

# Informazione semi-strutturata

*XML – eXtended Markup Language*



UNIVERSITÀ DI PISA

## Linguaggi di marcatura

- ◆ **Un linguaggio di marcatura è un insieme di convenzioni per la marcatura di testi.**
- ◆ **Marcatura di documenti**
  - *Marcatura* (o etichettatura) è un qualcosa che permette di rendere esplicita un'interpretazione di un testo.
  - storicamente: annotazioni in un testo che descrivono al tipografo come stampare o comporre una parte del testo
  - oggi: qualsiasi tipo di codice inserito in un testo in forma elettronica

## Linguaggi di marcatura

- ◆ **Un linguaggio di marcatura è un insieme di convenzioni per la marcatura di testi.**
- ◆ **Deve specificare:**
  - quali marcature sono permesse
  - quali marcature sono obbligatorie
  - come vanno composte le marcature
  - come distinguere tra marcatura e testo
- ◆ **Può inoltre specificare il significato della marcatura.**

3

## Tipi di marcatura

- ◆ **Si distinguono due tipi di marcatura:**
- ◆ **marcatura procedurale**
  - descrive come processare il documento
  - Esempi: postscript, pdf, rtf, formato di word
- ◆ **marcatura *descrittiva***
  - descrive la struttura logica del documento
  - Esempi: HTML, SGML, XML

4

## SGML - Standard Generalized Markup Language

- ◆ È il padre degli attuali linguaggi di marcatura
- ◆ È un linguaggio che permette di definire linguaggi di marcatura (è un *metalinguaggio*):
  - estremamente espressivo e configurabile (troppo)
  - alta espressività rende il processamento complicato
  - utilizzato in grossi progetti di documentazione
  - non studiato espressamente per il Web
- ◆ **Manca di alcune caratteristiche fondamentali per il Web:**
  - gestione dei link
  - gestione del conflitto sui nomi delle etichette
  - tutti i documenti devono essere ``validi" (oltre a essere ``ben formati")
- ◆ È troppo complicato per poter essere adoperato come linguaggio di marcatura

5

## HTML - HyperText Markup Language

- ◆ è un linguaggio di marcatura
- ◆ definito in termini di SGML
- ◆ insieme di etichette prefissato (RIGIDO)
- ◆ marcatura non denota il significato
- ◆ studiato espressamente per il Web
  - collegamenti ipertestuali
  - immagini

6

## Limitazioni di HTML

- ◆ **Insieme di etichette prefissato**
  - cosa uso / cosa è stato usato per rappresentare le informazioni di interesse
  
- ◆ **Marcatura non denota il significato**
  - che insieme di informazioni rappresenta una pagina / una collezione di pagine?
  - come faccio a sapere quali sono le informazioni che mi interessano?
  - come faccio ad estrarre le informazioni che mi interessano (e solo quelle)?
  
- ◆ **Marcatura usata per la presentazione del documento - HTML è adatto, anche se presenta delle limitazioni**

7

## Il Web oggi

- ◆ **Composto da una collezione (molto grande) di pagine Web**
- ◆ **Ogni pagina è un *documento* strutturato codificato in HTML**
- ◆ **Le informazioni nel Web di oggi**
  - HTML descrive la struttura logica dell'informazione allo scopo di *presentarla come pagina Web*
  - molte pagine ancora generate manualmente
  - sempre più pagine generate da applicazioni a partire da basi di dati
- ◆ **Le pagine HTML sono destinate alla sola rappresentazione ma non al riutilizzo**
  - marcatura descrive la presentazione
  - non c'è un legame diretto tra marcatura e informazione

8

## Una pagina HTML

```

<h1>Libro</h1>
<p>
  <i>I Promessi Sposi</i>.<br>
  <b>Alessandro Manzoni</b>.<br>
  La Padana, 1982.<br>
  Romanzo storico sulle traversie d'amore. Un po' datato in quanto
  ...
</p>

<b>Riferimenti bibliografici</b>
<ol>
  <li><i>Il Decamerone, </i>
    Francesco Boccaccio.
    <i>La Toscana</i>, 2000.
  </li>
  <li><i>Il Vangelo, </i>
    Matteo, Marco, Luca, Giovanni.
    <i>La Giudea</i>, 1000.
  </li>
  ...
</ol>

```

9

## Il Web domani

- ◆ Documenti strutturati e *dati* vengono generati ed *utilizzati* da applicazioni e da umani
- ◆ documenti visualizzati in pagine Web
- ◆ pagine Web generate a partire da dati strutturati (basi di dati)
- ◆ dati estratti da documenti sul Web
- ◆ dati scambiati via Web tra organizzazioni diverse
- ◆ *HTML non è adatto* – potrà essere utilizzato solo per la visualizzazione finale

10

## Requisiti per il Web di domani

- ◆ Serve un formalismo *più flessibile*:
- ◆ separazione tra:
  - *contenuto*
  - *presentazione*
  - *navigazione*
- ◆ definizione di *domini* o contesti
- ◆ indipendenza dalla piattaforma (*media*) e supporto multilingue

11

## Il Web domani (ma già anche oggi)- Esempio di Documento XML

```

<Libro>
  <titolo>I Promessi Sposi</titolo>
  <autore> <Nome> Alessandro </Nome> <Cognome>
Manzoni</Cognome> </autore>
  <editore>La Padania</editore>
  <anno>1982</anno>
  <commento>Romanzo storico sulle traversie
d'amore. Un po' datato in quanto ... </commento>

```

12

## Il Web domani (ma già anche oggi)- Esempio di Documento XML

<Bibliografia>

<Citazione>  
<Ctitolo>Il Decamerone</Ctitolo>

<Cautore>  
<Cnome>Francesco</Cnome>  
<Ccognome>Boccaccio</Ccognome>  
</Cautore>

<Ceditore>La Toscana</Ceditore>  
<Canno>2000</Canno>  
</Citazione>

13

## Il Web domani (ma già anche oggi)- Esempio di Documento XML

<Citazione>  
<Ctitolo>Il Vangelo</Ctitolo>

<Cautore> <Cnome>Matteo</Cnome> </Cautore>

<Cautore> <Cnome>Marco</Cnome> </Cautore>

<Cautore> <Cnome>Luca</Cnome> </Cautore>

<Cautore> <Cnome>Giovanni</Cnome> </Cautore>

<Ceditore>La Giudea</Ceditore>  
<Canno>1000</Canno>  
</Citazione>  
</Libro>

14

## Documenti XML

- ◆ **Il documento XML è verboso ma logicamente ben strutturato**
  - contiene contemporaneamente i nomi dei campi e i loro valori
- ◆ **Il modello relazione è più semplice perchè una ennupla di una tabella riporta solo i valori mentre lo schema (nomi e tipi dei campi) sono memorizzati una volta per tutte a parte.**

15

## Il linguaggio XML

- ◆ **XML = eXtensible Markup Language**
- ◆ **linguaggio di marcatura descrittiva**
- ◆ **sta suscitando enorme interesse**
  - a livello industriale
  - nella comunità scientifica
- ◆ **naturale successore di HTML come linguaggio per il Web**
  - più espressivo e flessibile (eXtensible)
  - più complicato
  - a lungo andare semplificherà la vita agli sviluppatori Web
- ◆ **ma da utilizzare solo per descrivere il contenuto logico dei dati e non la loro visualizzazione**
  - per la visualizzazione è necessario una descrizione a parte con un altro linguaggio (XSL) che genera una pagina HTML che (forse) resterà solo per la rappresentazione grafica
  - ma per lo stesso documento XML è possibile definire più visualizzazione

16



## Le origini di XML

- ◆ **1969**
  - Charles Goldfarb (IBM) dirige lo sviluppo di *GML*
- ◆ **1974**
  - Charles Goldfarb inventa *SGML*, il padre dei linguaggi di marcatura
- ◆ **1986**
  - *SGML* diventa uno *standard ISO* (ISO 8879 "Information Processing - Text and Office Systems - Standard Generalized Markup Language")
- ◆ **1989**
  - Tim-Berners Lee (CERN di Ginevra) inventa *HTML*
- ◆ **1995**
  - Fondazione del World Wide Web Consortium (*W3C*)

17

## Le origini di XML

- ◆ **1996**
  - Inizio dello sviluppo di *XML* presso il *W3C*
- ◆ **1998**
  - *XML 1.0* diventa una *raccomandazione W3C* (uno standard di fatto)
- ◆ **1996-oggi**
  - Sviluppo di *standard associati* ad *XML* (coordinato da *W3C*)
- ◆ **2002**
  - *XML 1.1* diventa una *raccomandazione candidata W3C* (Il consorzio internazionale per la standardizzazione di linguaggi e strumenti per il *WEB*)

18

## Obiettivi di sviluppo di XML

- ◆ XML deve essere usabile direttamente sul Web
- ◆ XML deve essere compatibile con SGML
- ◆ deve essere semplice progettare programmi che elaborano documenti XML
- ◆ documenti XML devono essere leggibili da umani e sufficientemente chiari
- ◆ documenti XML devono essere facili da creare
- ◆ la specifica di XML deve essere formale e rigorosa (anche se non concisa)
- ◆ la compattezza nella marcatura è di poca importanza

19

## Caratteristiche dei documenti XML

- ◆ marcatura *descrittiva* e non procedurale (descrive la struttura logica)
- ◆ marcatura è dettata dalla struttura logica del documento
- ◆ insieme di etichette può cambiare in base l'applicazione
- ◆ viene usato il concetto di *tipo di documento*
  - specificato attraverso una *Document Type Declaration* o *DTD* (è parte dello standard XML)
  - permette di imporre al documento una certa struttura (ovvero come si compongono le sue parti)
  - le parti sono delimitate dalla marcatura

20

## Specifiche aggiuntive

- ◆ **Si rendono necessarie per rendere XML funzionale sul Web**
- ◆ **presentazione del documento (fogli di stile): XSL (XSLT), CSS**
- ◆ **significato della marcatura**
  - collegamenti ipertestuali : XPath, XLink, XPointer
  - semantica : RDF, OIL
- ◆ **meccanismi più flessibili per la specifica della struttura: XML-Schema, DSD**
- ◆ **linguaggi di interrogazione per XML : XQuery, XML-QL, XSL**

21

## Specifiche aggiuntive

- ◆ **XML permette di definire nuovi linguaggi di marcatura**
  - etichette specifiche per il tipo di applicazione
  - modalità di visualizzazione possono essere definite (non direttamente in XML, ma tramite gli standard associati)
- ◆ **XML come EDI (Electronic Data Interchange)**
  - fornisce un meccanismo generico per rappresentare dati
  - fornisce un meccanismo generico per scambiare dati

22

## Forma (sintassi) di un documento XML

- ◆ Un documento XML è costituito da:
  - ◆ un *prologo* (opzionale), costituito da
    - una dichiarazione XML
    - una DTD
  - ◆ un'istanza di documento costituito da un insieme di elementi annidati
  - ◆ un elemento è costituito da una coppia di etichette di apertura e di chiusura e da tutto quello che racchiudono

23

## Forma (sintassi) di un documento XML

- ◆ `<autore>`
  - `<nome> Alessandro </nome>`
  - `<cognome> Manzoni </cognome>`
  - `</autore>`
  - *nome* dell'elemento esterno : autore
  - *etichetta di apertura* dell'elemento: `<autore>`
  - *etichetta di chiusura* dell'elemento: `</autore>`
  - *contenuto* dell'elemento: due elementi nome e cognome
  - contenuto dell'elemento nome: la stringa Alessandro
  - contenuto dell'elemento cognome: la stringa Manzoni

24

## Sintassi XML

- ◆ Il *contenuto* di un elemento è costituito da:
  - altri elementi (detti *figli*), delimitati da coppie di etichette
  - testo libero (non contenente marcatura), detto #PCDATA
  - altro che però non trattiamo nel corso
- ◆ le etichette devono essere *annidate correttamente*
- ◆ Esempio di documento non ben formato (la chiusura di nome deve precedere quella di autore):
 

```

      <libro>
        <autore><nome>Alessandro</autore></nome>
      </libro>
```
- ◆ ci deve essere *un solo elemento radice* (cioè la prima etichetta non può essere ripetuta più volta)

25

## Elementi vuoti

- ◆ Il contenuto di un elemento può essere vuoto.
- ◆ Due modi di denotare un *elemento vuoto*:
  - coppia di etichette di apertura e chiusura: <vuoto></vuoto>
  - etichetta di elemento vuoto: <vuoto/>
- ◆ **Uso di elementi vuoti**
  - Non sono inutili in quanto aggiungono informazione al documento attraverso attributi associati all'elemento

26

## Nomi di elemento

- ◆ XML è *case-sensitive* - esempio non ben formato:

```
<elenco-clienti>
  <cliente><codice>CC128</codice> ... </CLIENTE>
</Elenco-Clienti>
```

- ◆ Errore comune: dimenticare "/" nell'etichetta di chiusura - es:

```
<elenco-clienti>
  <cliente><codice>CC128</codice> ... <cliente>
</elenco-clienti>
```

27

- ◆ I nomi di elementi possono contenere solo: lettere, cifre, -, \_, :
- ◆ Nomi che iniziano con XML,xml,xML,...sono riservati.

<pre>&lt;nomiPermessi&gt;   &lt;xsl:template/&gt;   &lt;Nome_elemento_lungo/&gt;   &lt;Altro-nome-lungo/&gt;   &lt;nome.con.punti/&gt;   &lt;a1233-231-231/&gt;   &lt;_12/&gt; &lt;/nomiPermessi&gt;</pre>	<pre>&lt;nomi+Vietati&gt;   &lt;questiNO@#\$\$%^()+?*;\$/&gt;   &lt;Un;nome*2/&gt;   &lt;XmL-riservato/&gt;   &lt;8-inizia-con-cifra/&gt;   &lt;niente spazi/&gt;   &lt;riservati&lt;&amp;&gt;&gt; &lt;/nomi+Vietati&gt;</pre>
--	--

28

## Caratteri riservati

- ◆ **Il testo non può contenere i caratteri riservati per la marcatura**
  - carattere "<" (denota l'inizio di un'etichetta)
  - carattere "&" (per le entità - più avanti)
- ◆ **Si usano**
  - "&lt;"per"<" "&amp;"per"&"
  - <editore>Wiley &amp; sons</editore>
  - <disequazione>x\*y &lt; z</disequazione>
- ◆ **I caratteri ">", """, e """" possono essere sostituiti da "&gt;", "&quot;" e "&apos;".**

29

## Commenti

- ◆ servono a escludere una parte di documento dall'elaborazione
- ◆ possono comparire ovunque all'esterno della marcatura
- ◆ un processore XML può o meno rendere disponibili le parti di documento racchiuse tra commenti
- ◆ sono delimitati da <!-- e -->
- ◆ possono contenere qualsiasi carattere (inclusi "<" e "&"), tranne "-"
- ◆ non possono essere annidati

30

## Dichiarazione XML iniziale

- ◆ Documenti XML possono (e in realtà dovrebbero) iniziare con una *dichiarazione XML* che specifica la versione di XML utilizzata

```
<?xml version="1.0"?>
<testo>Questo documento e' conforme alla specifica
di XML 1.0.</testo>
```

- ◆ La dichiarazione XML può anche specificare la *codifica dei caratteri* usati:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<testo>Se non viene specificata la codifica,
si assume UTF-8</testo>
```

31

## Tipo di documento (DTD): schema di un documento XML

- ◆ Un documento XML *ben formato* può avere una *struttura arbitraria*:
  - elementi di nome arbitrario
  - attributi arbitrari in qualunque elemento
  - elementi organizzati in modo arbitrario
- ◆ Sono utili, ma poco più utili di testo non strutturato.
- ◆ Serve un meccanismo per *imporre struttura* ad un documento:
  - elementi ammessi
  - attributi ammessi
  - strutturazione degli elementi
  - definizione di nuove entità (interne ed esterne)

32



## Dichiarazione di tipo di documento (DTD)

- ◆ Un *DTD* specifica quali sono le strutture ammesse per l'istanza di documento:
- ◆ specifica il *tipo del documento*, ovvero il tipo dell'elemento radice
- ◆ contiene tre insiemi di *dichiarazioni*:
  - di *tipi di elemento*
  - di *attributi*
  - di *entità*
- ◆ L'insieme di tipi di elemento in un DTD viene detto *vocabolario*.
- ◆ Un DTD corrisponde allo schema di una tabella mentre un documento XML corrisponde a una ennupla di quella tabella.

33

## Dichiarazioni in XML

- ◆ In XML tutte le *dichiarazioni* hanno la forma:
- ◆ `<!OGGETTO-DICHIARATO ... >`
- ◆ Esempi:
  - `<!DOCTYPE ... >`
  - `<!ELEMENT ... >`
  - `<!ATTLIST ... >`
  - `<!ENTITY ... >`
  - `<!NOTATION ... >`
- ◆ In questo corso tratteremo solo le prime due (DOCTYPE,ELEMENT) ma faremo anche qualche cenno a ATTLIST.

34

## Dichiarazione di tipo di documento

- ◆ Un documento XML *può* contenere una DTD.
- ◆ La DTD *deve* precedere il primo elemento (l'elemento radice).
- ◆ Le dichiarazioni nella DTD possono essere:
  - *interne* al documento stesso  

```
<!DOCTYPE esempio [  
  <ELEMENT esempio (#PCDATA)> ]>
```
  - *esterne* al documento in un altro file  

```
<!DOCTYPE esempio SYSTEM "esempio.dtd">
```

35

## Dichiarazioni di tipo di elemento

- ◆ Sintassi di una dichiarazione di tipo di elemento:
  - `<ELEMENT nome-elemento content-model>`
- ◆ Il *content model* specifica la struttura degli elementi di nome *nome-elemento*.
- ◆ *content model* è un'espressione regolare costruita su:
  - i tipi di elemento
  - `#PCDATA`, che rappresenta del testo libero (senza etichette)
- ◆ Ci sono delle limitazioni nell'uso di `#PCDATA` nel *content model* che vedremo più avanti.

36

## Forma del content model

- ◆ **Operatori usati nel content model sono quelli delle espressioni regolari:**
  - *sequenza*, indicata con  $(a,b,\dots,h)$
  - *alternativa*, indicata con  $(a | b | \dots | k)$
  - *opzionalità*, indicata con  $a?$
  - *ripetizione zero o più volte*, indicata con  $a^*$
  - *ripetizione una o più volte*, indicata con  $a^+$
  
- ◆ **(dove  $a,b,\dots$  indicano un generico content model)**

37

## Documenti XML validi

- ◆ **Un documento XML viene detto *valido* se:**
  - è *ben formato*, ovvero rispetta tutti vincoli imposti da XML
  - contiene una *dichiarazione di tipo di documento*
  - è *conforme* alla DTD
  
- ◆ **Osservazione: lo spazio bianco *non* è rilevante (di norma)**

38

## Produzione di documenti XML a partire da uno schema concettuale –singola classe

Studente
<i>Matricola</i>
<i>Nome</i>
<i>DataNascita</i>

```

◆ <!DOCTYPE Studente [
  <!ELEMENT Studente (Matricola, Nome, DataNascita)>
  <!ELEMENT DataNascita (Giorno, Mese, Anno)>
  <!ELEMENT Matricola (#PCDATA)>
  <!ELEMENT Nome (#PCDATA)>
  <!ELEMENT Giorno (#PCDATA)>
  <!ELEMENT Mese (#PCDATA)>
  <!ELEMENT Anno (#PCDATA)>
]>

```

39

- ◆ Un documento XML conforme al DTD specificato corrisponde a una ennupla. Ad esempio:

```

<!DOCTYPE Studente SYSTEM "Studente.dtd!>
<Studente>
  <Matricola> 10101 </Matricola>
  <Nome> Francesco Rossi </Nome>
  <DataNascita>
    <Giorno> 10 </Giorno>
    <Mese> 11 </Mese>
    <Anno> 1980 </Anno>
  </DataNascita>
</Studente>

```

40

## DTD alternativo

- ◆ E' anche possibile rappresentare tutta una tabella in un documento XML utilizzando il seguente schema.

```
<!DOCTYPE ElencoStudenti [
  <!ELEMENT ElencoStudenti (Studente+)>
  <!ELEMENT Studente (Matricola, Nome, DataNascita)>
  <!ELEMENT DataNascita (Giorno, Mese, Anno)>
  <!ELEMENT Matricola (#PCDATA)>
  <!ELEMENT Nome (#PCDATA)>
  <!ELEMENT Giorno (#PCDATA)>
  <!ELEMENT Mese (#PCDATA)>
  <!ELEMENT Anno (#PCDATA)>
]>
```

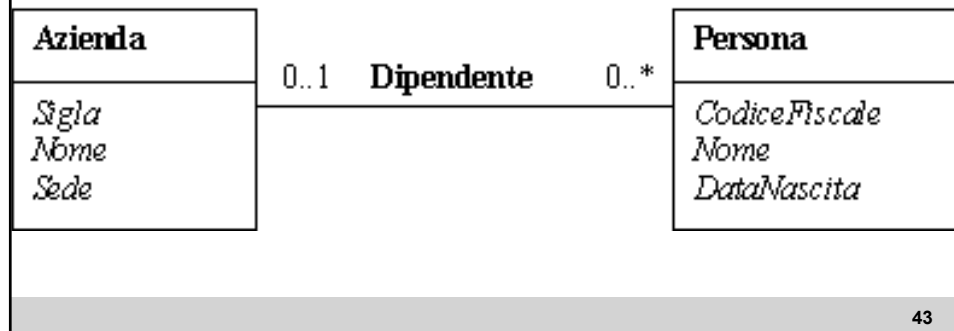
41

```
<ElencoStudenti>
  <Studente>
    <Matricola> 10101 </Matricola>
    <Nome> Francesco Rossi </Nome>
    <DataNascita>
      <Giorno> 10 </Giorno> <Mese> 11 </Mese> <Anno> 1980 </Anno>
    </DataNascita>
  </Studente>
  <Studente>
    <Matricola> 20000 </Matricola>
    <Nome> Francesco Bianchi </Nome>
    <DataNascita>
      <Giorno> 1 </Giorno> <Mese> 12 </Mese> <Anno> 1980 </Anno>
    </DataNascita>
  </Studente>
</ElencoStudenti>
```

42

## Associazioni uno a molti

- ◆ Vediamo adesso come rappresentare utilizzando un DTD non solo singole entità ma anche associazioni fra entità.
- ◆ Consideriamo la seguente associazione 1 ad N:



## Associazione uno a molti

- ◆ “Navigando” lo schema da sinistra a destra (da Azienda verso Persona) otterremo il seguente DTD:

```

<!DOCTYPE Azienda [
  <!ELEMENT Azienda    (Sigla, Nome, Sede, Persona*)>
  <!ELEMENT Persona    (CodiceFiscale, Nome, DataNascita)>
  <!ELEMENT Sigla      (#PCDATA)>
  <!ELEMENT Nome       (#PCDATA)>
  <!ELEMENT Sede       (#PCDATA)>
  <!ELEMENT CodiceFiscale(#PCDATA)>
  <!ELEMENT DataNascita (Giorno, Mese, Anno)>
  <!ELEMENT Giorno     (#PCDATA)>
  <!ELEMENT Mese       (#PCDATA)>
  <!ELEMENT Anno       (#PCDATA)>
]>

```

44

```

<!DOCTYPE Azienda SYSTEM "Azienda.dtd">
  <Azienda>
    <Sigla> SMS </Sigla>
    <Nome> Skill Management System </Nome>
    <Sede> Pisa </Sede>
    <Persona>
      <CodiceFiscale> GRKLP200 </CodiceFiscale>
      <Nome> Francesco Bianchi </Nome>
      <DataNascita>
        <Giorno> 10 </Giorno> <Mese> 11 </Mese> <Anno> 1980 </Anno>
      </DataNascita>
    </Persona>
    <Persona>
      <CodiceFiscale> GDFTP200 </CodiceFiscale>
      <Nome> Francesco Rossi </Nome>
      <DataNascita>
        <Giorno> 10 </Giorno> <Mese> 11 </Mese> <Anno> 1980 </Anno>
      </DataNascita>
    </Persona>
    <Persona>
      <CodiceFiscale> GRFG1200 </CodiceFiscale>
      <Nome> Paola Verdi </Nome>
      <DataNascita>
        <Giorno> 10 </Giorno> <Mese> 11 </Mese> <Anno> 1980 </Anno>
      </DataNascita>
    </Persona>
  </Azienda>

```

45

## Un elenco di aziende

```

<!DOCTYPE ElencoAziende [
  <!ELEMENT ElencoAziende(Azienda+)>
  <!ELEMENT Azienda (Sigla, Nome, Sede, Persona*)>
  <!ELEMENT Persona (CodiceFiscale, Nome,
  DataNascita)>
  <!ELEMENT Sigla (#PCDATA)>
  <!ELEMENT Nome (#PCDATA)>
  <!ELEMENT Sede (#PCDATA)>
  <!ELEMENT CodiceFiscale(#PCDATA)>
  <!ELEMENT DataNascita (Giorno, Mese, Anno)>
  <!ELEMENT Giorno (#PCDATA)>
  <!ELEMENT Mese (#PCDATA)>
  <!ELEMENT Anno (#PCDATA)>
]>

```

46

## Navigare lo schema da dx a sx (da Persona verso Azienda)

```
<!DOCTYPE Persona [
  <!ELEMENT Persona (CodiceFiscale, Nome, DataNascita, Azienda?) >
  <!ELEMENT Azienda (Sigla, Nome, Sede)>
  <!ELEMENT Sigla (#PCDATA)>
  <!ELEMENT Nome (#PCDATA)>
  <!ELEMENT Sede (#PCDATA)>
  <!ELEMENT CodiceFiscale(#PCDATA)>
  <!ELEMENT DataNascita (Giorno, Mese, Anno)>
  <!ELEMENT Giorno (#PCDATA)>
  <!ELEMENT Mese (#PCDATA)>
  <!ELEMENT Anno (#PCDATA)>
]>
```

- ◆ Il punto interrogativo indica il vincolo di cardinalità 0..1.

47

```
<!DOCTYPE Azienda SYSTEM "Persona.dtd">
<Persona>
  <CodiceFiscale> GRKLP200 </CodiceFiscale>
  <Nome> Francesco Bianchi </Nome>
  <DataNascita>
    <Giorno> 10 </Giorno> <Mese> 11 </Mese> <Anno> 1980
  </Anno>
  </DataNascita>
  <Azienda>
    <Sigla> SMS </Sigla>
    <Nome> Skill Management System </Nome>
    <Sede> Cosenza </Sede>
  </Azienda>
</Persona>
```

48



## Elenco Persone

```

<!DOCTYPE ElencoPersone [
  <!ELEMENT ElencoPersone(Persone+)>
  <!ELEMENT Persona (CodiceFiscale, Nome, DataNascita, Azienda?) >
  <!ELEMENT Azienda (Sigla, Nome, Sede)>
  <!ELEMENT Sigla (#PCDATA)>
  <!ELEMENT Nome (#PCDATA)>
  <!ELEMENT Sede (#PCDATA)>
  <!ELEMENT CodiceFiscale(#PCDATA)>
  <!ELEMENT DataNascita (Giorno, Mese, Anno)>
  <!ELEMENT Giorno (#PCDATA)>
  <!ELEMENT Mese (#PCDATA)>
  <!ELEMENT Anno (#PCDATA)>
]

```

49

## Associazioni molti a molti

- ◆ Esempio, l'associazione brano – compositore
- ◆ Navigando da compositore a brano

```

<!DOCTYPE Compositore [
  <!ELEMENT Compositore (CodiceSIAE, Nome, DataNascita, Brano+)>
  <!ELEMENT Brano (CodiceBrano, Titolo, Durata)>
  <!ELEMENT CodiceSIAE (#PCDATA)>
  <!ELEMENT Nome (#PCDATA)>
  <!ELEMENT DataNascita (Giorno, Mese, Anno)>
  <!ELEMENT Giorno (#PCDATA)>
  <!ELEMENT Mese (#PCDATA)>
  <!ELEMENT Anno (#PCDATA)>
  <!ELEMENT CodiceBrano (#PCDATA)>
  <!ELEMENT Titolo (#PCDATA)>
  <!ELEMENT Durata (#PCDATA)>
]

```

50

## Associazioni molti a molti

- ◆ Esempio, l'associazione brano – compositore
- ◆ Navigando da brano a compositore

```
<!DOCTYPE Brano [
  <!ELEMENT Brano      (CodiceBrano, Titolo, Durata, Compositore+)>
  <!ELEMENT Compositore (CodiceSIAE, Nome, DataNascita)>
  <!ELEMENT CodiceSIAE (#PCDATA)>
  <!ELEMENT Nome       (#PCDATA)>
  <!ELEMENT DataNascita (Giorno, Mese, Anno)>
  <!ELEMENT Giorno    (#PCDATA)>
  <!ELEMENT Mese      (#PCDATA)>
  <!ELEMENT Anno      (#PCDATA)>
  <!ELEMENT CodiceBrano (#PCDATA)>
  <!ELEMENT Titolo    (#PCDATA)>
  <!ELEMENT Durata    (#PCDATA)>
]>
```

51

## Associazioni ricorsive

- ◆ Utilizzando un DTD è possibile, inoltre, rappresentare associazioni ricorsive (e quindi ontologie):

```
<!DOCTYPE GenereMusicale [
  <!ELEMENT GenereMusicale
            (CodiceGenere, Descrizione, Super?)>
  <!ELEMENT Super      (CodiceGenere)>
  <!ELEMENT CodiceGenere (#PCDATA)>
  <!ELEMENT Descrizione (#PCDATA)>
]>
```

52

## I documenti XML sono alberi

- ◆ L'annidamento degli elementi definisce in modo esplicito una struttura ad albero:

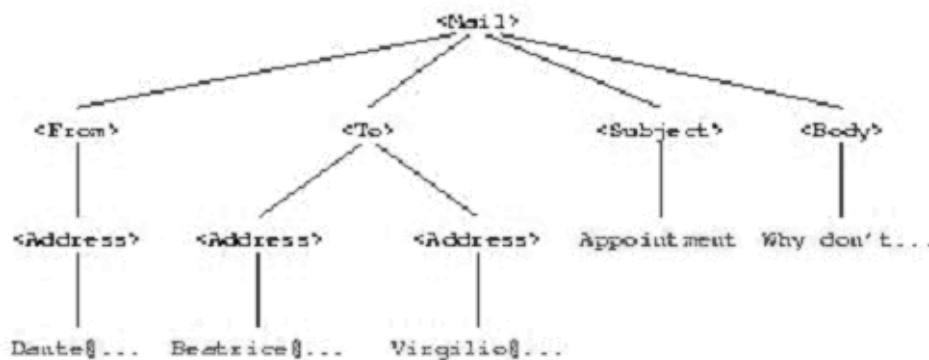
```

<Mail>
  <From> <Address> Dante@dsn.fi.it </Address> </From>
  <To> <Address> Beatrice@pitti.fi.it </Address>
      <Address> Virgilio@spqr.rm.it </Address>
  </To>
  <Subject> Appointment </Subject>
  <Body> Why don't we meet at disco.inferno at
          midnight. Tell also Caronte. Cheers,
          - D.A.
  </Body>
</Mail>

```

53

## I documenti XML sono alberi



54

## Interrogazioni a documenti XML: XQuery

- ◆ XQuery 1.0: An XML Query Language
- ◆ W3C Working Draft 12 November 2003
- ◆ This version:
  - <http://www.w3.org/TR/2003/WD-xquery-20031112/>
- ◆ Latest version:
  - <http://www.w3.org/TR/xquery/>



55

## Esempi di XML queries

- ◆ Molti esempi disponibili su:
- ◆ XML Query Use Cases
  - <http://www.w3.org/TR/xquery-use-cases/>
- ◆ Disponibile sul sito del corso (per consultazione)
- ◆ Uso paragonabile ad SQL, se si conosce lo schema (il DTD)
- ◆ Commentiamo qualche esempio ...

56