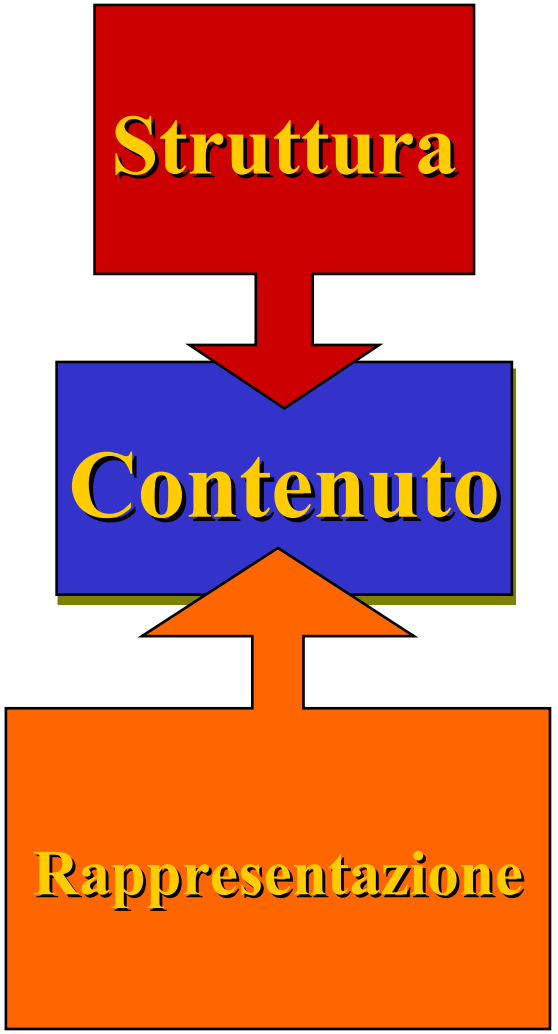
XML & DBMS

XML -- cenni

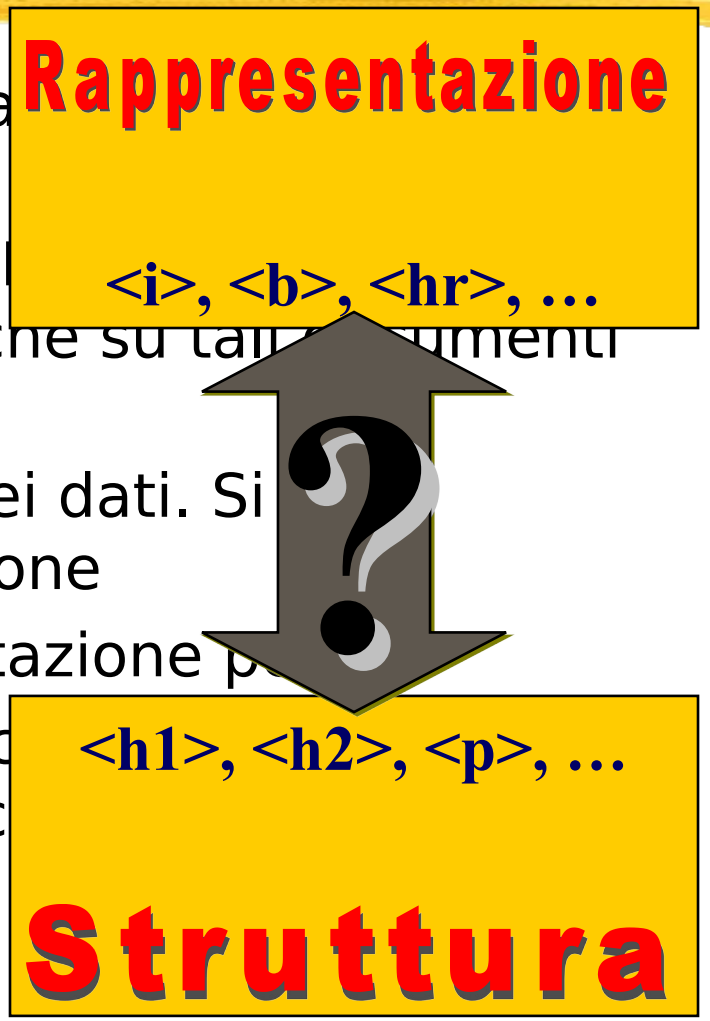
Struttura e Rappresentazione di un documento

- Un documento è composto di tre livelli.
- La rappresentazione può essere utile
  - per migliorare la leggibilità
  - per favorire la percezione della struttura
  - ...ma non per recuperare l'informazione
- Nel momento in cui occorre recuperare l'informazione le informazioni sulla struttura sono rilevanti
- Infatti attraverso la struttura dell'informazione è possibile ideare piani di recupero efficiente

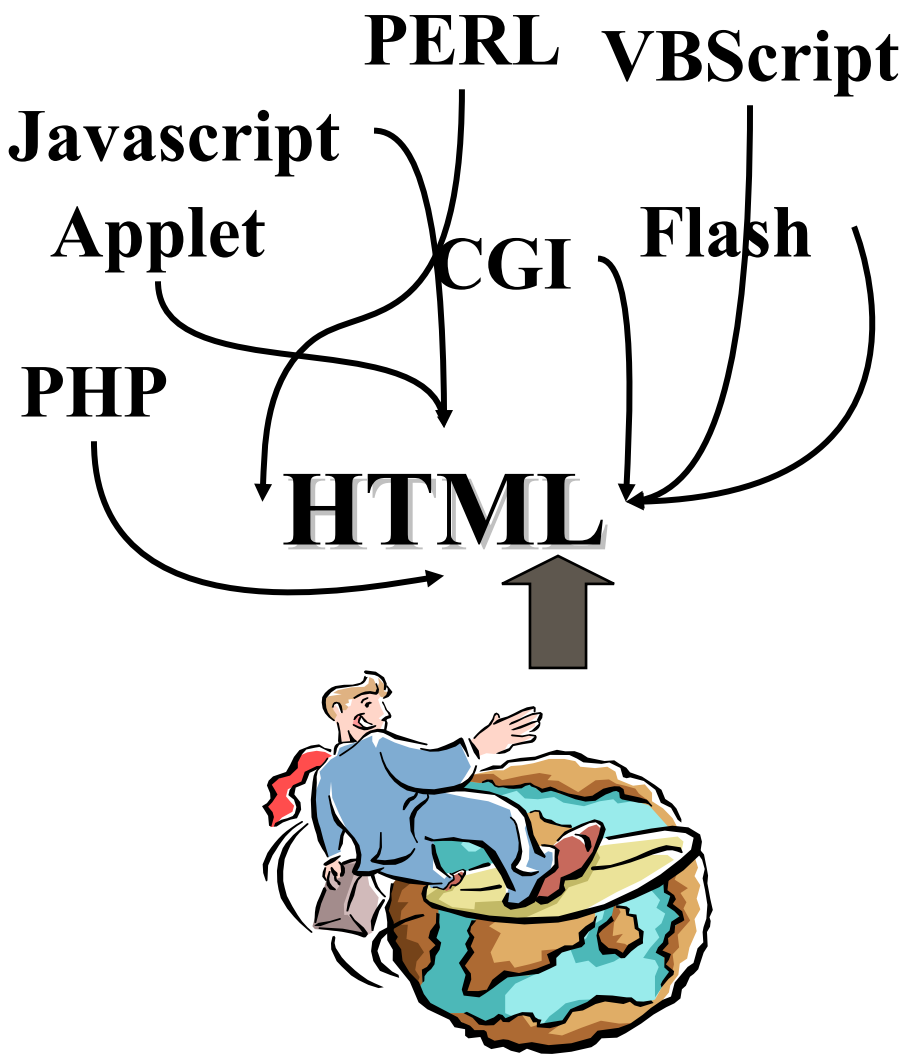


Crisi del Web

- La potenza di HTML rappresenta
  massima debolezza
- Se infatti realizzare pagine HTM
  facile riuscire a effettuare ricerche su tali documenti
  è molto inefficiente
- Manca il concetto di struttura dei dati. Si
  l'enfasi solo sulla rappresentazione
- Distinzione struttura/rappresentazione p
- Tale distinzione è la base della c
  linguaggi di markup di tipo desc
  HTML



Inadeguatezza di HTML - Nuove richieste



- Gli utenti richiedono pagine complesse,
- Le prestazioni hardware migliorano
- Javascript Shockvawe, Acrobat reader

- HTML diventa **assemblatore di tecnologie**

XML - Introduzione

- XML = Extensible Markup Language

  "XML is a language for creating markup languages that describe structured data."

  Mike Edwards, Microsoft

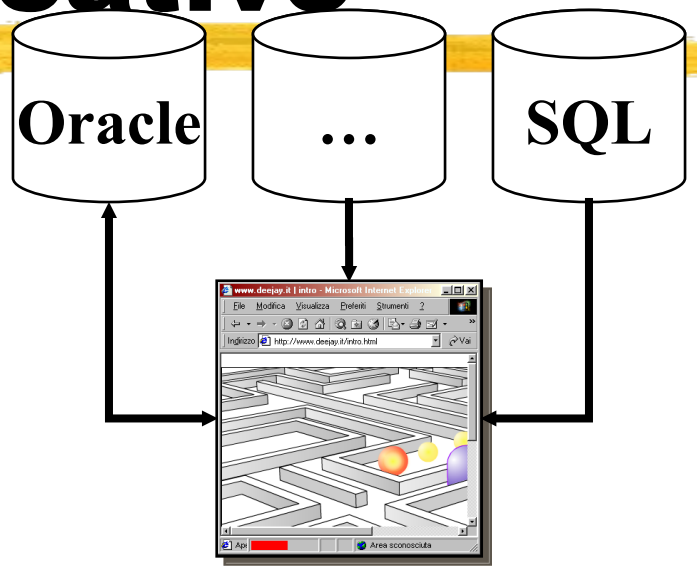
- Descrive i dati e non la loro rappresentazione
- Ha un formato aperto e leggibile visualmente simile all'HTML
- Elimina la necessità di Browser e tool di mezzo per aggiungere TAG speciali

XML - Introduzione

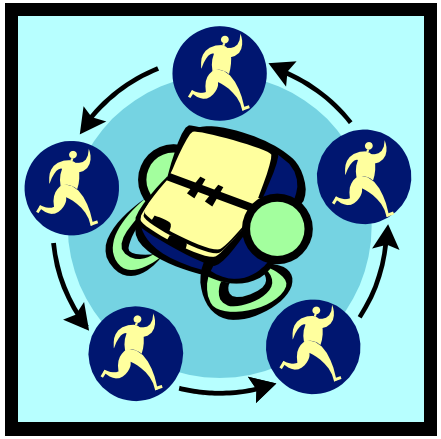
- eXtensible - non rigido come HTML
- Ottimo per la trasmissione di dati da server e browser
- Ottimo per la trasmissione da applicazione a applicazione, da macchina a macchina…
- E’ un **metalinguaggio usato per definire nuovi domini applicativi** o linguaggi specifici

XML - Aree Applicative

- Applicazioni che richiedono al Web Client di mediare tra due o più DB eterogenei

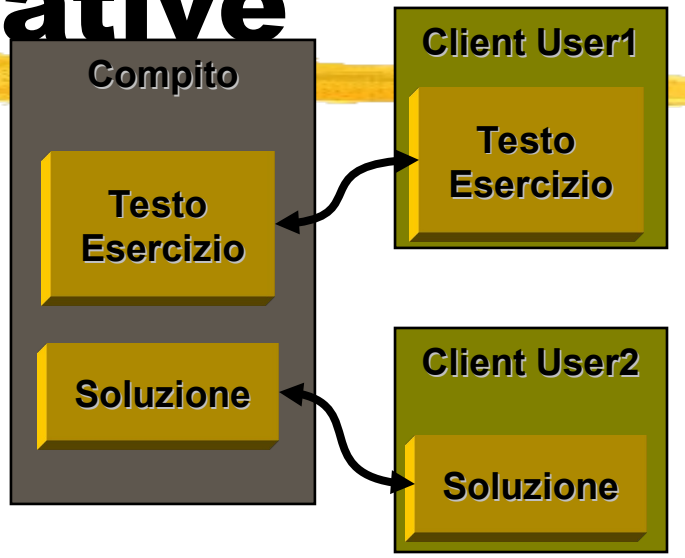


- Applicazioni che cercano di distribuire porzioni significanti di dati tra Client/Server

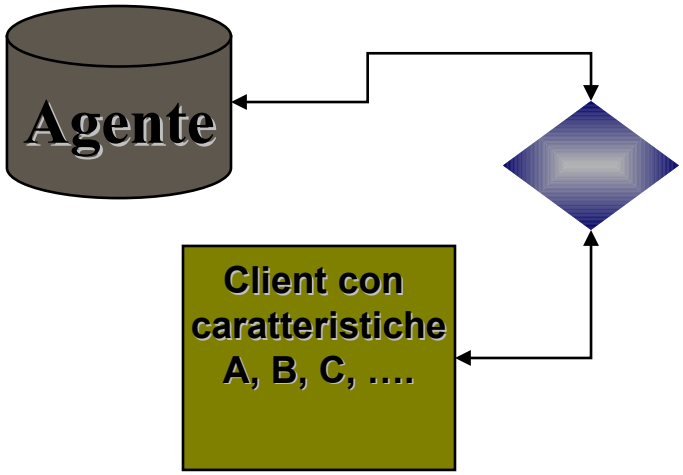


XML – Aree Applicative

- Applicazioni che richiedono al Web Client di mostrare più viste dello stesso dato a diversi utenti



- Applicazioni in cui Agenti Web Intelligenti decidono riguardo informazioni utili per un utente



XML - Vantaggi

- Attenzione esclusivamente al **contenuto**
- La fase di grafica può essere attuata successivamente o da un altra persona
- Si possono sviluppare **linguaggi ad HOC** specifici per certe comunità di utenti
- Matematici, chimici, una singola società, un ente…etc…

XML - Vantaggi

- La gestione semplice da SGML a XML rende le società che usano SGML per l'archiviazione dati proiettate alla pubblicazione di dati nel Web
- Strutturando le info bene, **i software di ricerca**, in IR, **possono essere precisi e veloci**
- scegliendo tra l'enorme quantità di materiale nel Web

XML e HTML

- XML non specifica né la semantica né un insieme di TAG
- XML **è un metalinguaggio** per la descrizione dei linguaggi di markup
- XML fornisce un modo per definire i tag ed i rapporti strutturali fra loro
- Poiché non vi è un insieme predefinito di TAG, non ci può essere alcuna semantica preesistente

XML – Standard collegati

- XML è una lingua “franca” che permette di descrivere la struttura di documenti
- Il W3C ha definito svariati standard per poter integrare la definizione di XML per consentire la:
  - Definizione metadata
  - Definizione link tra più documenti
  - Trasformazione/Rappresentazione documenti XML
  - Definizione di linguaggi di interrogazione
  - Manipolazione documenti XML “da programma”
  - Rappresentazione di informazione grafica

Definizione di metadata

- Come detto in precedenza, i TAG non definiscono la semantica dei dati che rappresentano
- Occorre quindi introdurre il concetto di **metadata** cioè “dati sui dati”

- Si parla di metadata a diversi livelli
  - a livello linguistico introducendo dizionari di nomi (Namespace)
  - a livello di tipo introducendo gli schema (Xschema)
  - a livello concettuale introducendo concetti e legami tra concetti (RDF -- Resource Description Framework)

Definizione link tra più documenti

- Lo standard XML permette di rappresentare solo i link interni al documento, ma non i link tra documenti diversi
- Inoltre, si vogliono poter rappresentare tutte le forme di link che abbiamo visto in precedenza

- Per questo motivo sono stati introdotti i seguenti standard
  - **XLink** che permette di definire link tra più documenti
  - **XPointer** che permette di indirizzare una parte specifica di un documento

Rappresentaz./Trasformaz. documenti XML

- Lo standard XML permette di strutturare documenti XML
- No rappresentazione
- Nasce **XSL**.
- XSL permette di
  - Identificare parti di documento da rappresentare (Xpath)
  - Trasformare le parti (attraverso il linguaggio XSLT)
  - Rappresentare (HTML, PDF, ...) il documento risultante (attraverso il linguaggio XSL-FO)

Definizione di linguaggi di interrogazione

- I documenti XML possono essere memorizzati in sorgenti di informazioni (DB).
- Ognuno può sottomettere alla sorgente richieste di documenti che soddisfano certe query.
- Esistono diversi linguaggi per poter interrogare sorgenti XML
- Attualmente non esiste uno standard W3C
- La proposta potrebbe essere QUILT

Manipolazione documenti XML “da programma”

- I documenti XML possono essere visti come canale di comunicazione tra applicazioni client-server
- Cioè permettono alle applicazioni di scambiarsi informazioni.
- Un’applicazione deve saper leggere e manipolare documenti XML
- Sono stati introdotti standard per manipolare documenti XML
- Gli standard principalmente utilizzati sono **DOM** e **SAX**

SVG – Scalable Vector Graphics

- Linguaggio per la descrizione di grafica 2D in XML 1.0
- SVG tiene conto tre tipi di oggetti grafici:
  - **vector graphic shapes** (path che consistono in linee rette e curve),
  - immagini
  - testo.
- Gli oggetti grafici possono essere raggruppati, trasformati
- Gli oggetti SVG possono essere interattivi e dinamici.
  - Lo vedremo in dettaglio dopo.

SMIL

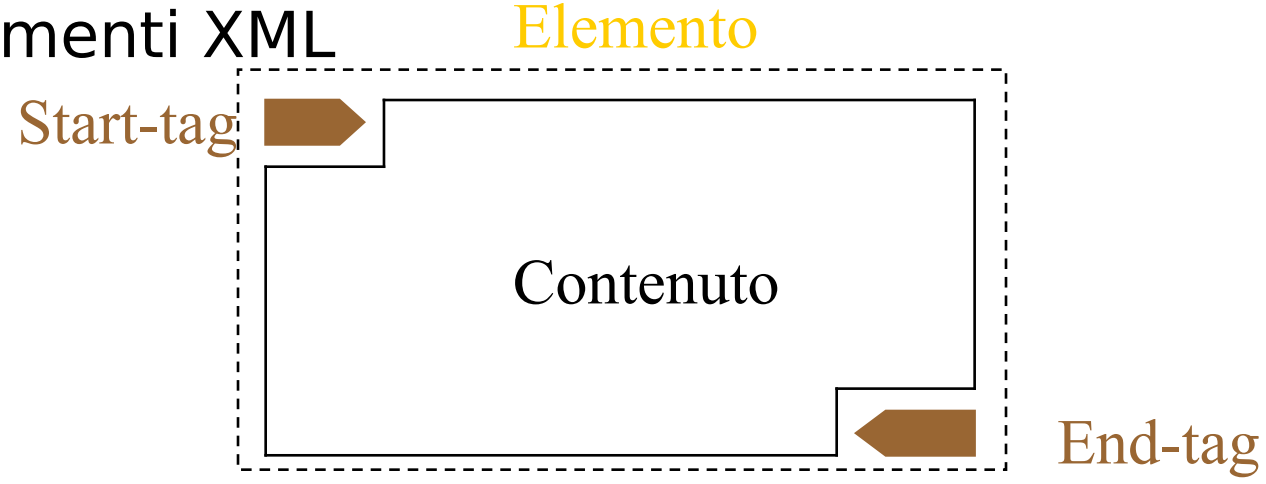
- Synchronized Multimedia Integration Language
- Permette la gestione di presentazioni nelle quali interagiscono audio e video, ed in generali elementi multimediali.
- Basato su XML è di fatto uno dei linguaggi definiti su di esso
  - Recommendation
  - Lo vedremo in dettaglio dopo.

XML e DTD

Sintassi ed esempi

Elemento

- Un **elemento** è un blocco elementare dei documenti XML



- Un elemento è una parte del documento delimitata da 2 TAG
  - es: <AUTHOR>Marco Mesiti</AUTHOR>

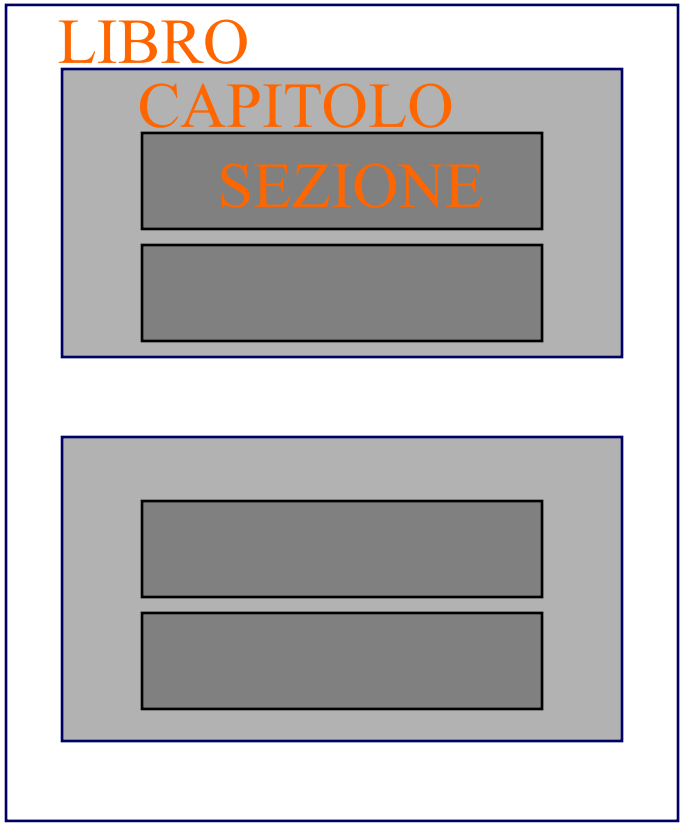
Tag

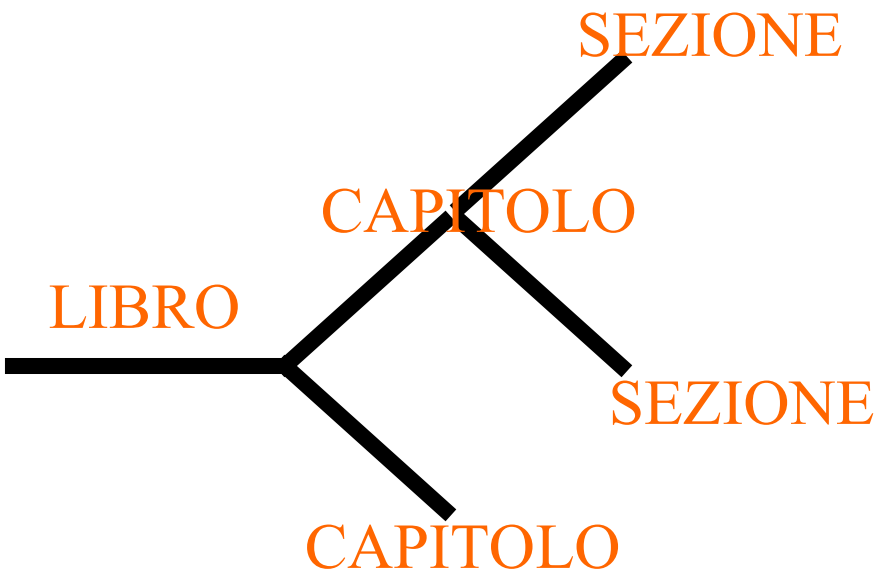
- I tag appaiono, normalmente, in coppia,
  - tag di apertura (start-tag)
  - tag di chiusura (end-tag)

```
<NOME_TAG>Marco Mesiti</NOME_TAG>
```

- Il tag rappresenta il nome dell'elemento

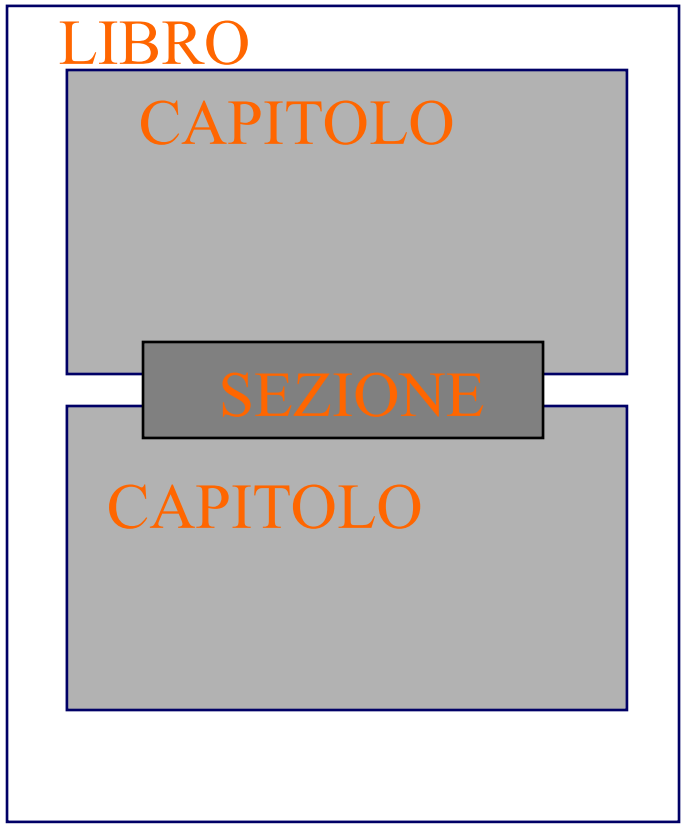
Struttura gerarchica degli elementi





Struttura gerarchica degli elementi

No!



Ogni elemento deve essere completamente incluso da un altro elemento

Contenuto di un Elemento

- altri elementi (*sub-elements*)

```
<address>
                    <street> 33, Terry Dr.</street>
                    <city> Morristown </city>
</address>
```

- testo (*data content*)

```
<street> 33, Terry Dr.</street>
```

- contenuto misto (*mixed content*)

```
<par>Today, <date>05-06-2000</date> Mr. <name>Bill
Gates<name> is in California to talk to ... </par>
```

Elemento documento (Document Element)

- E’ l’elemento più esterno in un documento XML
- Viene anche detto elemento radice
- Contiene tutti gli elementi del documento
  esempio:

```
<libro>
…
</libro>
```

- Deve sempre esistere

Elemento vuoto (Empty Elements)

- E’ un elemento senza contenuto
  - Non ha il tag finale
  - Ha una particolare rappresentazione del Tag iniziale

    esempio:

```
<foto/>
```

Attributi

- Un elemento può avere degli attributi
- Gli attributi possono essere pensati come aggettivi che descrivono gli elementi
- Ogni attributo ha
  - nome
  - valore
- Tutti gli attributi di un elemento devono essere distinti
- Gli attributi vengono inseriti come parte dello start-tag:

```
<AUTHOR  nome_attributo=”valore_attributo">
        Marco Mesiti </AUTHOR>
```

Esempio di elemento con attributi

```
<AUTHOR  laurea=”informatica">
        Marco Luca Mesiti
</AUTHOR>
<AUTHOR  laurea=”informatica">
        <name>Marco</name>
        <name>Luca</name>
        <lastname>Mesiti</lastname>
</AUTHOR>
<AUTHOR laurea=”informatica”
                name1 = ”Marco” name2 = ”Luca”
                lastname = ”Mesiti” />
```

Elementi Vs Attributi

Quando è meglio usare elementi o attributi per rappresentare un'informazione?

•Sicuramente elementi se si tratta di informazione strutturata

- Un elemento, quando:
  - Si richiede di recupe-rare i dati velocemente
  - E' visibile a tutti
  - E' rilevante per il signi-ficato del documento
- Un attributo, quando:
  - Esprime una scelta
  - E' utilizzato dal sistema
  - Non è rilevante per il significato del docu-mento

Un documento XML

- Un file XML è un semplice file di testo con tag XML al suo interno
- Esso ha una estensione **.xml**
  - nome_file.xml
- Un file XML contiene tre sezioni
  - Una dichiarazione che si tratta di un file XML
  - Una dichiarazione (opzionale) del tipo di documento e sul nome della DTD associata
  - Il contenuto del documento con tag XML

Un semplice esempio (1)

BOOKS.xml

```
<?xml version="1.0"?>
<BOOKLIST>
   <BOOK>
   <TITLE edition=”2000">The XML Companion</TITLE>
   <AUTHOR>Neil Bradley</AUTHOR>
   </BOOK>
   …
   <BOOK>
   <TITLE edition=”2000" type=”XML">
   Data on the Web
   </TITLE>
   <AUTHOR>Serge Abiteboul</AUTHOR>
   <AUTHOR>Peter Buneman</AUTHOR>
   <AUTHOR>Dan Suciu</AUTHOR>
   </BOOK>
</BOOKLIST>
```

Un semplice esempio (2)

LEASE.xml

```
<?xml version="1.0"?>
<Lease>
   <Leasee>ABC Industries </Leasee> agrees to lease the property
   at <Address>123, Main St., Chicago, IL</Address> from
   <Lessor>XYZ Properties</Lessor> for a term of not less than
   <LeaseTerm TimeUnit="Months">18</LeaseTerm> at the cost of
   the <Price Currency="USD" TimeUnit="Months">1000</Price>.
</Lease>
```

DTD: Document Type Definition

Sintassi Di Una DTD

DTD - Document Type Definition

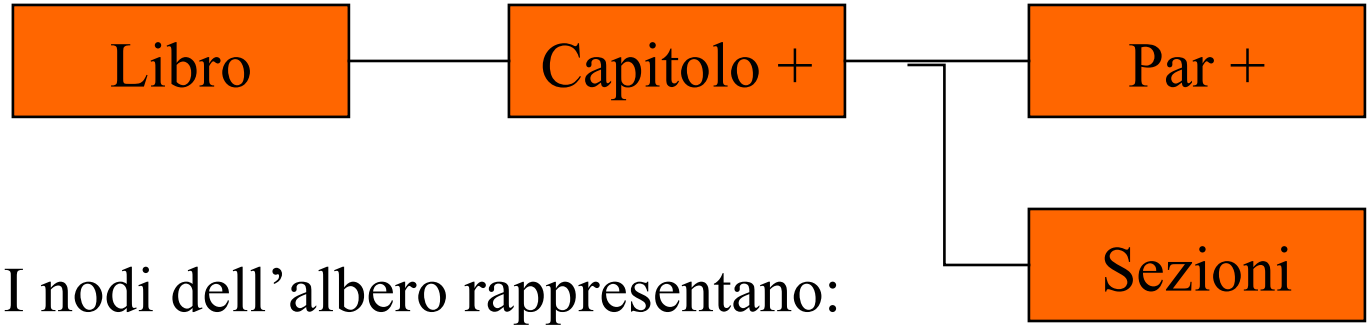
- E’ opzionale, ma è consigliabile la presenza
- E’ un insieme di regole per definire la struttura di un documento XML
- Tali regole:
  - stabiliscono gli elementi che possono essere usati
  - stabiliscono gli attributi da inserire negli elementi
  - impongono vincoli sulle relazioni tra gli elementi (fratelli, elemento-sottoelemento,...)
- Il DTD è un modo per fare un check sulla strutturazione corretta di un documento XML

Perché è importante usare i DTD?

- Sono utili per i programmatori. E’ la definizione del tipo di documento che andranno a processare
- Utili per definire fogli di stile
- Utile per creare documenti “corretti”. Il DTD può essere visto come un vincolo sull’informazione da inserire nel documento
- Utili per creare interfacce dinamiche per i documenti XML

DTD - rappresentazione ad albero

Un DTD può essere visto come un albero



- I nodi dell'albero rappresentano:
  - elementi
  - attributi
- I nomi di elementi e attributi possono essere seguiti da caratteri speciali (+,*,?)

- Gli archi dell'albero permettono di rappresentare la relazione elemento-sottoelemento, elemento-attributo

DTD - elementi

- Gli elementi possono essere contenitori o essere vuoti.
- Gli elementi contenitori possono contenere:
  - testo
  - altri elementi (sottoelementi)
  - un mix dei precedenti
- Nel caso di elementi che contengono sottoelementi, con il DTD si può specificare come i “sottoelementi occorrono”

DTD - occorrenza di un sottoelemento

- Un sottoelemento potrebbe essere obbligatorio
- Ad esempio un libro deve avere un titolo, quindi l'elemento libro deve avere un sottoelemento titolo

- Usando la grammatica della DTD

```
<!ELEMENT libro (… titolo …)>
```

DTD - occorrenza di un sottoelemento

- Un sottoelemento può essere opzionale
- Ad esempio, un libro può avere un sottotitolo, quindi l'elemento sottotitolo è opzionale per l'elemento libro

- Usando la grammatica della DTD

```
<!ELEMENT libro (… sottotitolo? …)>
```

DTD - occorrenza di un sottoelemento

- Un sottoelemento può essere ripetibile
- Ad esempio, un libro ha degli autori, almeno uno
- Quindi l'elemento libro può avere una lista di elementi autore, cmq almeno uno

- Usando la grammatica della DTD

```
<!ELEMENT libro (… autore+ …)>
```

DTD - occorrenza di un sottoelemento

- Un sottoelemento può essere ripetibile e opzionale
- Ad esempio, un libro può avere dei traduttori, oppure no
- Quindi l'elemento libro può avere una lista di elementi traduttore, ma anche nessuno

- Usando la grammatica della DTD

```
<!ELEMENT libro (… traduttore* …)>
```

DTD - Sequenze di sottoelementi

- Un libro presenta diverse informazioni: gli autori, il titolo, eventualmente i traduttori e così via. L'elemento libro può contenere quindi una sequenza di sottoelementi

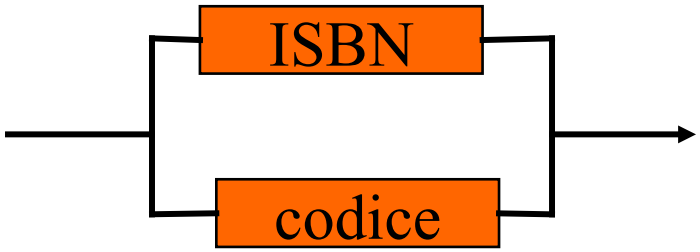
- Usando la grammatica della DTD

```
<!ELEMENT libro (titolo, autore* sottotitolo?, …)>
```

- Nota è rilevante l'ordine

DTD - Alternative di sottoelementi

- Un libro può avere un IBDN oppure un codice



- Usando la grammatica della DTD

```
<!ELEMENT libro (… ISBN | codice …)>
```

DTD - riassumento

- I costrutti visti possono essere combinati nel modo preferito, al fine di modellare l'informazione

```
<!ELEMENT A (B*, C, D?)>
<!ELEMENT A (B | C+)>
<!ELEMENT A (B, (C | D+)?, E*)>
```

DTD dell'esempio RICETTA

DTD ricettario.dtd

```
<!ELEMENT ricettario (ricetta)*>
<!ELEMENT ricetta (titolo, ingred+,
      passo+, note?)>
<!ATTLIST ricetta numero ID #REQUIRED>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT ingred (#PCDATA)>
<!ELEMENT passo (#PCDATA)>
<!ELEMENT note ANY>
<!ATTLIST passo numero ID #REQUIRED>
```

Dichiarare un elemento non vuoto (2/2)

```
<!ELEMENT nome (#PCDATA)>
```

- Il nome è semplicemente una stringa di caratteri,
- Eventuali entità presenti nella stringa vengono trasformate nei caratteri corrispondenti

Dichiarazione di un elemento vuoto

```
<!ELEMENT foto EMPTY>
```

- Ad esempio l'oggetto foto potrebbe pensarsi equivalente al tag HTML <IMG>
- Pertanto non ha un marcatore di chiusura né tantomeno un contenuto
- Possiede solo degli attributi

Dichiarazione di un elemento ANY

```
<!ELEMENT note ANY>
```

- Indica che il contenuto dell’elemento può essere arbitrario
- In altre parole, non viene fissato alcun vincolo sul contenuto dell’elemento

Dichiarazione elemento “mixed-content”

- Abbiamo detto che un elemento può avere un contenuto misto
- Questo può essere richiesto a livello di DTD
- La dichiarazione di un elemento a contenuto misto deve seguire le seguenti regole
  - viene dichiarato una alternativa di sottoelementi
  - il primo sottoelemento deve essere #PCDATA
  - L’alternativa di sottoelementi deve essere ripetibile

Dichiarazione elemento “mixed-content”

- Vediamo un esempio:

```
<!ELEMENT emph (#PCDATA|sub|super)*>
<!ELEMENT sub (#PCDATA)>
<!ELEMENT super (#PCDATA)>

<emph>H<sub>2</sub>O è l’acqua</emph>
```

Dichiarazione di attributi

```
<!ATTLIST foto src CDATA #REQUIRED desc CDATA
  #IMPLIED>
```

- src, nome del file bitmap necessario (#REQUIRED)
- desc, descrizione è opzionale (#IMPLIED)

```
<!ATTLIST telefono tipo (casa | ufficio | cellulare)
  'casa'>
```

- l’elemento telefono ha un attributo tipo,
- che indica se il numero è di casa, dell'ufficio o di un cellulare
- il valore di default è 'casa’

Un esempio

```
<!DOCTYPE Orders[
<!ELEMENT Orders(SalesOrder)+>
<!ELEMENT SalesOrder(Customer,OrderDate,Line*)>
<!ELEMENT Customer(CustName,Street,City,State,PostCode,tel*)>
<!ELEMENT CustName (#PCDATA)>
<!ELEMENT Street (#PCDATA)>    <!ELEMENT State (#PCDATA)>
<!ELEMENT PostCode (#PCDATA)>          <!ELEMENT tel (#PCDATA)>
<!ELEMENT OrderDate (#PCDATA)>         <!ELEMENT Line (Part,Quantity)>
<!ELEMENT Part(Description,Price)>     <!ELEMENT Quantity (#PCDATA)>
<!ELEMENT Description (#PCDATA)>       <!ELEMENT Price (#PCDATA)>
<!ATTLIST SalesOrder SONumber CDATA #REQUIRED>
<!ATTLIST Customer CustNumer CDATA #REQUIRED>
<!ATTLIST Line LineNumber CDATA #IMPLIED>
<!ATTLIST Part PartNumber CDATA #REQUIRED>
```

XML e Database

il problema

- Problema:
  - è possibile/necessario memorizzare documenti XML in un DBMS?
  - Quale tecnologia è necessaria a questo scopo?

- Risposta:
  - è certamente possibile memorizzare e gestire documenti XML in un DBMS
  - la tecnologia necessaria a questo scopo dipende dal perché vogliamo gestire documenti XML in un DBMS

Tipologie di documenti XML

- Due possibili usi per documenti XML:
  - *Data Centric*: i documenti possono rappresentare lo strumento con il quale dati tradizionali (es. relazionali) vengono trasferiti su Web
    - XML come veicolo per trasporto di dati
    - Esempio: ordini di vendita, scheduling di voli, menù
  - *Document Centric*: l'informazione è rappresentata dal documento in sé
    - XML come modello per la rappresentazione dei dati
    - Esempio: libri, documenti in genere

Documenti Data Centric

- Struttura regolare
- livello di dettaglio piuttosto fine
- contenuto omogeneo
- l’ordine con cui gli elementi allo stesso livello appaiono è ininfluente
- Utilizzati per “machine consumption”
- Esempi: ordini di vendita, scheduling di voli, menù,…

Esempio: ordini di vendita

```
<Orders>
   <SalesOrder SONumber="12345">
         <Customer CustNumber="543">
           <CustName>ABC Industries</CustName>
           ...
         </Customer>
         <OrderDate>981215</OrderDate>
         <Line LineNumber="1">
           <Part PartNumber="123">
             <Description>
                    Turkey wrench: Stainless steel, one piece...
             </Description>
             <Price>9.95</Price>
           </Part>
           <Quantity>10</Quantity>
         </Line>
         <Line LineNumber="2">
           ...
         </Line>
   </SaleOrder>
</Orders>
```

Documenti Document Centric

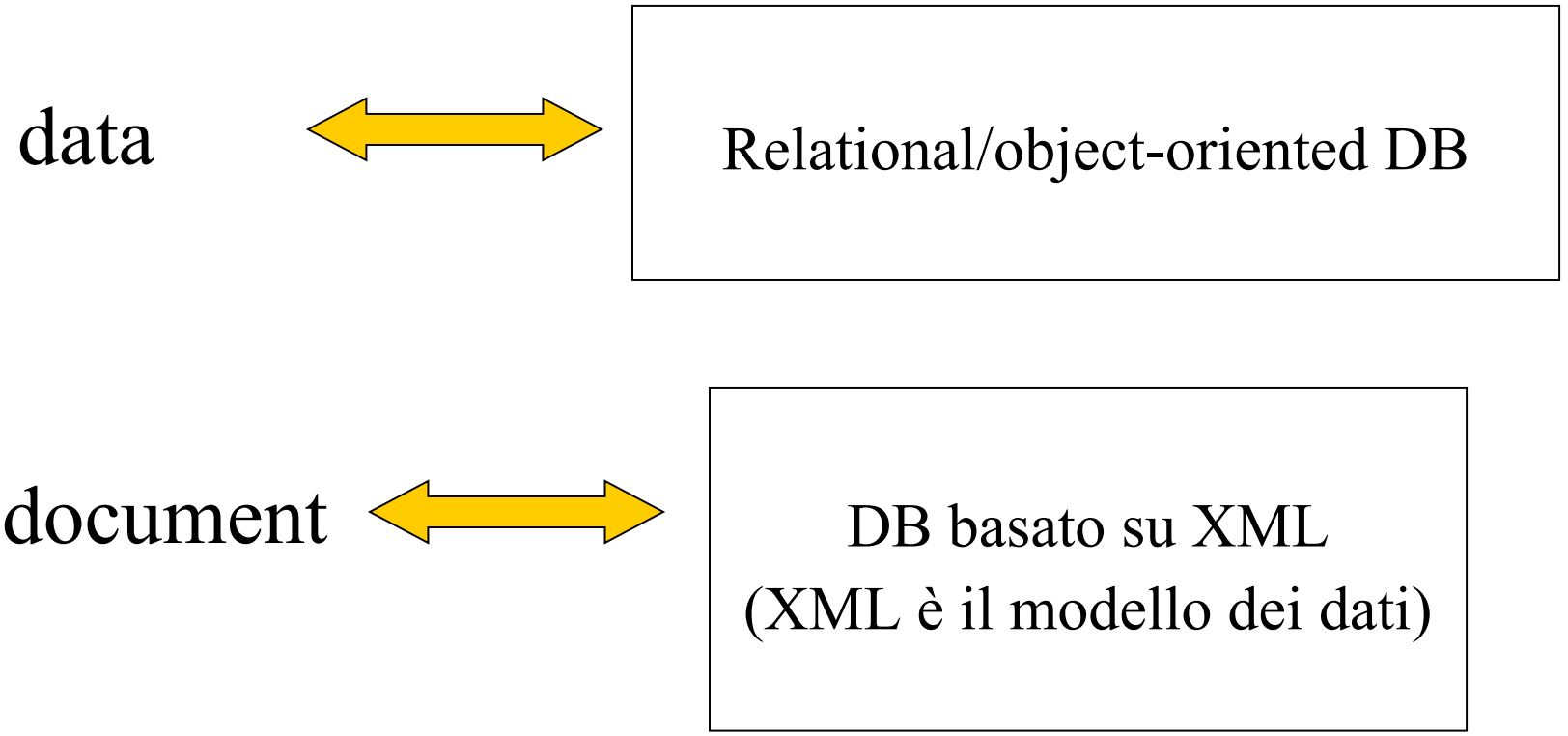
- Struttura irregolare
- Livello di dettaglio meno fine
- contenuto eterogeneo
- l’ordine degli elementi allo stesso livello è significativo
- in genere progettati per “human consumption”
- Esempi: libri, email, …

Product Description

```
<Product>
<Name>Turkey Wrench</Name>
<Developer>Full Fabrication Labs, Inc.</Developer>
<Summary>Like a monkey wrench, but not as big.</Summary>
<Description>
<Para>The Turkey wrench, which comes in both right- and left-handed
  versions ....</Para>
<Para>You can:</Para>
<List>
  <Item><Link URL="Order.htm">Order your turkey wrench</Link></Item>
  <Item><Link URL="Wrench.html">Read about wrenches</Link></Item>
  <Item><Link URL="catalog.zip">Download the catalog</Link></Item>
</List>
  ....
</Description>
</Product>
```

XML e DBMS

- Ciascuna tipologia di documenti richiede una particolare tecnologia per la sua gestione



XML e DBMS

- Due categorie di DBMS:
  - XML-Native DBMS:
    - comprendono un insieme di nuovi sistemi la cui architettura è stata progettata per supportare totalmente le funzionalità necessarie alla gestione di documenti XML
    - tecnologia non ancora matura
    - utili per Document Centric
    - Esempio:eXcelon
  - XML-Enabled DBMS:
    - comprendono tutti i DBMS che mantengono integra la propria architettura estendendola con funzionalità necessarie alla gestione di documenti XML
    - sono tipicamente Object-Relational (DB2, Oracle8i,...)
    - utili per Data Centric e parzialmente per Document Centric

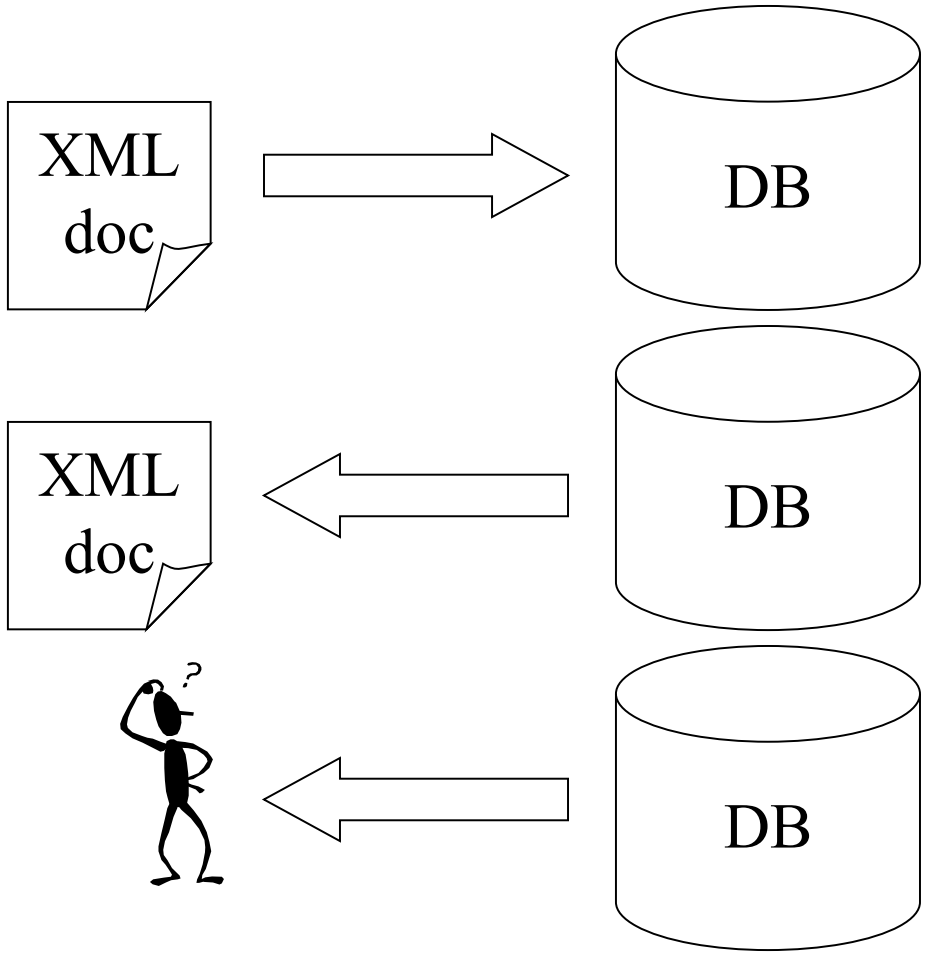
XML e DBMS

- Nel seguito.
  - Problematiche relative alla gestione di documenti Data Centric e Document Centric in XML-Enabled DBMS

XML-Enabled DBMS e documenti Data Centric

Problematiche per Data Centric

- Tre problematiche di base:
  - come rappresentare i dati contenuti nei documenti XML nel DBMS
  - come generare documenti XML partendo dai dati contenuti nel DBMS
  - come interrogare i dati estratti da documenti XML



Rappresentazione dati

- È necessario definire un mapping tra la struttura dei documenti XML e lo schema del DB
  - Per memorizzare i dati contenuti in un documenti XML in un DB, deve esistere una o più tabelle con lo schema richiesto dal mapping
- rappresentazione strutturata
- Vantaggi:
  - approccio piuttosto semplice
  - i dati sono facilmente interrogabili
- Svantaggi:
  - Scarsa flessibilità: la tabella deve essere conforme al documento
  - il documento di partenza non è più recuperabile

DBMS relazionale

- Un documento XML viene rappresentato come una singola tabella o un insieme di tabelle
- la struttura del documento XML è simile alla seguente:

```
<database>
<table>
<row>
<column1>...</column1>
<column1>...</column1>
...
</row>
...
</table>
...
</database>
```

- approccio tipico per DBMS relazionali, object-relational

Esempio

Documento XML

```
<clienti>
   <row>
     <numero> 7369 </numero>
     <nome> PAUL </nome>
     <cognome> SMITH </cognome>
   </row>
   <row>
     <numero> 7000 </numero>
     <nome> STEVE </nome>
     <cognome> ADAM </cognome>
   </row>
</clienti>
```

Tabella Clienti

Numero	Nome	Cognome
2000	MIKE	SCOTT
7369	PAUL	SMITH
7000	STEVE	ADAM

Esempio

Documento XML

```
<clienti>
  <row>
      <numero> 7369 </numero>
       <lista_clienti>
          <cliente>
             <nome> PAUL </nome>
             <cognome> SMITH </cognome>
          </cliente>
          <cliente>
            <nome> STEVE </nome>
            <cognome> ADAM </cognome>
          </cliente>
       </lista_clienti>
  </row>
</clienti>
```

Tabella Lista_Clienti

Numero
2000
7369

Tabella Clienti

Numero	Num_cliente	Nome	Cognome
2000	1	MIKE	SCOTT
7369	2	PAUL	SMITH
7369	3	STEVE	ADAM

Interrogazione dati

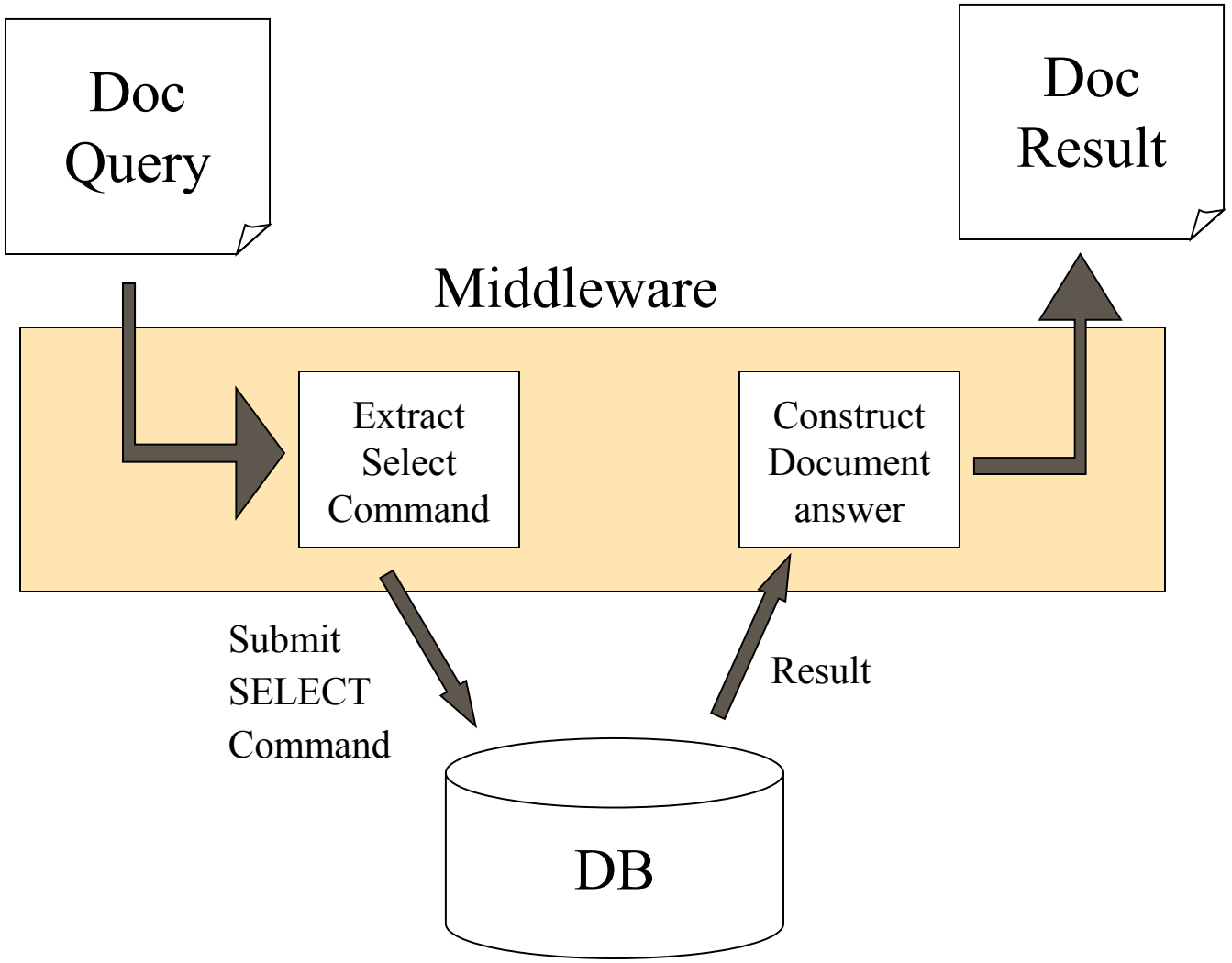
- Poiché i dati vengono rappresentati secondo il modello supportato dal DBMS (es. relazionale), è possibile utilizzare i linguaggi supportati dal DBMS per l'interrogazione dei dati memorizzati
- approccio template-based:
  - la query viene rappresentata nel documento XML
  - necessità di middleware

Flight Information

```
<?xml version="1.0">
<FlightInfo>
   <Intro>The following flights have available seats:</Intro>
   <SelectStmt>
        SELECT Airline, FltNumber, Depart, Arrive FROM Flights
   </SelectStmt>
   <Conclude>We hope one of these meets your needs</Conclude>
</FlightInfo>
```

```
<?xml version="1.0">
<FlightInfo>
   <Intro>The following flights have available seats:</Intro>
   <Flight>
        <Row>
           <Airline>ACME</Airline><FltNumber>123</FltNumber>
           <Depart>Dec 12, 1998
   13:43</Depart><Arrive>...<Arrive>
        <Row>
   </Flight>
   <Conclude>We hope one of these meets your needs</Conclude>
</FlightInfo>
```

Interrogazione dati



Generazione documenti XML

- Problema: fornire una rappresentazione XML ai dati recuperati tramite query dal DBMS
- si utilizza il mapping inverso rispetto a quello utilizzato per la memorizzazione
- operazione importante per attribuire un formato standard ai dati ritrovati, prima di inviarli sulla rete

Esempio

```
SELECT nome, cognome
FROM Clienti
WHERE Numero = “7369
```

Tabella Clienti

Numero	Nome	Cognome
2000	MIKE	SCOTT
7369	PAUL	SMITH
7000	STEVE	ADAM

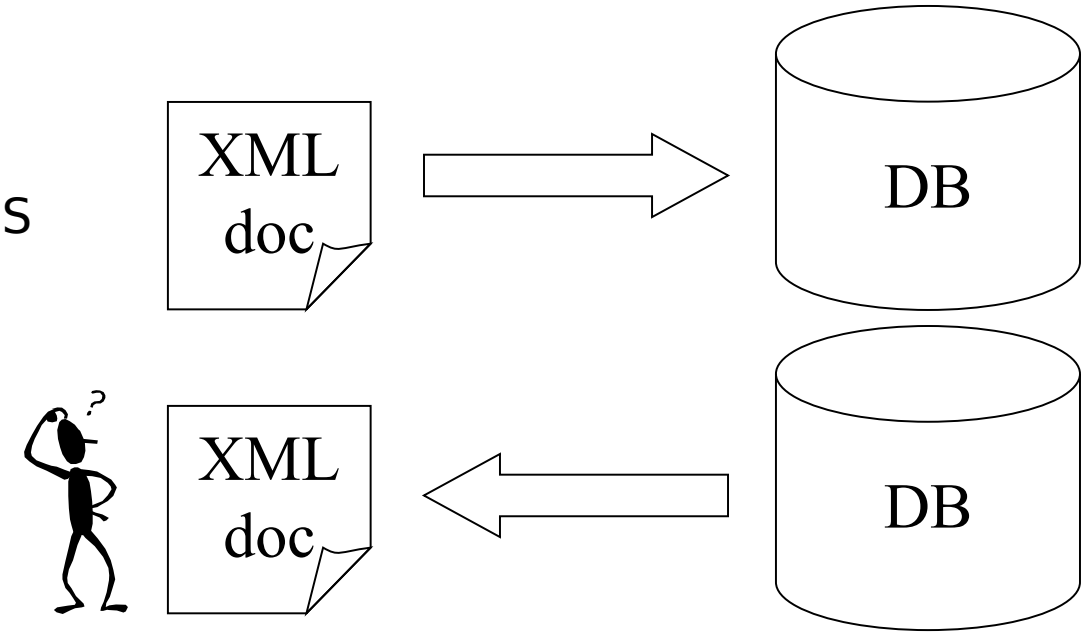
Documento XML

```
<clienti>
   <row>
      <nome> PAUL </nome>
      <cognome> SMITH </cognome>
   </row>
</clienti>
```

XML-Enabled DBMS e documenti Document Centric

Problematiche per Document Centric

- Due problematiche di base:
  - come rappresentare i documenti XML nel DBMS
  - come interrogare i documenti XML



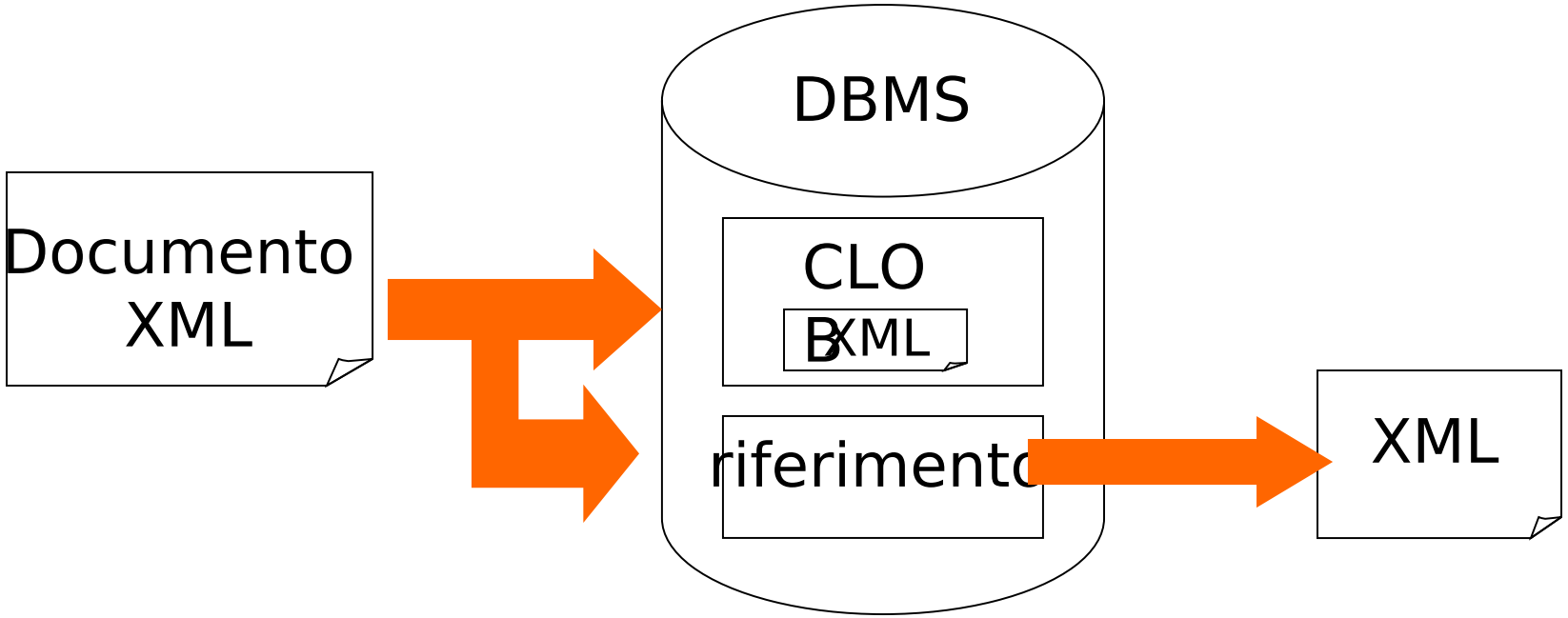
Rappresentazione

- Permette di mantenere integro il documento XML
- Due approcci:
  - rappresentazione non strutturata
    - documento come unico oggetto
  - rappresentazione ibrida
    - documento parzialmente rappresentato secondo la rappresentazione strutturata e parzialmente secondo la rappresentazione non strutturata

Rappresentazione non strutturata

- Il documento viene tipicamente mappato in un singolo campo di una tabella di tipo:
  - CLOB (Character Large Object): il documento è fisicamente contenuto nel campo della tabella
    - alcuni DBMS (IBM DB2) supportato tipi ad hoc: XMLVARCHAR
  - riferimento: il campo contiene il riferimento al documento, memorizzato altrove, sul file system
- Vantaggi:
  - flessibile
- Svantaggi:
  - i dati sono non strutturati
  - interrogazione più complessa
  - la tabella può contenere documenti eterogenei (diversi DTD)

Rappresentazione non strutturata



Esempio

Documento XML

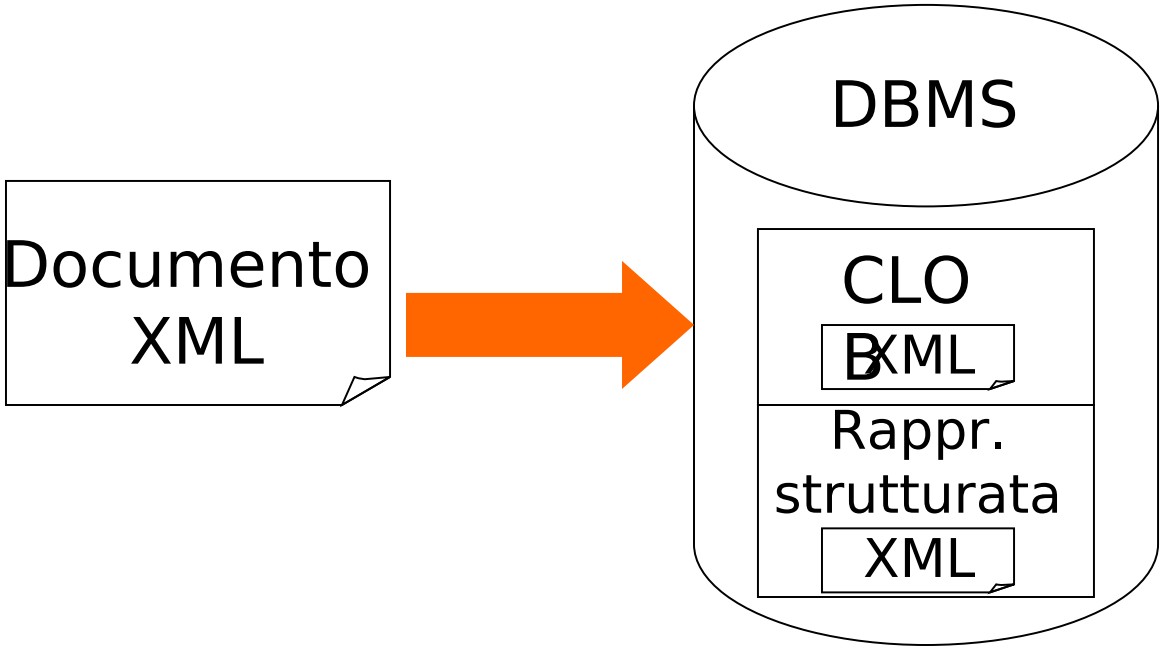
```
<clienti>
   <row>
     <numero> 7369 </numero>
     <nome> PAUL </nome>
     <cognome> SMITH </cognome>
   </row>
   <row>
     <numero> 7000 </numero>
     <nome> STEVE </nome>
     <cognome> ADAM </cognome>
   </row>
</clienti>
```

Tabella Clienti

Id	Documento_XML
10	<clienti> <row> <numero> 7369 </numero> <nome> PAUL </nome> <cognome> SMITH </cognome> </row> <row> <numero> 7000 </numero> <nome> STEVE </nome> <cognome> ADAM </cognome> </row> </clienti>

Rappresentazione ibrida

- Rappresentazione che combina rappresentazione strutturata e non strutturata

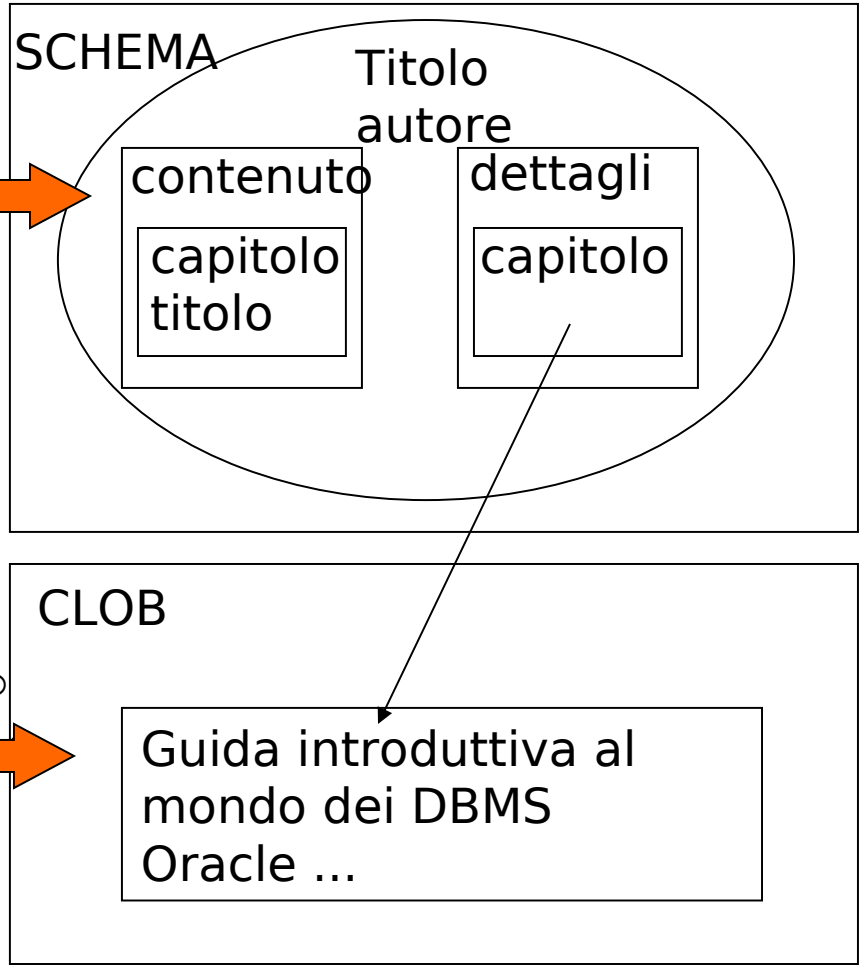


Esempio

Documento XML

```
<libro>
  <titolo> Oracle Guide </titolo>
  <autore> M. Abbey  </autore>
  <contenuto>
     <capitolo n='1'>
       <titolo> Introduzione
       </titolo>
        …
     </capitolo>
     …
  </contenuto>
  <dettagli>
     <capitolo n='1'>
       <sezione n='1'>
         Guida introduttiva al mondo
                  dei DBMS Oracle
       </sezione>
     </capitolo>
  </dettagli>
 …
</libro>
```

Tabella LIBRO



Interrogazione documenti

- Dal punto di vista del DBMS, un documento memorizzato in modo non strutturato non è che un documento di testo
- in genere i DBMS supportano strumenti per ritrovare i documenti in base al contenuto
- nel caso di documenti XML, mettono a disposizione operatori avanzati da utilizzare in statement SQL per recuperare documenti XML in base al contenuto

XML e Oracle 8i

- XML-enabled
- supporta rappresentazione strutturata, non strutturata in campi CLOB e BFILE, e ibrida
- interrogazione rappresentazione non strutturata tramite Intermedia Context
- generazione documenti XML a partire dal contenuto DB

XML e IBM DB2

- XML enabled
- supporta rappresentazione strutturata, non strutturata in campi ad hoc, e ibrida
- Nuovi tipi di dato:
  - XMLVARCHAR: documenti XML memorizzati come VARCHAR
  - XMLCLOB: documenti XML memorizzati come CLOB
  - XMLFILE: riferimento ad un documento XML, memorizzato su file system
- interrogazione rappresentazione non strutturata tramite:
  - operatori specifici, che permettono di navigare la struttura del documento
  - Text Extender, che supporta funzionalità aggiuntive di analisi del contenuto
- generazione documenti XML a partire dal contenuto DB

XML e SQL-Server 2000

- SQL Server 2000 è un XML-enabled DBMS
- supporta le seguenti funzionalità:
  - gestione documenti document centric:
    - tramite campi di tipo text (nessun supporto particolare)
  - gestione documenti data centric:
    - generazione documenti XML a partire dal contenuto della base di dati
    - inserimento di documenti data-centric
  - supporto per XDR (XML-Data Reduced) schema
    - viste in formato XML sullo schema di una base di dati
    - interrogazione di tali viste con XPath
  - accesso a SQL Server da HTTP

Gestione documenti document- centric

- T-SQL supporta comandi per:
  - generare un insieme di tuple da un documento XML
  - generare un documento XML come risultato di una query

Due parole su XPath

- Xpath permette di navigare la struttura ad albero di un documento XML, utilizzando una sintassi simile a quella utilizzata per navigare nel file system, estesa con l'utilizzo di condizioni di selezione (predicati)
- Restituisce I nodi finali del cammino specificato
- per ogni nodo lungo il percorso, è possibile specificare dei predicati
- per specificare il passaggio da un livello all'altro: /
  - /Customers/ContactName
- per identificare un attributo: @nome attributo
- per specificare una condizione: []
  - /Customer/Order[@OrderID="1"]
- per specificare il nodo padre: ..
- Per specificare il nodo corrente: .

Esempi

- Seleziona dalla radice (elemento più esterno), I clienti con attributo ClientID = ="ALFKI"
  - Customer[@CustomerID="ALFKI"]
- supponendo che Customer abbia come sottoelemento Order, seleziona tutti gli ordini in cui OrderID = 1
  - /Customer/Order[@OrderID="1"]
- Supponendo che Customer ammetta come sottoelemento ContactName, selezionare tutti I Customer che:
  - hanno almeno un ContactName
    - /Customer[ContacName]
  - non hanno ContactName
    - /Customer[not(ContacName)]

Esempi

- Selezionare gli ordini che si riferiscono al cliente identificato da "ALFKI"
  - /Customer/Order[../@CustomerID="ALFKI"]
  - /Customer[@CustomerID="ALFKI"]/Order
- selezionare l’ordine identificato da "Ord-10643”, relativo al cliente identificato da "ALFKI"
  - /Customer[@CustomerID="ALFKI"]/Order[@OrderID="Ord-10643"]
- selezionare I clienti che hanno effettuato almeno un ordine di un prodotto inquantità superiore a 5
  - /Customer[Order/OrderDetail[@Quantity>5]]

Esempi

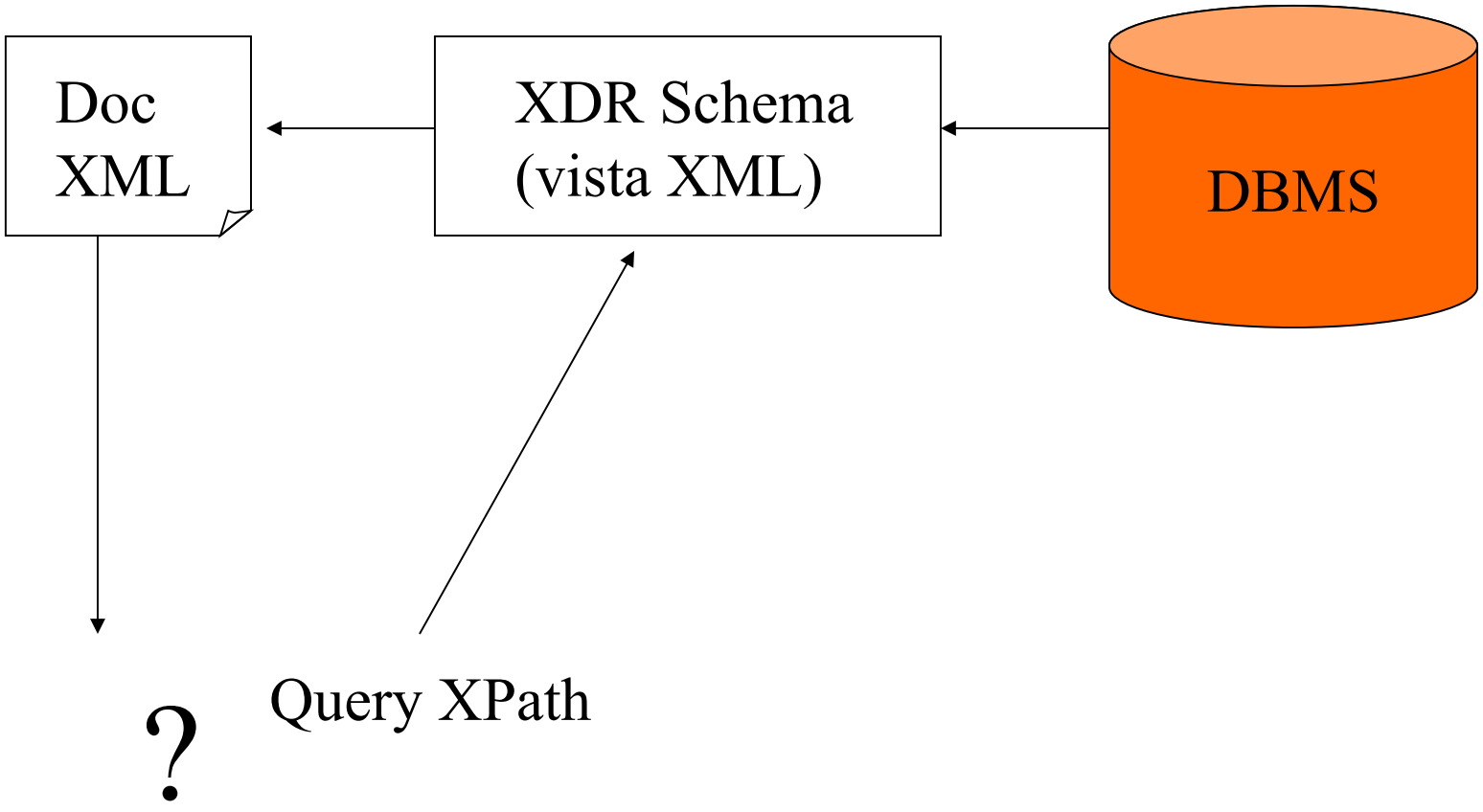
- Selezionare I vari prodotti ordinati, che complessivamente costano più di 2000
  - /child::OrderDetail[@UnitPrice * @Quantity = 2000]
- seleziona il cliente "ALFKI" e il cliente “ANATR”
  - /Customer[@CustomerID="ALFKI" or @CustomerID="ANATR"]

Esecuzione query XPath

- Xpath può essere utilizzato in SQL Server per interrogare documenti XML
- tali documenti devono essere creati definendo opportune viste XML sulla base di dati
  - tali viste possono essere specificate utilizzando gli schemi XML-Data Reduced (XDR)

- Quindi, in SQL Server per utilizzare Xpath, è necessario:
  - Creare viste XML del database tramite schemi XML-Data Reduced (XDR)
  - Interrogare tali viste con Xpath

Idea di base



Viste XML

- E’ possibile creare viste XML del contenuto del database utilizzando XML-Data Reduced Schema (XDR)
- Uno schema XDR descrive la struttura del documento XML che si vuole generare, quindi descrive la vista XML che si vuole eseguire sulla base di dati
- A differenza del DTD, mi permette di descrivere la struttura di un documento XML utilizzando la sintassi XML
- La struttura del documento deve quindi essere mappata sulla struttura del database

Viste XML

- Se si vogliono recuperare dati da una sola tabella e se I nomi delle tabelle e dei campi dai quali vengono estratti I dati coincide con il nome degli attributi e degli elementi con I quali si vuole costruire il documento XML
  - sintassi immediata
- se la condizione precedente non è soddisfatta, si utilizzano le “annotazioni”
  - informazioni che specificano come effettuare Il mapping tra elementi e attributi XML e dati nel database

Mapping di default

```
<?xml version="1.0" ?>
 <Schema        xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes"
xmlns:sql="urn:schemas-microsoft-com:xml-sql">
<ElementType name="Employees" >
<AttributeType name="EmployeeID" />
        <AttributeType name="FirstName" />
<AttributeType name="LastName" />

<attribute type="EmployeeID" />
<attribute type="FirstName" />
<attribute type="LastName" />
</ElementType>
</Schema>
```

Risultato

```
<ROOT>
<Employees EmployeeID ="1" FirstName="N
   LastName="Davolio"></Employee>
<Employees EmployeeID ="2" FirstName="A
   LastName="Fuller"></Employee>
</ROOT>
```

Tabella

nployees(EmployeeID,FirstName,LastName)

Mapping espliciti

- É possibile specificare come gli elementi e gli attributi del file XML vengono associati alle tabelle e ai campi delle tabelle
- si utilizzano le “annotazioni”
- per indicare a quale relazione si riferisce un certo elemento:
  - sql:relation
- per indicare a quale campo è associato un elemento o un attributo:
  - sql:field
- esiste un ampio numero di annotazioni, che permette di creare schemi XDR molto flessibili

Esempio 1

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes"
        xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <ElementType name="Customer" sql:relation="Customers" >
    <AttributeType name="CustomerID" />
    <AttributeType name="ContactName" />
    <AttributeType name="Phone" />

    <attribute type="CustomerID" />
    <attribute type="ContactName" />
    <attribute type="Phone" />
  </ElementType>
</Schema>
```

Risultato

```
<ROOT xmlns:sql="urn:schemas-
        microsoft-com:xml-sql">
  <Customer CustomerID="ALFKI"
        ContactName="Maria Ande
        Phone="030-0074321" />
</ROOT>
```

Tabella

ustomers(CustomerID,ContactName,Phone)

Come interrogare le viste XDR?

- É possibile utilizzare Xpath per interrogare le viste XDR
- la query Xpath deve essere rappresentata:
  - nell'URL
  - all'interno di un template

Utilizzo di Xpath query

- Direttamente nell'URL
  - http://paperino/myDB/schema/

    XDRSchema1.xml/Customer[@CustomerID="ALFKI"]?root=ROOT

- all'interno di un template

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:xpath-query mapping-schema=”nomefileXDRschema.xml">
    /Customer[@CustomerID="ALFKI"]
  </sql:xpath-query>
</ROOT>
```

Esempio

- XDR Schema:

```
<?xml version="1.0" ?>
    <Schema        xmlns="urn:schemas-microsoft-com:xml-data"  xmlns:dt="urn:schemas-
   microsoft-com:datatypes"         xmlns:sql="urn:schemas-microsoft-com:xml-sql">
   <ElementType name="Employees" >
   <AttributeType name="EmployeeID" />
          <AttributeType name="FirstName" />
   <AttributeType name="LastName" />

   <attribute type="EmployeeID" />
   <attribute type="FirstName" />
   <attribute type="LastName" />
   </ElementType>
</Schema>
```

- Xpath: /Employees[@EmployeeID=”1"]

Esempio (continua)

- Template:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:xpath-query mapping-schema="SampleSchema1.xml">
    /Employees[@EmployeeID=”1"]
  </sql:xpath-query>
</ROOT>
```

- Possibile risultato

```
<ROOT>

<Employees EmployeeID ="1" FirstName="Nancy ”
   LastName="Davolio"></Employee>

</ROOT>
```