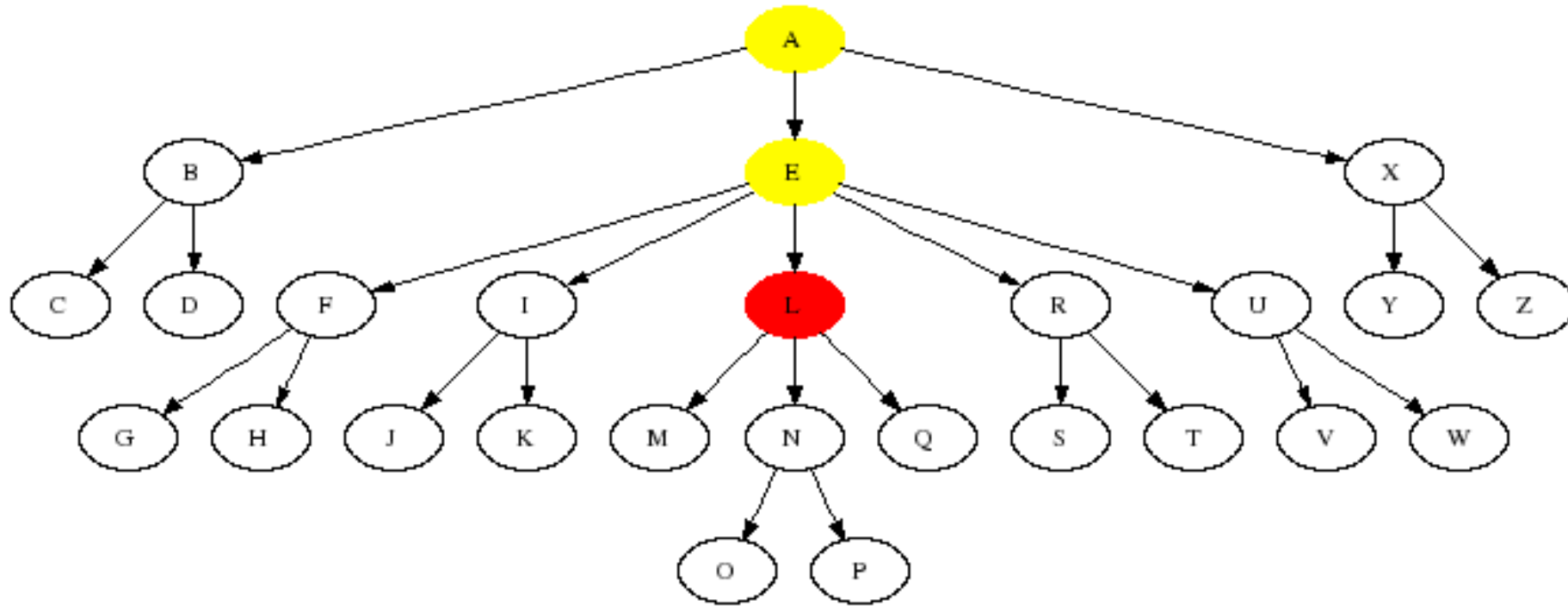


**ASSE parent::\***



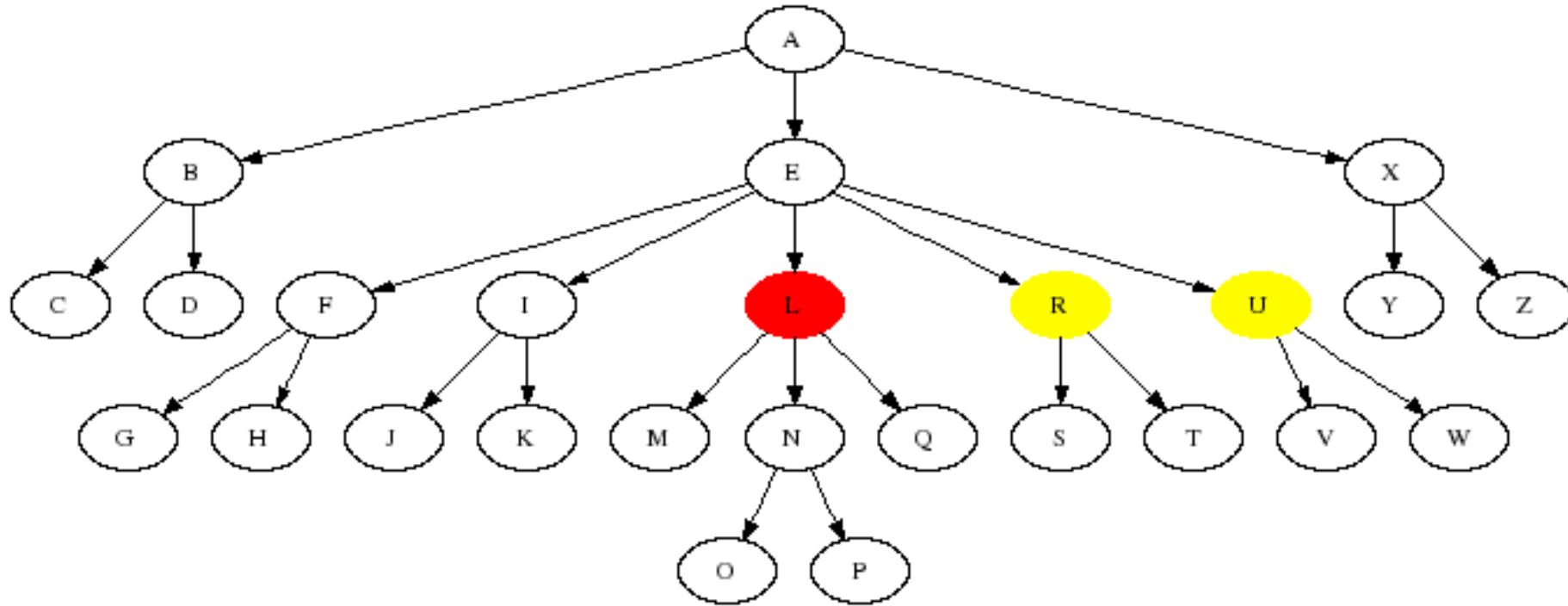
# ASSE descendant::\* e descendant-or-self::\*

self = nodo di contesto (rosso)

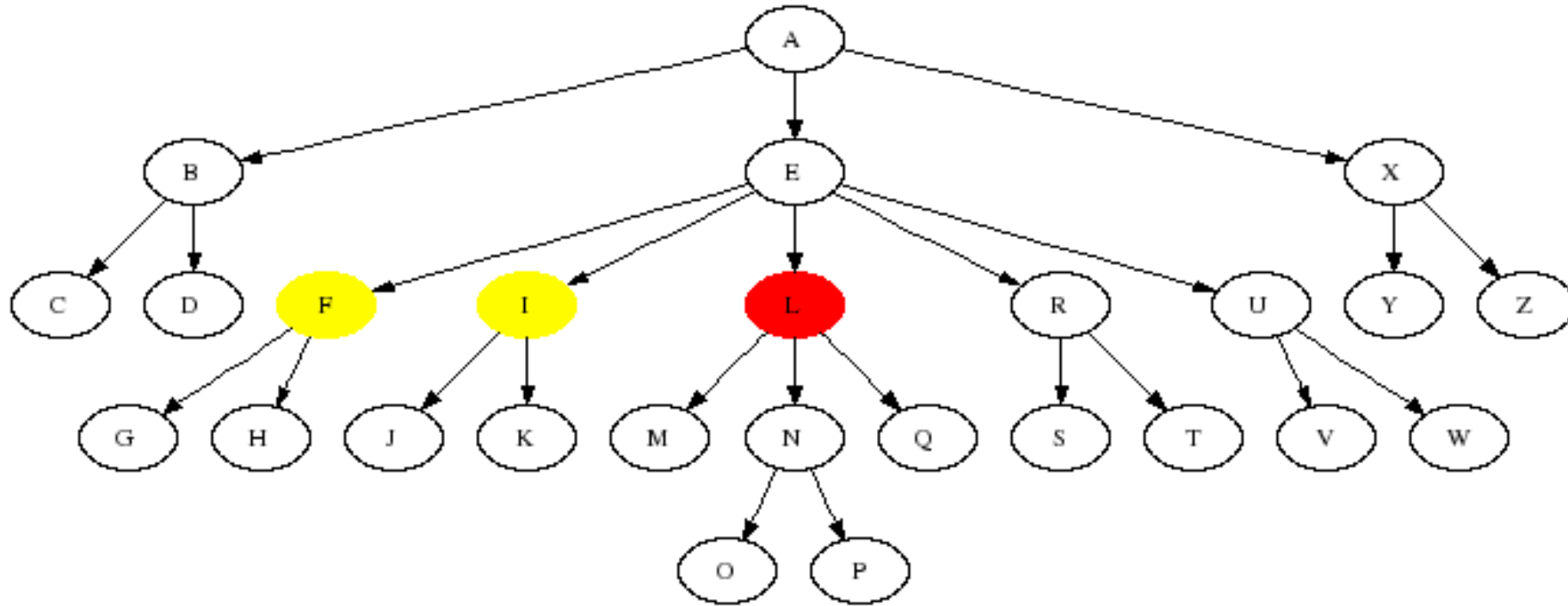


# ASSE ancestor::<sup>\*</sup> e ancestor-or-self::<sup>\*</sup>

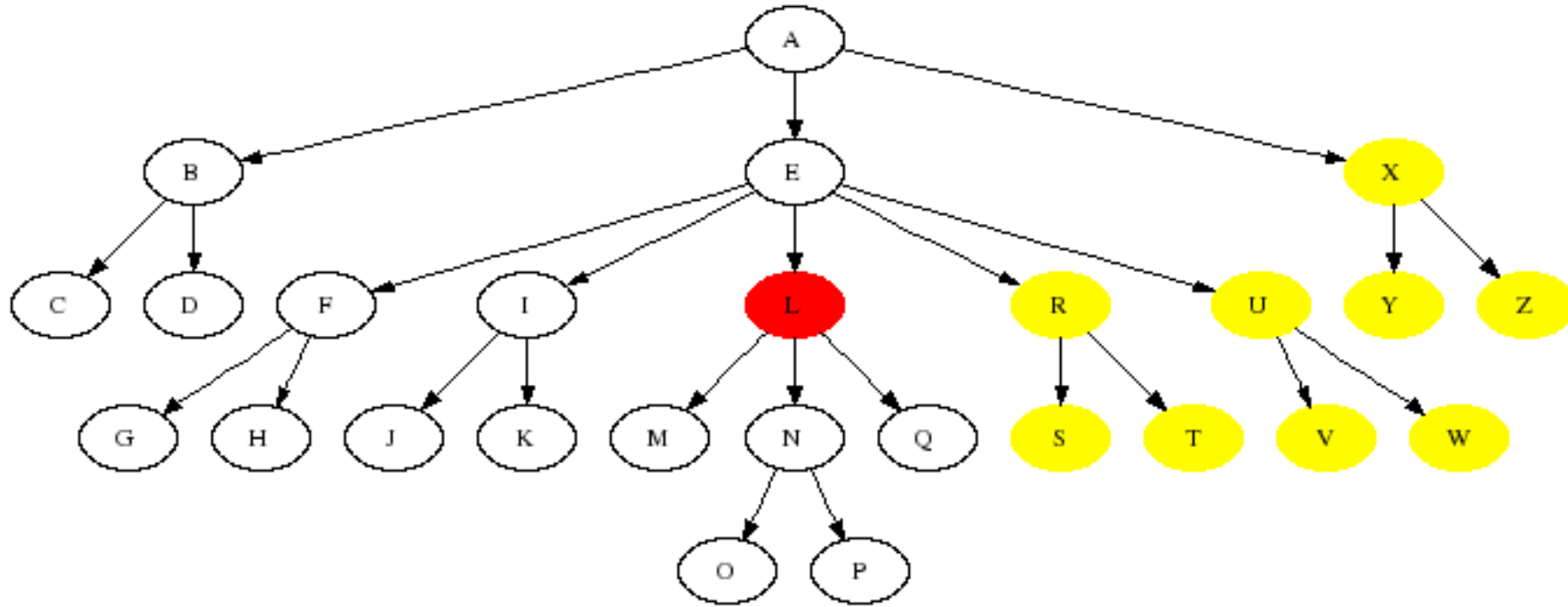
self = nodo di contesto (rosso)



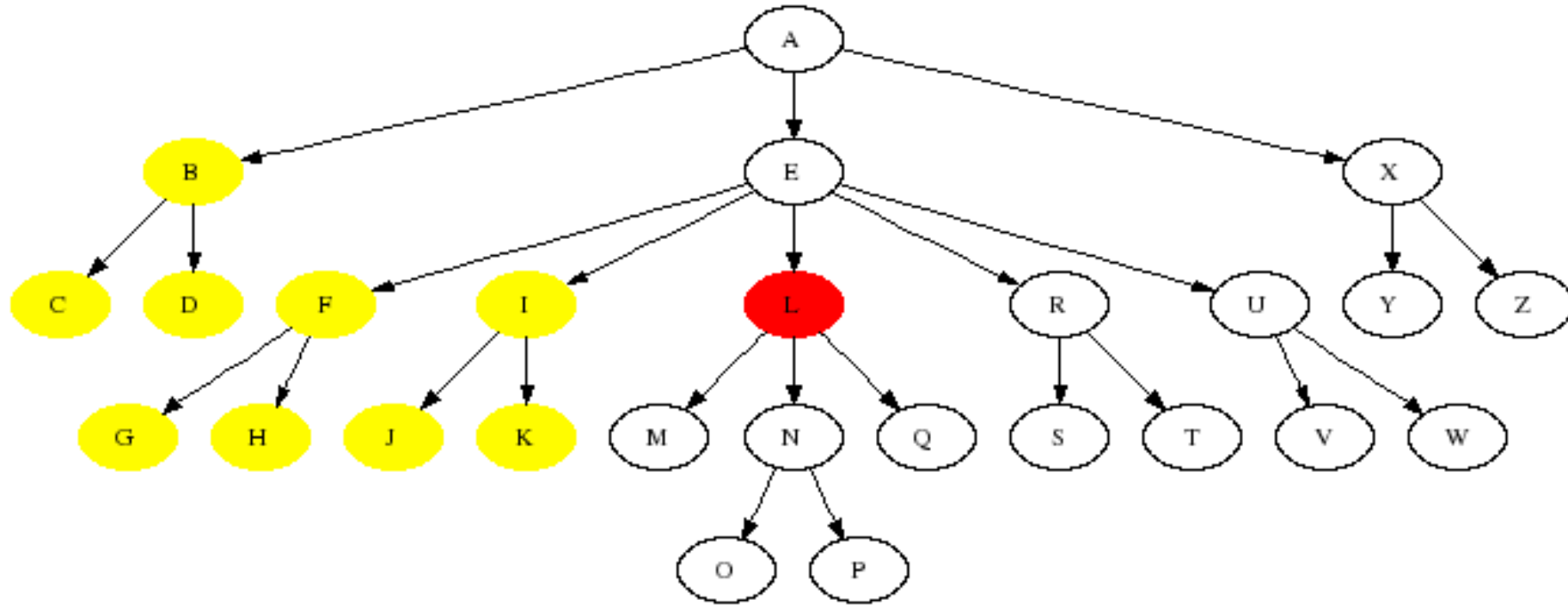
**ASSE following-sibling::\***



**ASSE preceding-sibling::\***



**ASSE following:\***



**ASSE preceding:\***



# ABBREVIAZIONI ASSI

Per alcuni assi più comuni sono previste delle abbreviazioni

- **child::a** (abbr. 'a')
- **descendant-or-self::a** (abbr. '//a')
- **self::\*** (abbr. '.')
- **parent::\*** (abbr. '..')
- **attribute::a** seleziona l'attributo 'a' del nodo di contesto (abbr. '@a')

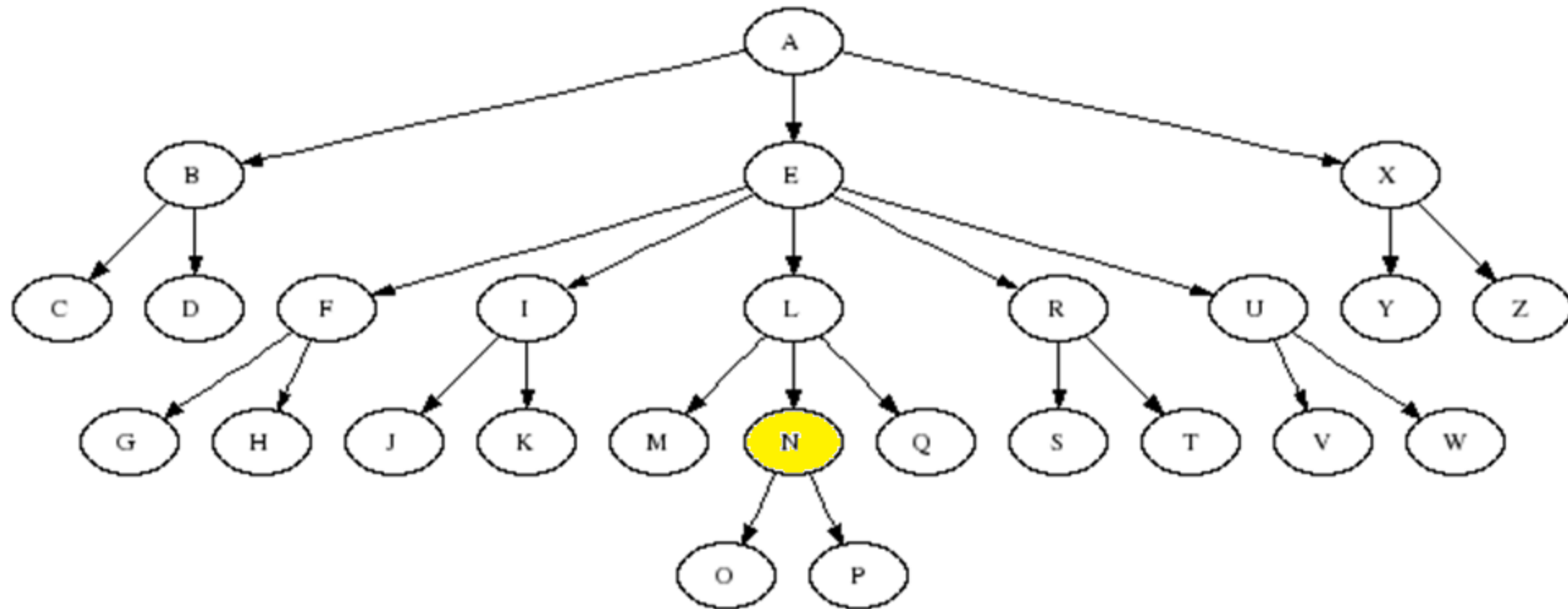
# OPERATORI E FUNZIONI

Finora abbiamo visto come selezionare degli elementi navigando nell'albero del documento XML.

XPath offre inoltre

- **operatori di comparazione:** =, !=, <, <=, >, >=
- **operatori aritmetici:** +, -, \*, div (divisione), mod (modulo)
- **funzioni:** su insiemi di nodi, su stringhe e su numeri

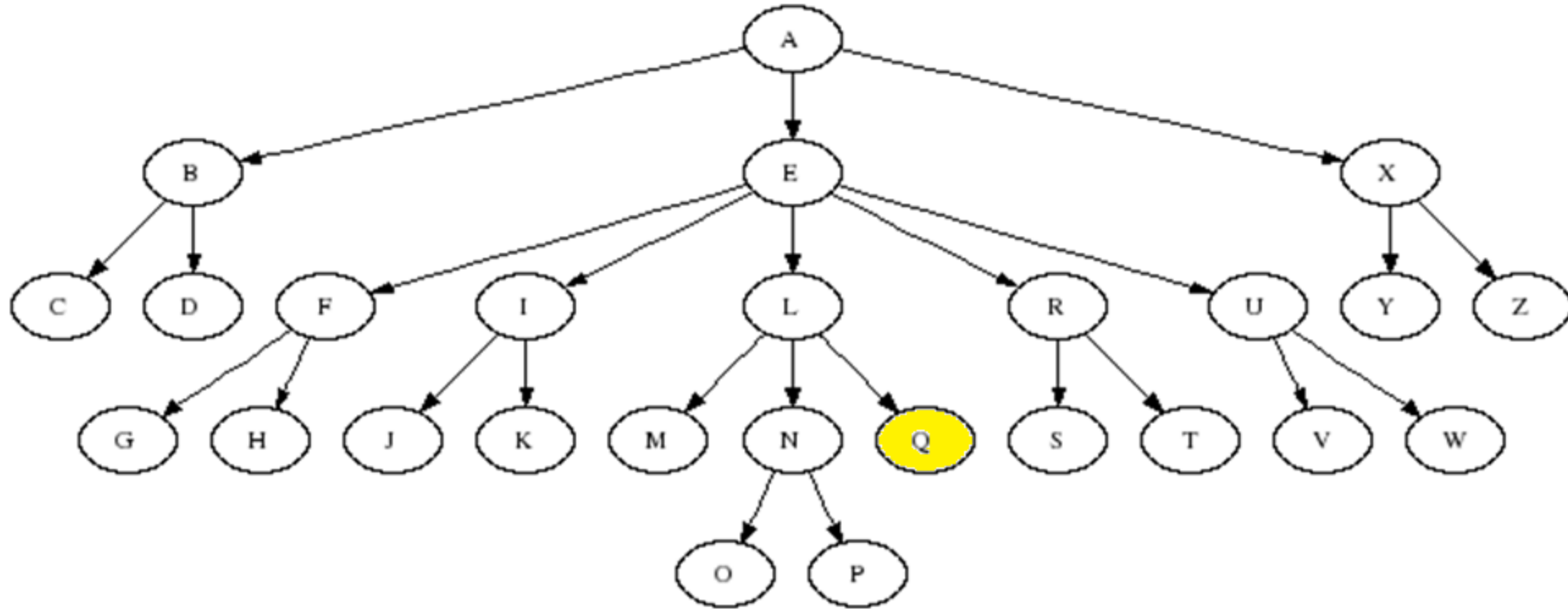
# FUNZIONI SU INSIEMI DI NODI: POSITION()



**/A/E/L/\* [position()=2]**

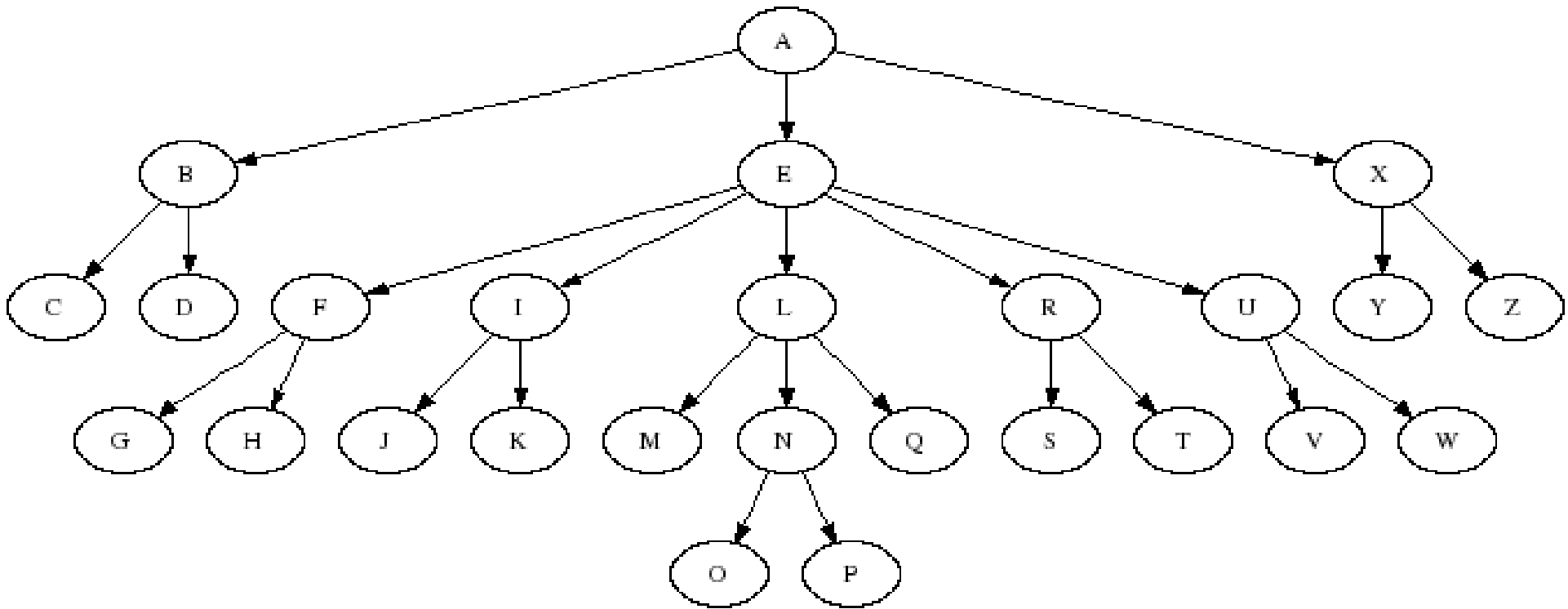
abbr: /A/E/L/\*[2]

# FUNZIONI SU INSIEMI DI NODI: LAST()



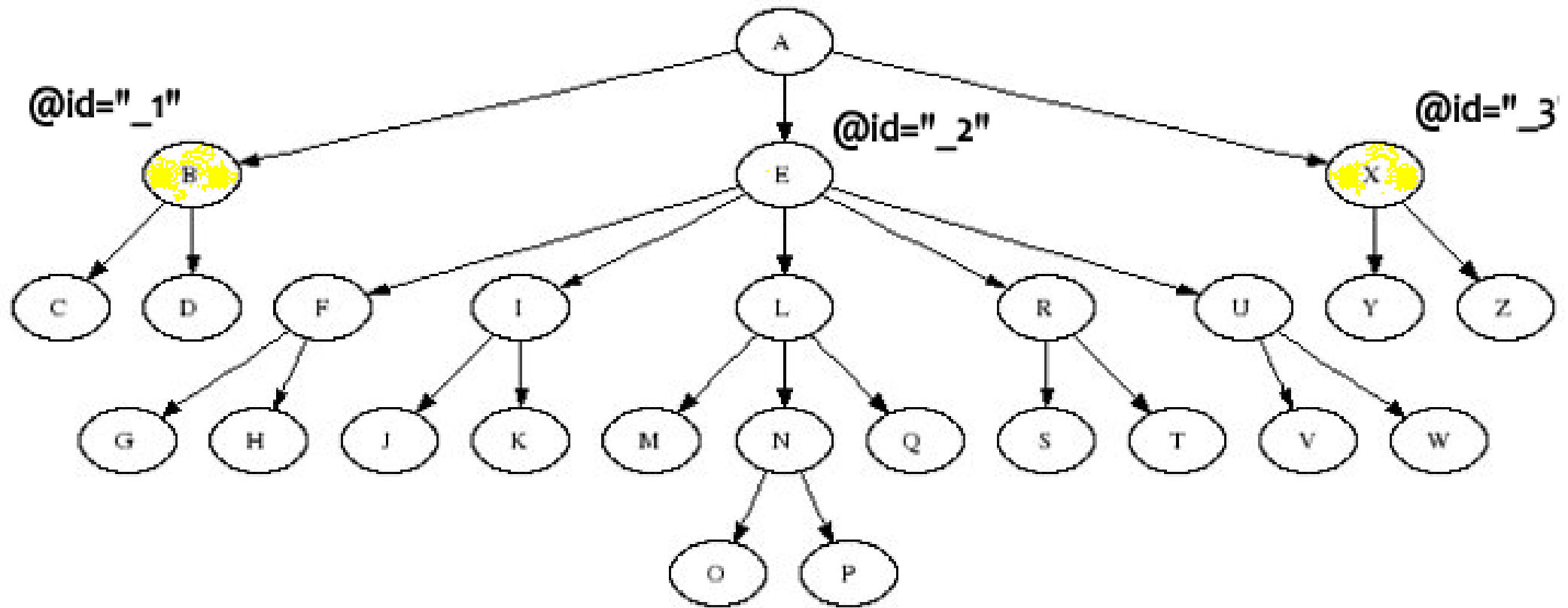
//L/\*[last()]

# FUNZIONI SU INSIEMI DI NODI: COUNT(PATH)



**count (//L/\*)**  
(restituisce 3)

# FUNZIONI SU INSIEMI DI NODI: ID("IDREF(S)")



id("\_1 \_3")

# FUNZIONI SU STRINGHE

Le funzioni su **stringhe** più rilevanti sono

- `contains(string1, string2)` : restituisce `true` se la prima stringa contiene la seconda
- `starts-with(string1, string2)` : restituisce `true` se la prima stringa inizia con la seconda (`string2` è un prefisso di `string1`)
- `ends-with(string1, string2)` : restituisce `true` se la prima stringa termina con la seconda (`string2` è un suffisso di `string1`)
- `string-length(string)` : restituisce la lunghezza della stringa

# FUNZIONI SU NUMERI

Le funzioni su **numeri** più rilevanti sono

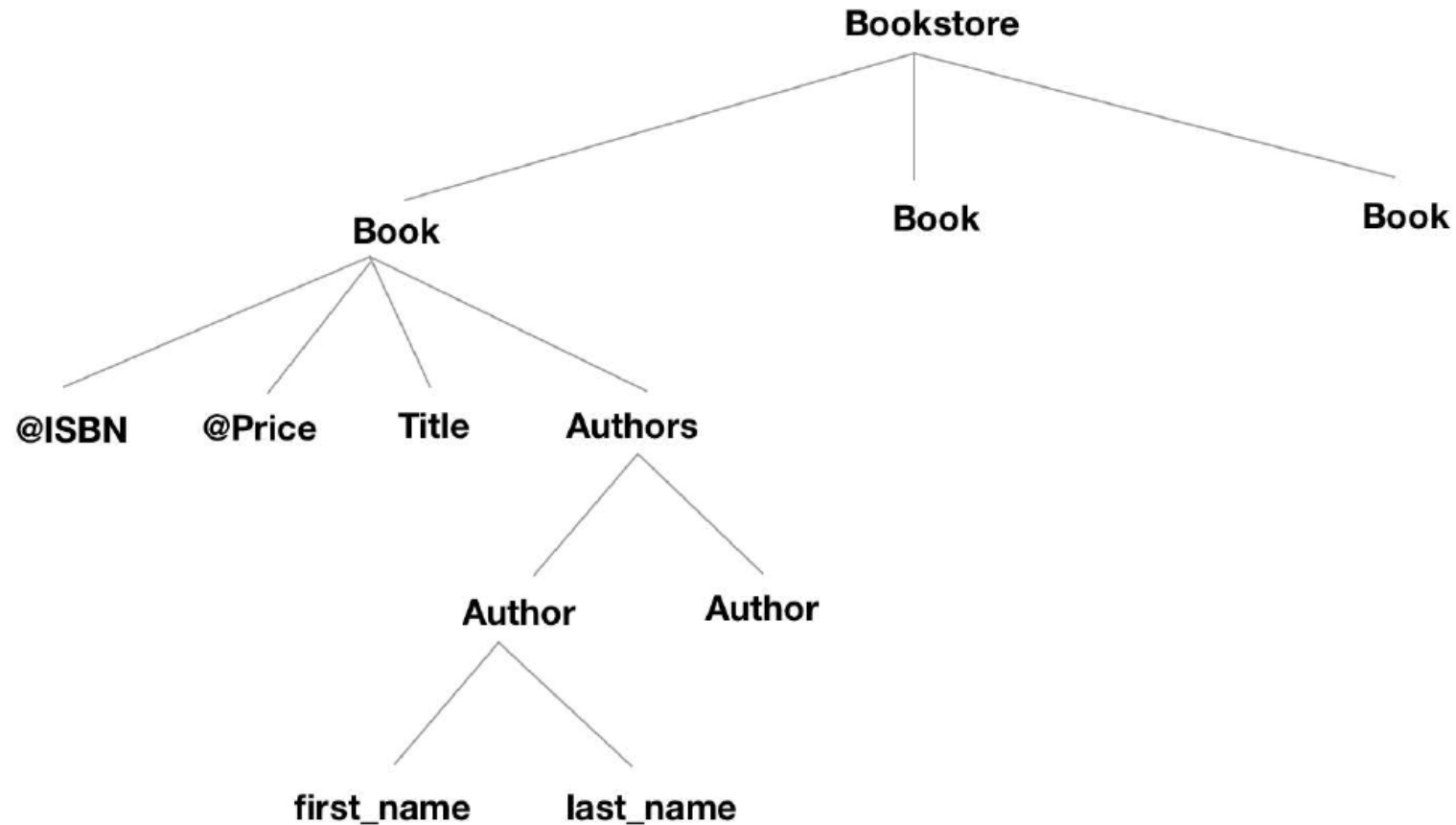
- `round(num)` : arrotonda `num` al numero intero più vicino
- `floor(num)` : arrotonda `num` per difetto
- `ceiling(num)` : arrotonda `num` per eccesso
- `sum(path)` : restituisce la somma del contenuto di tutti i nodi selezionati con il percorso di locazione `path`



# ESEMPIO BOOKSTORE

```
<!DOCTYPE Bookstore [  
  <!ELEMENT Bookstore (Book | Magazine)*>  
  <!ELEMENT Book (Title, Authors, Remark?)>  
  <!ATTLIST Book ISBN CDATA #REQUIRED  
                Price CDATA #REQUIRED  
                Edition CDATA #IMPLIED>  
  <!ELEMENT Magazine (Title)>  
  <!ATTLIST Magazine Month CDATA #REQUIRED Year CDATA #REQUIRED>  
  <!ELEMENT Title (#PCDATA)>  
  <!ELEMENT Authors (Author+)>  
  <!ELEMENT Remark (#PCDATA)>  
  <!ELEMENT Author (First_Name, Last_Name)>  
  <!ELEMENT First_Name (#PCDATA)>  
  <!ELEMENT Last_Name (#PCDATA)>  
>
```

# ESEMPIO BOOKSTORE: STRUTTURA LOGICA



# ESEMPIO BOOKSTORE: QUERY

Restituisce il titolo dei libri scritti dall'autore con cognome "Ullman" e con prezzo inferiore a 90

```
/Bookstore/Book[@Price<90 and Authors/Author/Last_Name="Ullman"]
```

Provare qui: <https://www.freeformatter.com/xpath-tester.html#ad-output> oppure

```
xmllint --xpath '/Bookstore/Book[@Price<90 and Authors/Author/Last_Name="Ullman"]' Bookstore.xml
```

# ESEMPIO BOOKSTORE: SOTTOMISSIONE QUERY

**Formatters**

- JSON Formatter
- HTML Formatter
- XML Formatter
- SQL Formatter

**Validators**

- JSON Validator
- HTML Validator
- XML Validator - XSD
- XPath Tester**
- Credit Card Number Generator & Validator
- Regular Expression Tester (Regex)
- Java Regular Expression Tester (Regex)
- Cron Expression Generator - Quartz

**Encoders & Decoders**

- Url Encoder & Decoder
- Base 64 Encoder & Decoder
- Convert File Encoding
- QR Code Generator

**Code Minifiers / Beautifier**

- JavaScript Beautifier
- CSS Beautifier
- JavaScript Minifier
- CSS Minifier

**Converters**

- XSD Generator
- XSLT (XSL Transformer)
- XML to JSON Converter
- JSON to XML Converter
- CSV to XML Converter
- CSV to JSON Converter
- Epoch Timestamp To Date

**Cryptography & Security**

-

## XPath Tester / Evaluator

Allows you to test your XPath expressions/queries against a XML file. This tool runs better than other existing XPath online tools as it supports most of the XPath functions (string(), number(), name(), string-length() etc.) and does not limit you to working against nodes. It fully supports XPath 2.0 / 3.0 specification. See the [XPath Examples](#) section for details.

The XPath tester fully supports XML namespaces. See the XPath Examples section for details. The namespace prefix "fn" and "math" are reserved to XPath functions.

\*The maximum size limit for file upload is 2 megabytes. Results bigger than 500k will be written to a new window for performance reason and to prevent your browser from being unresponsive.

### XML Input

Option 1: Copy-paste your XML document here

```
<?xml version="1.0" ?>
<!-- Data for XPath, XQuery, and XSLT demos -->

<!DOCTYPE Bookstore [
  <ELEMENT Bookstore (Book | Magazine)*>
  <ELEMENT Book (Title, Authors, Remark?)>
```

Option 2: Or upload your XML document

Nessun file selezionato

UTF-8

### XPath expression

/Bookstore/Book[@Price<90 and Authors/Author/Last\_Name="Ullman"]

Include the XML item type in output:

Yes

TEST XPATH

TEST XPATH IN NEW WINDOW

### XPath result:

```
Element='<Book Edition="3rd" ISBN="ISBN-0-13-713526-2" Price="85">
  <Title>A First Course in Database Systems</Title>
```

# ESERCIZIO BOOKSTORE

Scrivere le query per reperire le seguenti informazioni

1. Autori presenti in seconda posizione
2. First\_Name di tutti gli autori presenti
3. Titolo dei libri con osservazioni (Remark) contenenti la parola "great"