ALEA Tech Reports

Lo standard XBRL (eXtensible Business Reporting Language) e la comunicazione finanziaria d'impresa

Appendice 2

XML: GUIDA INTRODUTTIVA

Walter Aste e Davide Panizzolo

Tech Report Nr. 20
Maggio 2004

Alea - Centro di ricerca sui rischi finanziari
Dipartimento di informatica e studi aziendali
Università di Trento - Via Inama 5 - 38100 - Trento
Aleaweb **http://www.aleaweb.org/**

1.1 XML (EXTENSIBLE MARKUP LANGUAGE)

In questo capitolo viene affrontata la tematica inerente a XML, un innovativo linguaggio di contrassegno che negli ultimi anni sta ottenendo un notevole successo nell'ambiente internet.

L'attenzione dello studio viene quindi spostata verso argomenti attinenti all'informatica, lasciando sullo sfondo i concetti economico-gestionali descritti precedentemente. L'obiettivo di questo capitolo è di introdurre un'analisi delle funzionalità principali di XML, strumento che viene usato sul web per descrivere dati strutturati. Si procede dunque inizialmente con una presentazione generale dell'argomento, in modo da formalizzare alcuni termini informatici ed individuare l'ambito operativo in cui sono implementate le soluzioni. Successivamente si propone una rassegna dei principali linguaggi di contrassegno antecedenti a XML, così da evidenziarne le analogie ed i limiti che li contraddistinguono.

La seconda parte dell'analisi di XML comprende una descrizione della struttura dei file creati. Per facilitare la lettura del testo, contenente concetti tecnici e piuttosto astratti, sarà fornito un esempio che verrà ripreso nel corso di tutta l'esposizione. Tale supporto sarà particolarmente utile nel momento in cui saranno introdotti i linguaggi di definizione della struttura dei documenti (Schemi e DTD). L'esempio di documento XML è stato realizzato utilizzando il software XmlSpy 2004, un programma creato appositamente per facilitare la scrittura di codice XML. Le sue funzionalità coprono le principali aree di interesse che il progettista XML deve tenere in considerazione nello sviluppo dei file. Il software è scaricabile al sito www.xmlspy.com, dove è possibile richiedere una licenza gratuita e della durata di 30 giorni, oppure richiedere una licenza permanente a pagamento. Tuttavia questa applicazione non è strettamente indispensabile per la creazione di un documento XML, dato che l'unico strumento richiesto è un semplice editor di testo (ad esempio Notepad).

La parte finale del capitolo affronta la descrizione di alcuni linguaggi informatici, quali XSLT e XPATH, usati congiuntamente per ottenere la formattazione grafica dei documenti XML.

Nel corso dello studio si è cercato di fornire una descrizione esaustiva delle principali caratteristiche che contraddistinguono XML ed i linguaggi paralleli. Tuttavia l'analisi proposta non ha lo scopo di illustrare in modo specifico e dettagliato i comandi e la sintassi degli elementi creati. Quindi per un approfondimento di questi concetti si rinvia la lettura ad appositi manuali informatici che sono facilmente reperibili in qualsiasi libreria o biblioteca.

1.1.1 Introduzione all'XML

Il termine XML è l'acronimo in lingua inglese di **eXtensible Markup Language**, ossia in altri termini identifica un linguaggio di contrassegno che ha la caratteristica principale di essere estensibile. L'XML è nato nel febbraio del 1998 come raccomandazione del W3C, un'organizzazione il cui compito consiste nella costruzione e nell'aggiornamento dei principali standard adottati in internet.

Un **linguaggio di contrassegno** o di marcatura si può generalmente definire come un sistema attraverso il quale è possibile scomporre e ordinare il contenuto di un documento per mezzo di strumenti chiamati **tag**. Quindi è possibile strutturare i dati inclusi in un determinato file, organizzandoli in strutture gerarchiche che ne permettono una manipolazione più efficace. E' da sottolineare che il termine "linguaggio di markup" non è sinonimo di "linguaggio di programmazione", in quanto non prevede funzioni quali i cicli iterativi e le condizioni logiche nelle sue proprietà. Esempi di linguaggi di programmazione sono Cobol, Delphi, Python, ma non XML.

XML è stato presentato dal W3C alla comunità informatica come la principale novità introdotta nell'ambiente internet dai tempi della comparsa del primo Html (HyperText Markup Language). In quest'ottica numerose imprese e organizzazioni appartenenti a settori eterogenei dell'economia e con finalità molteplici, stanno investendo negli ultimi anni considerevoli energie, sforzi e ingenti somme di denaro nel suo sviluppo e per promuoverne l'adozione a livello mondiale. Si nutrono molte aspettative nei confronti dell'XML in quanto esso

sembrerebbe essere una soluzione ottimale per utilizzare al meglio le potenzialità offerte dal contrassegno. XML rappresenta quello che la lingua inglese è stata per le attuali lingue internazionali, ossia una lingua franca che è utilizzabile sul web e a cui ogni produttore di software ed utente dovrebbe prestare attenzione nel momento in cui intende creare un documento informatico o trasferire informazioni ad altri soggetti.

Una delle principali ragioni del crescente interesse nei confronti di questa tecnologia si riferisce al fatto che è relativamente semplice da apprendere e da applicare a casi concreti. Pur risultando più essenziale in rapporto ad altri linguaggi di markup, XML offre funzionalità estremamente eterogenee che consentono di strutturare documenti della complessità desiderata. In un futuro ormai prossimo XML ed i suoi derivati mirano a sostituire, o comunque ad integrare, i linguaggi di markup proprietari attualmente esistenti (ad esempio ASP di Microsoft). Si osserva che le ultime versioni rilasciate dei principali software integrano caratteristiche XML per garantire la portabilità delle informazioni e quindi per riuscire ad ottenere una maggiore integrazione con le altre applicazioni. XML, infatti, è completamente indipendente dalla piattaforma hardware sul quale viene eseguito ed inoltre non è limitato dall'applicazione che lo gestisce. L'obiettivo è quello di creare programmi che siano in grado di interpretare e di gestire correttamente informazioni molteplici, provenienti da sistemi che storicamente non sono compatibili, a causa sia di questioni hardware che software (es. IBM e Apple). La comunità informatica crede fortemente che XML sia il formato adatto per diventare lo standard universale relativo alla trasmissione di informazioni sia fra sistemi omogenei che fra sistemi eterogenei.

La finalità principale che ha portato ad usare il nuovo linguaggio nel mondo del commercio è stata quella di realizzare il così detto "ufficio senza carta", obiettivo che è rimasto un'utopia per numerosi decenni, ma che potrebbe divenire realtà combinando la nuova tecnologia con i vantaggi offerti da internet. I benefici derivano dal fatto che si rendono possibili ricerche migliori nei database aziendali e si ottimizza la comunicazione all'interno e all'esterno della impresa. Inoltre si ottiene il vantaggio di consentire agli attori aziendali di scambiarsi informazioni in modo più chiaro e completo rispetto alle modalità precedenti.

E' da ricordare che se un'applicazione software è implementata in modo da prevedere l'utilizzo di documenti scritti in XML, può inviare o ricevere qualunque tipo di informazione nei confronti di qualsiasi altra applicazione che utilizzi a sua volta XML. Fino a poco tempo fa il grosso limite nella trasmissione dei dati era costituito dall'impossibilità di far dialogare applicazioni diverse, problema dovuto al fatto che i flussi informativi viaggiavano attraverso dei sistemi che molto spesso erano proprietari. Usando XML nel momento dell'invio dei dati si spedisce anche il "dizionario" relativo al documento e dunque una qualsiasi applicazione in grado di gestire XML è in grado di leggere il file, indipendentemente dal markup utilizzato.

Un'ulteriore caratteristica dell'XML è quella di basarsi essenzialmente sul testo. Quindi analizzandone il contenuto è possibile osservare i dati grezzi per comprenderne appieno e velocemente il significato. XML consente di comporre il documento in modo che possa essere recepito esattamente sia dal lato macchina che dal lato umano e garantisce meccanismi che consentono di condividere la conoscenza delle strutture dei dati con altri soggetti.

Nella definizione dell'acronimo XML è opportuno specificare con più attenzione il termine **eXtensible**, cioè l'**estensibilità** che gli è propria, caratteristica che nel contesto di riferimento assume un ruolo molto importante. XML non è propriamente un linguaggio di contrassegno, ma più precisamente è un **metalinguaggio di markup**, ossia uno strumento adatto per creare linguaggi di marcatura personalizzati. XML è dunque una tecnologia che permette di realizzare un qualsiasi linguaggio di markup ad hoc, con l'unico vincolo che questo sia composto da semplice testo. Questa è la funzionalità che ha portato XML ad essere usato in ogni ambito di applicazione e che probabilmente costituirà la chiave del suo successo negli anni futuri. L'XML offre l'opportunità a organizzazioni ed a gruppi di persone interessate di creare un proprio linguaggio di markup, destinato a rispondere alle particolari esigenze richieste dal tipo di informazione trattato. La specifica del W3C non sarebbe così utile e interessante se ogni persona che avesse l'intenzione di usare il linguaggio sviluppasse una

grammatica propria, poiché il numero delle varianti esistenti crescerebbe a dismisura, impoverendo così il ruolo di XML come standard e come supporto alla comunicazione. In molti ambiti applicativi gruppi di esperti hanno già implementato delle soluzioni particolari; ad esempio nel campo della finanza e dello scambio di informazioni commerciali è stato progettato l'XBRL (eXtensible Business Reporting Language), per il campo chimico e scientifico si è costruito il CML (Chemical Markup Language), e così via. Per una descrizione più approfondita della tematica inerente a XBRL si rinvia al quarto capitolo.

La forza di XML è quella di consentire all'utente di organizzare le informazioni a seconda delle proprie esigenze usando tag appropriati e consentendo così di inviarle liberamente a chiunque. Ciò è consentito dal fatto che XML non è uno standard proprietario, ma bensì è uno standard aperto. Questo significa che è possibile utilizzarlo liberamente senza pagare nessuna royalty o diritto d'autore. Inoltre XML non prevede alcun tipo di restrizione, quindi non esistono brevetti, marchi di fabbrica o copyright.

Non essendo predisposto un insieme di tag predefiniti sui quali basare la costruzione dei documenti, risulta che non esiste una semantica pre-impostata. Con il termine **semantica** si identifica il significato che le singole parole assumono all'interno di un documento. Quindi la semantica che accompagna un documento creato con XML o con un suo derivato è definita dalle applicazioni che lo elaborano. L'unica cosa che si descrive nel file è la struttura di definizione e di annidamento dei tag, dunque le relazioni gerarchiche con le quali gli elementi sono legati l'uno all'altro. In principio XML è stato sviluppato appositamente per consentire un uso appropriato di documenti strutturati sul web.

XML è una specifica creata e gestita dal W3C (World Wide Web Consortium) e questo fa sì che il prodotto sia gratuito e ben strutturato fin dal progetto del documento. Il problema che accompagna molti file creati in passato è quello di non avere una documentazione esaustiva che li descrive. In questi casi un soggetto incontra molte difficoltà se intende per una qualsiasi ragione modificare o semplicemente aggiornare un file che non ha progettato o realizzato di persona. Infatti spesso non si conosce a priori il significato ed il ruolo di determinati valori contenuti nel documento e quindi risulta difficile capire cosa andare a cambiare. XML risolve questa grave carenza strutturale visto che si presenta in un formato costituito esclusivamente da testo e che quindi offre notevoli garanzie dal punto di vista della leggibilità e della documentazione allegata. E' inoltre una soluzione che offre una combinazione ottimale di flessibilità e di semplicità, la quale consente di effettuare modifiche ai file senza particolari problemi. Da questo punto di vista XML comporta la possibilità di creare raccolte organizzate di informazioni, gestite in modo trasparente ed efficiente. Quindi si offre l'opportunità di ridurre le inefficienze informative che caratterizzavano le situazioni precedenti alla adozione del nuovo standard.

Panoramica sui principali linguaggi di markup

La specifica XML è stata sviluppata e portata avanti dal W3C, lo stesso consorzio di aziende che ha partecipato alla standardizzazione di HTML. In particolare XML nasce dall'esigenza di migliorare i linguaggi di markup utilizzati in passato che, pur essendo potenti e sotto certi aspetti piuttosto semplici da usare, comportavano spesso molte limitazioni e difficoltà di adattamento ai contesti operativi. Per meglio apprezzare e capire le nuove opportunità ed i vantaggi introdotti dall'adozione di XML si procede ad una descrizione dei due principali linguaggi di markup utilizzati precedentemente, cioè HTML e SGML.

- **SGML**: SGML è l'acronimo di **Standard Generalized Markup Language**, linguaggio di contrassegno molto potente usato prevalentemente in contesti in cui occorre trattare notevoli quantitativi di dati (es. ambito aziendale, militare o statistico). L'SGML è stato sviluppato dall'IBM nel 1969 e nel 1986 ha ottenuto la certificazione ISO. E' un linguaggio che, analogamente a XML, consente di modellare linguaggi di programmazione personalizzati, in modo da formattare coerentemente la struttura del

testo e consentire dunque una rapida selezione delle informazioni. Inoltre i documenti SGML sono facilmente trasferibili e riprogrammabili in altri formati a seconda delle necessità. Il contrassegno usato si basa sul contenuto delle informazioni e non sul loro layout e ciò rende agevole il loro riadattamento in contesti diversi.

La difficoltà di implementazione pratica di SGML ha ridotto drasticamente i suoi utilizzi reali. Le complicazioni si individuano soprattutto nella gestione e nell'elaborazione dei documenti che necessitano di competenze specialistiche e di ingenti investimenti per l'applicazione all'ambiente operativo. Attualmente l'SGML viene usato solamente da ristretti gruppi di utenti che si localizzano in grandi aziende, nelle università e in dipartimenti governativi situati prevalentemente negli Stati Uniti.

XML deriva direttamente dall'SGML, ma fa propria una sintassi che risulta più concisa e più semplificata. In particolare nel nuovo linguaggio sono eliminate gran parte delle regole di SGML che nel tempo si sono dimostrare poco utilizzate o comunque poco interessanti alla massa dei programmatori. XML si configura come un tentativo di avvicinare le potenzialità straordinarie di SGML alle applicazioni in contesti reali e si propone di rendere più economico il suo utilizzo. Dal momento che XML è un sotto-insieme proprio di SGML, molte applicazioni realizzate appositamente per SGML possono essere tradotte rapidamente in XML. Si parla comunque di un intervallo di tempo che copre mesi, se non anni di lavoro, visto che comunque l'implementazione completa di un programma in SGML richiede normalmente uno staff apposito di personale e un tempo molto lungo di analisi e di sviluppo.

- **HTML:** analogamente a XML, anche l'HTML è una versione molto semplificata dell'SGML che è stata realizzata con la particolare finalità di descrivere documenti ipertestuali. Il linguaggio HTML è stato introdotto nel 1994 e fin dalle prime applicazioni si è capito che esso rappresentava una svolta epocale per i tradizionali sistemi di utilizzo del web e che il suo utilizzo avrebbe costituito la chiave per uno sviluppo globale della rete. Attraverso l'uso di pochi tag (dal momento che il set a disposizione era veramente ridotto), chiunque poteva progettare un sito web apprezzabile. Lo scopo iniziale del linguaggio era quello di consentire a programmatori professionisti e dilettanti di creare pagine web funzionanti. HTML è stato usato nel tempo per rappresentare qualsiasi tipo di dato e qualsiasi tipo di cosa, dai cataloghi on-line ai testi accademici.

  Tuttavia il linguaggio presenta numerose limitazioni e non è assolutamente adatto a descrivere il contenuto dei dati, ma si limita semplicemente a formattare il testo per consentirne una visualizzazione ottimale a video. La struttura dei documenti HTML non consente di studiare analiticamente il contenuto reale della pagina web, questo perché la maggior parte dei tag è destinato a influenzare solo il layout grafico della pagina e non a descrivere il significato dei dati contenuti in essa. Quindi le informazioni incluse in uno stesso tag assumono significati completamente diversi a seconda del contesto in cui vengono usati e non hanno un valore oggettivo e univoco per ogni soggetto esterno che le legge. Ciò nonostante in breve tempo le pagine internet sviluppate con questa tecnologia sono aumentate a dismisura.

  HTML non è estensibile e quindi ha una propria tassonomia, che però è risultata presto inefficiente. I limiti di un linguaggio basato esclusivamente sulla formattazione grafica del testo sono risultati subito palesi. Fattori quali la mancanza di uno sviluppo di standard comuni tra i browser (Microsoft Explorer e Netscape), oltre che l'immaturità degli strumenti di progettazione della rete esistenti in quel periodo e l'incapacità del W3C di rispondere tempestivamente alle esigenze richieste dal mercato, hanno aumentato l'insoddisfazione dei progettisti.

  L'XML è nato con lo scopo di sostituire l'HTML e di offrire così un'architettura che sia allo stesso tempo potente e flessibile. L'obiettivo di fondo è quello di fare in modo

che i dati rimangano dei dati, riportando l'attenzione su una struttura del documento basata sui contenuti (cioè sui dati), invece di basarsi esclusivamente sulla rappresentazione grafica che si realizza nel browser. XML comprende le funzionalità dell'HTML ma non si limita ad esse, garantendo così strutture di contrassegno che permettono una ricerca delle specifiche informazioni nel testo.

Tuttavia all'HTML resta l'importante merito di aver iniziato l'opera di semplificazione dell'SGML e di aver reso possibile la diffusione dei linguaggi di markup presso amatori e personale non specializzato. Proprio grazie all'analisi degli errori strutturali di HTML è stato possibile progettare al meglio l'XML e rendere familiare alla massa degli utenti la struttura gerarchica degli elementi nidificati e le proprietà degli attributi. L'XML ha l'obiettivo di riproporre le intenzioni iniziali dell'HTML ma fornisce maggiori garanzie per una crescita coerente e graduale dell'ambiente della rete.

Anche se come visto precedentemente l'XML non è una vera e propria alternativa del linguaggio HTML, le sue caratteristiche comportano la fine dello sviluppo tecnico del suo predecessore.

Il W3C (Word Wide Web Consortium)

Come accennato in precedenza l'XML è una specifica realizzata dal W3C, il **World Wide Web Consortium**, il quale ha anche partecipato allo sviluppo dell'HTML e dell'SGML. E' opportuno chiarire meglio le finalità di questa organizzazione con l'intenzione di avere un'idea più chiara delle motivazioni che hanno spinto a implementare XML.

Il W3C nasce nel 1994 e si configura come un gruppo di interesse composto da organizzazioni membro; attualmente ne fanno parte più di quattrocento ed il consorzio ha dipartimenti di studio localizzati in Europa (Francia), Stati Uniti e Giappone. L'organismo non è un'agenzia governativa e il suo scopo principale è quello di creare standard da utilizzare sul web al fine di incrementare il potenziale della rete fino ai massimi livelli. Inoltre il W3C persegue l'obiettivo di rendere disponibile l'insieme delle informazioni accessibili in rete (ovvero il web vero e proprio) a chiunque, indipendentemente dalla dotazione hardware, software e dalle caratteristiche socio-culturali dell'utente. Il web è visto come un mezzo per scambiare informazioni tra i soggetti ed a questo scopo il W3C fornisce gli strumenti attraverso i quali è possibile tradurre le informazioni in linguaggio macchina. I principi ispiratori dell'organizzazione sono quelli di semplicità, compatibilità e modularità dei progetti.

Il W3C è un consorzio neutrale che cerca di promuovere l'interoperabilità dei componenti software mediante la promozione di linguaggi informatici innovativi e la diffusione di protocolli non proprietari. Come ci si può rendere conto, attualmente si ha una notevole libertà di scelta nell'acquisto di un software, senza avere l'ansia che i nuovi dati prodotti non siano compatibili con i programmi che sono già in nostro possesso. Questo risultato, a cui si è giunti solo in parte e nel corso del tempo, è merito di organizzazioni quali il W3C che cercano di creare ambienti di sviluppo più collaborativi e che fanno in modo che ognuno si prenda le proprie responsabilità per quello che viene pubblicato in internet. Inoltre il W3C fa propria l'idea che la flessibilità del web è un requisito indispensabile; essa è l'anima di internet e quindi occorre promuovere sistemi distribuiti per aumentarne le potenzialità.

Le specifiche prodotte dal consorzio sono classificabili in tre fasce di importanza:

- Le **note** *(notes)*, ossia specifiche proposte da un'organizzazione membro del consorzio che sono state pubblicate anche se non ancora approvate;
- Le **proposte di lavoro** *(working drafts),* ovvero specifiche sottoposte ad esame;
- Le **raccomandazioni** (recommendations), cioè proposte di lavoro che sono state approvate e che diventano quindi degli standard per il web.

Quando i progettisti del W3C iniziarono a sviluppare XML volevano combinare la semplicità di HMTL con la potenza di SGML. Il risultato del loro lavoro doveva rispettare alcune considerazioni di base quali:

- Si cercava di rendere il linguaggio immediatamente funzionante sul web, in modo da garantirne una rapida diffusione tra gli utenti ed offrire funzionalità rivolte a molteplici applicazioni software e non soltanto ai browser;
- XML doveva assicurare la compatibilità con lo standard SGML, soprattutto per agevolare le grandi imprese che avevano usato questa tecnologia in passato. L'obiettivo era quello di permettere agli utenti di scrivere velocemente documenti XML e di consentire ai progettisti di creare applicazioni per la manipolazione dei dati che non comportassero troppe problematiche;
- XML doveva offrire la possibilità di essere letto in modo chiaro sia dall'uomo che dalla macchina e quindi doveva essere conservato in modo da avere un livello minimo di caratteristiche opzionali (ossia soluzioni ad hoc per particolari contesti e ambienti di sviluppo), al fine di ricercare la massima compatibilità tra i sistemi;
- XML doveva essere implementato velocemente e lanciato sul mercato in tempi rapidi per rispondere tempestivamente alle pressanti esigenze degli operatori. Tuttavia si doveva garantire la presenza di una documentazione accurata e precisa del progetto per evitare così gli errori commessi in passato con gli altri standard.

Analizzando la storia recente dello sviluppo di XML si può affermare che queste premesse iniziali sono state rispettate.

Pianificazione e collaborazione

L'XML offre potenzialità incredibili agli utenti, ma occorre che i soggetti coinvolti nella realizzazione di un progetto basato su questa tecnologia conoscano analiticamente gli strumenti messi a disposizione e le modalità operative per impiegarli al meglio. Il problema di fondo è che l'uso di XML può comportare molti problemi ai progetti che risultano male impostati, soprattutto per quello che riguarda la manutenzione e l'aggiornamento dei documenti. Molti costrutti potrebbero apparire perfetti in superficie, ma in realtà non lo sono e costituiscono vere e proprie trappole per gli sviluppatori. Infatti il linguaggio non si limita a quello che appare sullo schermo, ma implica la costruzione di strutture e la creazione di relazioni tra i dati stessi.

Nell'ottica di un impiego in azienda di XML è importante garantire che la soluzione creata sia efficiente e quindi è indispensabile coinvolgere attivamente le varie unità aziendali interessate e sviluppare un ambiente collaborativo fra i soggetti. Un numero maggiore di persone deve contribuire al processo; si deve attivare una comunicazione costante fra gli agenti ed è opportuno integrare maggiormente in azienda l'attività del team di sviluppo del progetto per quello che riguarda l'automazione del flusso di lavoro. Si rende così possibile il passaggio a soluzioni flessibili e personalizzabili per la gestione dei documenti e dei dati aziendali, che tuttavia richiedono un processo di pianificazione che può richiedere tempo e risorse. Infatti occorre convertire gradualmente la situazione precedente, caratterizzata da strumenti standardizzati ormai assimilati nei processi aziendali, con gli strumenti resi possibili da XML. In particolare il linguaggio consente di raggiungere praticamente ogni applicazione per l'elaborazione dei dati e per lo sviluppo del web. Nonostante il lessico di contrassegno non si limiti ad essere esclusivamente uno strumento per il web, sarà proprio attraverso il web che molte organizzazioni entreranno a contatto con questa nuova tecnologia.

Con XML è possibile gestire l'interscambio dei dati fra i sistemi, permettendo all'impresa di sfruttare in modo più completo le risorse informative interne, rispetto a quello che consentiva in precedenza HTML. Convincere il management a sostituire la tecnologia presente in azienda con soluzioni basate su XML non è sempre facile. Le perplessità derivano soprattutto dai vincoli relativi agli investimenti effettuati nel passato, attuati molto spesso in impresa per attivare applicazioni software in grado di evitare il problema dei conflitti di comunicazione fra i sistemi. Tuttavia è necessario capire che le tecnologie antecedenti a XML rappresentano solo

delle soluzioni temporanee alle difficoltà di interscambio dei dati e non offrono garanzie per il futuro. XML secondo le aspettative dovrebbe diventare il formato standard universale per i file circolanti nel web e dovrebbe eliminare tutti i costi che attualmente si rendono necessari per la conversione dei documenti nei diversi formati proprietari.

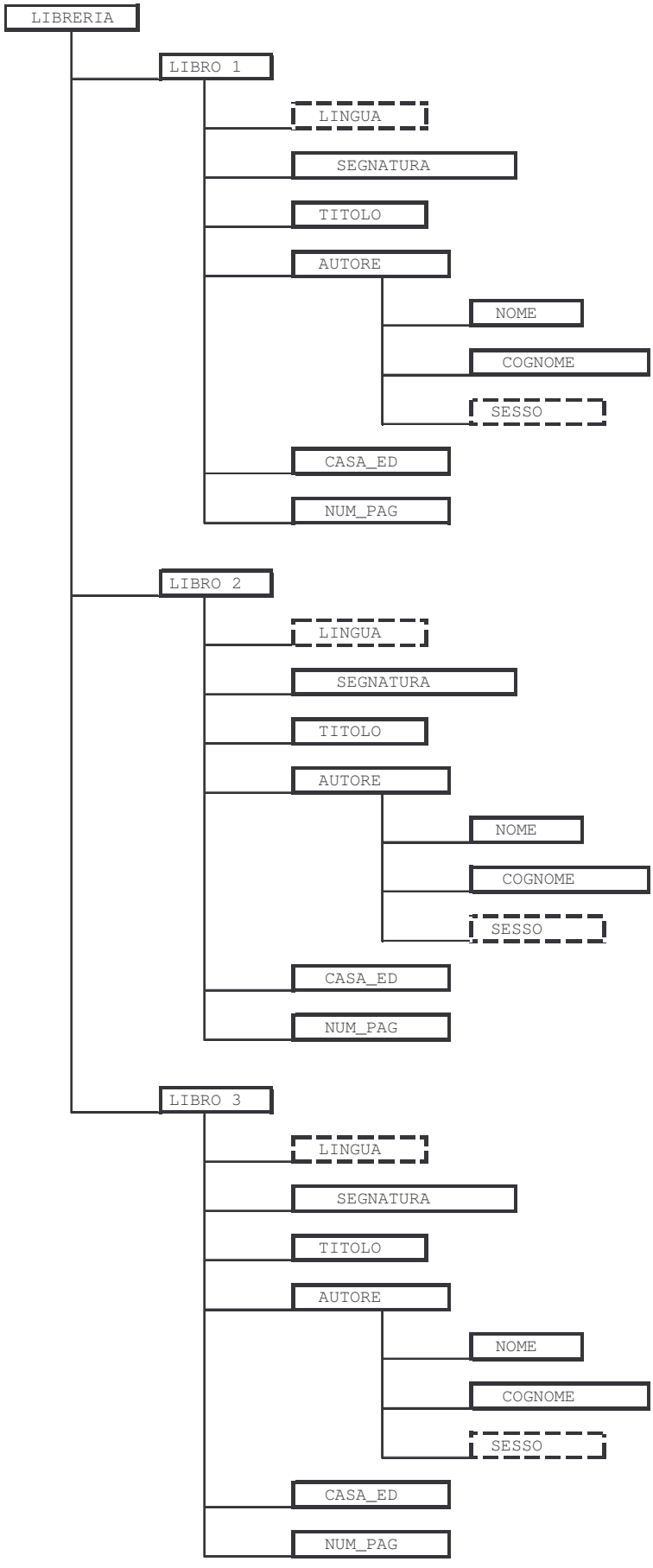
1.1.2 Struttura di XML

L'XML offre il vantaggio fondamentale, rispetto agli altri linguaggi di markup, di poter fornire una struttura ad un documento. Ciò presuppone una standardizzazione dell'organizzazione del progetto costruito dagli sviluppatori. Questa necessità si concretizza in un'analisi più dettagliata dei legami e delle relazioni che intercorrono tra i dati, che dunque vengono definite in modo più esplicito del consueto. L'HTML al contrario non collega in nessun modo particolare le parti che compongono un documento e non impone vincoli sostanziali nell'uso degli elementi. La definizione chiara e precisa di un progetto consente di garantire che tutti gli elementi richiesti in partenza siano effettivamente presenti e localizzati in posizioni corrette, rendendo così più facile la lettura da parte dei progettisti. Inoltre si agevola il compito dei sistemi di gestione dei documenti nel verificare la completezza dei file e si fornisce così un percorso all'informazione in modo da consentire una rapida selezione.

Esistono una serie di regole strutturali che un documento XML deve necessariamente rispettare per essere considerato valido. In particolare in ogni file XML è possibile individuare una struttura logica ed una struttura fisica. La **struttura logica** (*logical structure*) di un documento è un modello che descrive quali elementi vengono inclusi e in quale ordine. Tramite questa architettura si analizza dunque l'organizzazione delle parti costituenti il testo. La **struttura fisica** (*physical structure*) invece è formata dall'insieme dei dati effettivi relativi ad un documento e quindi si identifica come il contenuto del documento stesso.

Per meglio comprendere questi due concetti si farà uso di un semplice esempio che sarà poi ripreso nelle pagine successive per facilitare la descrizione dei concetti illustrati. In particolare l'esempio proposto fa riferimento alla creazione di un elenco ordinato di libri presenti in una ipotetica libreria e dei quali si vogliono conoscere le caratteristiche fondamentali per consentire una rapida consultazione del catalogo e per permettere di estrapolare informazioni in base a specifici criteri di selezione (ad esempio fare un filtraggio dei dati in base alla data di pubblicazione del libro). Si propone quindi una lista di tre libri appartenenti all'elemento radice "libreria", ciascuno dei quali presenta degli elementi descrittivi, quali il titolo, l'autore (con definizione di nome, cognome e sesso), la segnatura, la lingua, l'anno di pubblicazione e la casa editrice.

Analizzando il contenuto di un file XML si nota che i dati sono vincolati da precise relazioni gerarchiche. Attraverso l'analisi dei legami è possibile rappresentare la struttura logica con la tipica schematizzazione ad albero del documento XML. In tal modo è possibile osservare la nidificazione dei componenti, ossia il processo attraverso il quale è possibile incorporare un elemento in un altro. Per una descrizione più completa della sintassi richiesta per la dichiarazione di un documento XML si rinvia la lettura ai paragrafi che illustrano le DTD e soprattutto quelli che riguardano gli Schemi XML. Riprendendo l'esempio proposto si ottiene:


LIBRERIA
LIBRO 1
LINGUA
SEGNATURA
TITOLO
AUTORE
NOME
COGNOME
SESSO
CASA_ED
NUM_PAG
LIBRO 2
LINGUA
SEGNATURA
TITOLO
AUTORE
NOME
COGNOME
SESSO
CASA_ED
NUM_PAG
LIBRO 3
LINGUA
SEGNATURA
TITOLO
AUTORE
NOME
COGNOME
SESSO
CASA_ED
NUM_PAG

E' possibile vedere che il documento XML è composto da una pluralità di oggetti chiamati **elementi** (*elements*), oppure nodi della struttura ad albero. L'elemento rappresenta il costrutto essenziale sul quale viene poi organizzata la struttura del documento. Il nodo base (*root element*), che contiene tutti gli altri nodi è chiamato **radice** (che nell'esempio specifico è LIBRERIA). Il nodo radice deve essere necessariamente presente e deve includere tutti gli altri elementi, al di fuori delle istruzioni di elaborazione. Nel momento in cui un elemento è incluso in un altro (ad esempio LIBRO nei confronti di LIBRERIA) si definisce **figlio del nodo esterno** *(child element),* mentre se due nodi sono sullo stesso livello (ad esempio TITOLO e SEGNATURA), questi ultimi sono detti **nodi fratelli** (*sibling elements*). Si ricorda inoltre che il testo contenuto fra due tag è detto **nodo di testo** (*text element*) e che è consentito usare **elementi vuoti** (*empty elements*), cioè senza alcun argomento interno. Per quanto riguarda il nome che si può assegnare agli elementi, il linguaggio XML non risulta particolarmente restrittivo; ogni combinazione di caratteri è valida, purché non si usino caratteri riservati (ad esempio < o >) e si rispetti la distinzione nel testo tra caratteri minuscoli e maiuscoli. Vi è inoltre la possibilità di includere nel documento degli **attributi** (*attributes*) relativi ai vari elementi che sono composti da due parti: il nome dell'attributo stesso e il valore assegnato. Ad esempio l'attributo dell'elemento LIBRO chiamato "lingua" assume un valore pari a "italiano". Gli attributi consentono di avere un maggiore controllo sugli elementi, specificandone ex-ante l'aspetto, la posizione ed i comportamenti.

Per quanto riguarda la struttura fisica del progetto XML si presenta il documento XML vero e proprio preso precedentemente in considerazione:

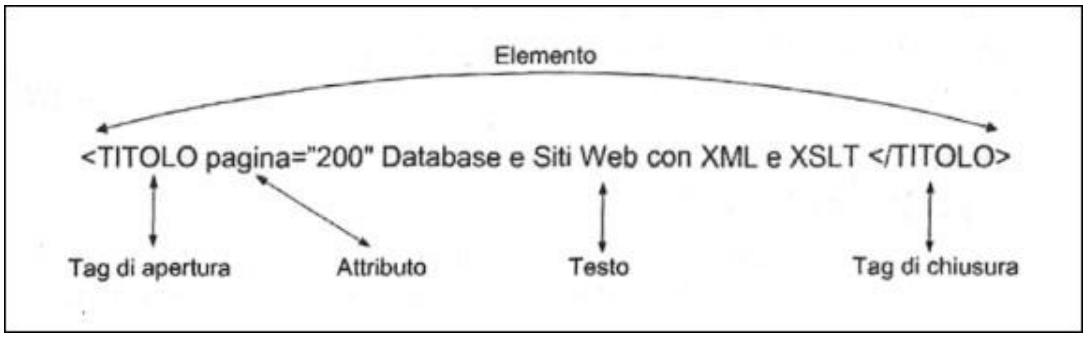
```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 edited with XMLSPY v2004 rel. 2 (http://www.xmlspy.com) by Aste
Walter
  -->
-   <LIBRERIA   xmlns:xsi="http://www.wA1.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\WINDOWS\Desktop\Xml\prova        di
XmlSchema.xsd">
- <LIBRO lingua="italiano">
  <SEGNATURA>2fN TOLK6</SEGNATURA>
  <TITOLO>Il Signore degli Anelli</TITOLO>
- <AUTORE sesso="maschile">
  <NOME>John</NOME>
  <COGNOME>Tolkien</COGNOME>
  </AUTORE>
  <ANNO>2000</ANNO>
  <CASA_ED>Bompiani</CASA_ED>
  <NUM_PAG>1360</NUM_PAG>
  </LIBRO>
- <LIBRO lingua="inglese">
  <SEGNATURA>5AM VER1</SEGNATURA>
  <TITOLO>The league of extraordinary gentlemen</TITOLO>
- <AUTORE sesso="maschile">
  <NOME>Alan</NOME>
  <COGNOME>Moore</COGNOME>
  </AUTORE>
  <ANNO>2002</ANNO>
  <CASA_ED>Magic Press</CASA_ED>
  <NUM_PAG>200</NUM_PAG>
```

```
  </LIBRO>
- <LIBRO lingua="italiano">
  <SEGNATURA>X-NA VER 2</SEGNATURA>
  <TITOLO>L'isola misteriosa</TITOLO>
- <AUTORE sesso="maschile">
  <NOME>Jules</NOME>
  <COGNOME>Verne</COGNOME>
  </AUTORE>
  <ANNO>1995</ANNO>
  <CASA_ED>Einaudi</CASA_ED>
  <NUM_PAG>498</NUM_PAG>
  </LIBRO>
  </LIBRERIA>
```

Si può notare come i dati siano racchiusi tra i tag di apertura e di chiusura, il che permette una agevolazione nelle operazioni di ricerca e di filtraggio delle informazioni. Mentre la struttura logica organizza il documento in modo da aiutare la lettura dell'organizzazione dei dati, la struttura fisica riflette direttamente il contenuto, senza prestare particolare attenzione alla posizione dei dati nel documento. La struttura logica è utile all'autore del file XML e al suo lettore, ma non molto per l'applicazione software, la quale deve basarsi prevalentemente sulla organizzazione fisica dei dati per un riutilizzo ottimale delle informazioni contenute nel documento XML. Il linguaggio di markup consente quindi di fornire la materia prima, ossia i dati contrassegnati in modo appropriato, sui quali costruire applicazioni eterogenee. L'informazione è marcata in modo da specificare insiemi limitati e dettagliati di dati che ne consentono così un accesso più flessibile e più affidabile rispetto agli altri formati di base per la memorizzazione dei dati (ad esempio campi a lunghezza fissa o campi con delimitatori). In particolare i dati sono spesso suddivisi in parti più piccole di quelle che si rendono necessarie per l'analisi.

1.1.3 Documenti XML validi e documenti XML ben formati

Un documento XML si presenta come un comune file di testo all'interno del quale sono definite una serie di istruzioni denominate da contrassegno. Un esempio di sintassi di un elemento XML è:



fonte: G. Forlino (2003: pag 14)

Un **documento valido** (*valid XML document*) ha una DTD associata, oppure uno Schema XML (per un approfondimento di questi temi si rinvia la lettura ai paragrafi successivi), nei confronti dei quali sono state verificate tutte le regole strutturali che comportano. Un documento valido è anche un documento ben formato. Con il termine **documento XML ben formato** (*well-formed XML document*) si intende indicare un particolare file XML che rispetta

una serie di regole e più precisamente le regole di sintassi stabilite dal consorzio W3C nelle specifiche di XML 1.0.
I principali requisiti sono:

1. Ogni documento deve presentare un **prologo** che identifica alcune informazioni, quali la versione specifica di XML a cui è conforme il documento e la natura XML del documento. Il prologo deve sempre occupare la parte iniziale del file.

```
<?xml version="1.0" encoding="UTF-8" ?>
 <!-- edited with XMLSPY v2004 rel. 2 (http://www.xmlspy.com)
 by Aste Walter -->
```

2. Si deve rispettare il corretto **processo di nidificazione dei tag** (*nest-building process*), nel senso che ogni coppia di tag di apertura e di chiusura deve includere perfettamente ogni altra coppia di tag che inizia al suo interno.

```
<LIBRO> <TITOLO>.....</TITOLO></LIBRO>
```

3. Deve esistere un tag radice che contiene tutti gli altri. E' un elemento univoco dal quale discendono gerarchicamente tutti gli altri.

```
- <LIBRERIA
- + <LIBRO lingua="italiano">
- + <LIBRO lingua="inglese">
- + <LIBRO lingua="italiano">
- </LIBRERIA>
```

4. Il nome del tag di apertura deve coincidere con il relativo tag di chiusura, facendo sempre attenzione ad usare i corretti caratteri minuscoli o maiuscoli. Tutti i tag aperti devono essere chiusi.

```
<SEGNATURA>2fN TOLK6</SEGNATURA>
```

5. Uno stesso attributo può apparire una sola volta in un tag di apertura e il suo nome deve rispettare le regole di definizione degli elementi. Il valore degli attributi deve essere racchiuso tra apici.

```
- <LIBRO lingua="italiano">
```

6. Nel caso sia necessario includere nel documento dei caratteri particolari che si riferiscono al contrassegno (ad esempio < o >) o ad altri caratteri speciali, occorre far uso di un'**entità** (ad esempio < è pari al carattere <). Infatti in XML esistono dei simboli che rivestono particolare significato e di cui non si può fare un libero uso. Con il termine entità si descrive genericamente un'unità di memorizzazione dei dati.

Dunque se il documento rispetta le specifiche è ben formato; un documento ben formato può essere successivamente dichiarato valido se soddisfa gli ulteriori vincoli imposti dagli Schemi.

Per ulteriori approfondimenti relativi ai concetti esposti si consigliano le seguenti letture:

⇒ specifica ufficiale del W3C:
  - http://www.wA1.org/TR/2000/REC-xml-20001006

⇒ alcuni testi informatici indicati nella bibliografia e facilmente reperibili in libreria o in biblioteca:
  - Livingston D. (2002), "XML", Tecniche Nuove: Milano;
  - Phillips L. (2000), "Usare XML", Mondadori: Milano;
  - Pialorsi P. (2002), "Xml: il nuovo linguaggio del web", Mondadori: Milano;
  - St. Laurent S. (1999), "XML le basi", Tecniche Nuove: Milano;

⇒ manuali tecnici in lingua inglese acquistabili sul sito www.amazon.com:
  - Harold E.R, Means W.S (2002), "XML in a nutshell, 2th edition", O'Reilly & Associates;
  - Young M.J. (2001), "XML step by step, Second Edition", Paperback;
  - Chaudhri A.B. (2003), "XML Data Management: Native XML and XML-Enabled Database System", Addison-Wesley Pub Co.;
  - Eckstein R. (2000), "XML pocket reference", O'Reilly & Associates.

1.2 DTD e Xml Schema

Sia HTML che XML necessitano di un'applicazione che ne esegua le procedure. Normalmente la visualizzazione di un documento XML si rende possibile tramite l'uso di un comune browser, il quale mostra la struttura gerarchica ad albero, consentendo all'utente di definire il livello di dettaglio di visualizzazione. Un **parser XML** è un semplice programma, che normalmente risiede all'interno dei moduli del browser, il cui compito è leggere il documento XML ed estrarne le informazioni. Il ruolo fondamentale del parser è dunque quello di interpretare un testo specificandone la natura.

L'XML, derivando direttamente le sue caratteristiche dall'SGML, prevede un'interpretazione del contenuto di un documento più complessa di quella che si verifica abitualmente nella costruzione di file HTML. Occorrono una serie di regole specificate dal progettista ed a cui il documento deve attenersi affinché sia ritenuto valido dal parser. In dettaglio si analizza il contrassegno per verificare la corrispondenza alla specifiche contenute nella Dichiarazione di Tipo di Documento (DTD) e negli Schemi XML. Esistono parser chiamati "validanti", che hanno il compito di riscontrare la correttezza della struttura del file rispetto allo Schema e che si distinguono dai parser non validanti, i quali si limitano a verificare che il documento sia ben formato.

Le DTD

XML è un linguaggio di markup estensibile dal momento che permette di creare altri linguaggi di markup personalizzati a seconda delle esigenze. Mentre nell'HTML un singolo tag ha uno specifico significato che è compreso in modo univoco da tutti i browser, nell'XML non si ha questa possibilità, visto che personalizzando il contrassegno, ogni tag può avere significati completamente differenti a seconda del contesto in cui viene sviluppato. Il W3C ha previsto la possibilità di fornire un dizionario al documento XML, ossia di allegare un insieme di regole che permette all'applicazione di leggere correttamente il markup utilizzato. Questo ruolo è stato inizialmente svolto dalle DTD (**Document Type Definition**).

Una DTD è un file di testo che definisce analiticamente le singole parti che costituiscono un documento XML, quali gli elementi, gli attributi, le entità e le relazioni tra gli elementi. Le DTD usano una sintassi che non è quella di XML e quindi comportano l'acquisizione di ulteriori conoscenze da parte dei progettisti. Le definizioni di tipo di documento possono essere incluse nel file XML o essere esterne e venire richiamate nel documento. La loro dichiarazione è localizzata subito dopo la dichiarazione XML, ossia dopo la prima riga del

testo del codice. Lo scopo finale di tale procedimento è quello di implementare un metodo standard attraverso il quale rendere conformi documenti diversi, in modo da aumentare le possibilità di comunicazione. La DTD consente di garantire che il documento contenga tutte le informazioni necessarie all'applicazione, in un formato tale da essere letto in modo esatto. Quindi si rende possibile l'interpretazione corretta del codice da parte della macchina. Le DTD consentono di definire gli elementi presenti nel documento e ne determinano in modo piuttosto generale il tipo di contenuto che è consentito (ad esempio si possono elencare gli elementi figli del nodo genitore validi, determinandone l'ordine, il numero di ripetizioni e il tipo di testo ammesso). La specificazione del formato dei dati supportata dalla DTD è piuttosto elementare visto che sono garantite solamente due modalità: tipo PCDATA, cioè dati che saranno poi interpretati dal parser e che sono costituiti dalle classiche combinazioni di caratteri e contrassegno, e tipo CDATA, cioè dati senza contrassegno che vengono esclusi dalla validazione del parser. Ogni altro tipo di formato dei dati non è previsto. E' consentito inoltre procedere ad una definizione degli attributi, chiarendone l'eventuale valore di default, il tipo ed i valori che sono ammessi. Sempre nelle DTD vengono dichiarate le entità che verranno successivamente richiamate nel documento. La dichiarazione di tipo di documento collega la DTD al documento reale ed è scritta nel documento XML. Tuttavia permettendo l'uso di DTD esterne, gli sviluppatori aumentano la possibilità di un loro uso comune tra più documenti, garantendone la compatibilità. Le DTD sono un metodo attraverso il quale i progettisti possono descrivere in modo più dettagliato l'organizzazione degli elementi e degli attributi in un file XML, rendendo così più agevole l'estrazione delle informazioni e la loro condivisione. A titolo illustrativo si presenta la DTD relativa all'esempio che è stato realizzato nel corso del capitolo. Tuttavia non verrà fornita una descrizione del codice, in quanto le DTD sono una tecnologia diventata ormai obsoleta nel mondo di XML. Inoltre non si intende trattare in questo contesto un'analisi approfondita della sintassi del linguaggio che è rinviata ad appositi manuali informatici.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--DTD generated by XMLSPY v2004 rel. 2 (http://www.xmlspy.com)-->
<!ELEMENT LIBRERIA (LIBRO)+>
<!ELEMENT LIBRO (SEGNATURA, TITOLO, AUTORE+, ANNO, CASA_ED, NUM_PAG)>
<!ATTLIST LIBRO
   lingua (italiano | inglese | francese | spagnolo | tedesco)
#IMPLIED
>
<!ELEMENT SEGNATURA (#PCDATA)>
<!ELEMENT TITOLO (#PCDATA)>
<!ELEMENT AUTORE (NOME, COGNOME)>
<!ATTLIST AUTORE
   sesso (maschile | femminile) #IMPLIED
>
<!ELEMENT ANNO (#PCDATA)>
<!ELEMENT CASA_ED (#PCDATA)>
<!ELEMENT NUM_PAG (#PCDATA)>
<!ELEMENT NOME (#PCDATA)>
<!ELEMENT COGNOME (#PCDATA)>
```

Per ulteriori approfondimenti relativi ai concetti esposti si consigliano le seguenti letture:

⇒ specifica ufficiale del W3C:

- http://www.wA1.org/TR/2000/REC-xml-20001006

⇒ alcuni testi informatici indicati nella bibliografia e facilmente reperibili in libreria o in biblioteca:

- Livingston D. (2002), "XML", Tecniche Nuove: Milano;
- Phillips L. (2000), "Usare XML", Mondadori: Milano;
- St. Laurent S. (1999), "XML le basi", Tecniche Nuove: Milano;

⇒ manuali tecnici in lingua inglese acquistabili sul sito www.amazon.com:

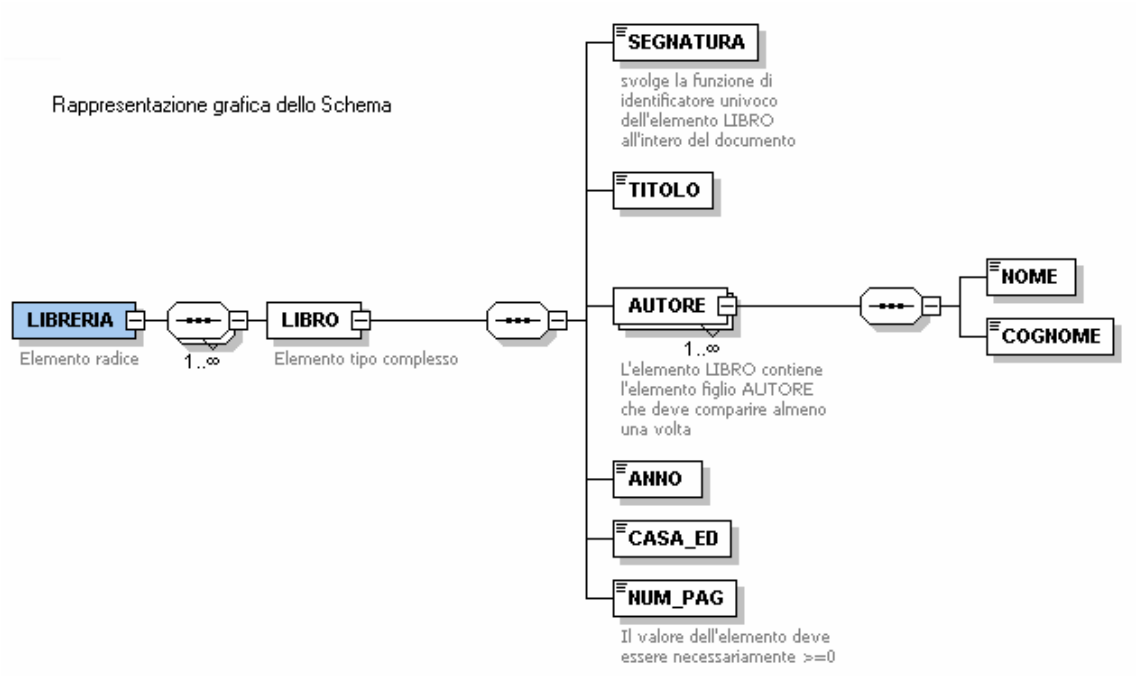
- St Laurent S., Biffor R.J. (1999), "Inside XML DTD'S: Scientific and Technicals", Mc Graw Hill Companies;

Schemi XML

Negli ultimi anni l'importanza delle DTD è andata via via diminuendo a causa della complessità di implementazione che le contraddistingue ed a causa dello scarso controllo sul contenuto dei dati che è consentito. Tuttavia la DTD non è l'unica tecnologia attraverso la quale è possibile definire e controllare la struttura dei documenti scritti in XML. Recentemente si è sviluppato l'uso degli **schemi XML**, un linguaggio scritto in XML e che è una specifica rilasciata il 2 maggio 2001 dal W3C. Visto che XML è un metalinguaggio, il W3C ha sviluppato una grammatica che consente di descrivere la struttura dei documenti usando la stessa sintassi.

Gli Schemi pur essendo sostanzialmente dei file XML, sono trattati differentemente rispetto ad un comune documento e permettono quindi di raggiungere in modo più semplice e con maggiore precisione gli obiettivi che le DTD promettevano. I semplici formati PCDATA e CDATA delle DTD sono evidentemente insufficienti per le necessità di scambio di dati strutturati e quindi la nuova tecnologia è stata adottata dalla comunità informatica con il preciso compito di essere la nuova generazione di linguaggi di definizione delle strutture dei dati. Gli Schemi consentono un controllo sull'organizzazione dei dati che permette di verificare sia la validità del documento XML, sia che esso sia ben formato.

Viene di seguito proposto il codice dello Schema XML relativo all'esempio esposto precedentemente. Nelle pagine seguenti tale codice sarà scomposto e analizzato, seppur in modo generale, al fine di descrivere le principali regole di sintassi proprie del linguaggio.



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 2 (http://www.xmlspy.com) by Aste
Walter -->
<xs:schema
 xmlns:xs="http://www.wA1.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="LIBRERIA">
     <xs:complexType>
        <xs:sequence maxOccurs="unbounded">
           <xs:element name="LIBRO">
              <xs:annotation>
                 <xs:documentation>Elemento tipo complesso
                 </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                 <xs:sequence>
                    <xs:element name="SEGNATURA" default="XXXXXX">
                       <xs:annotation>
                          <xs:documentation>svolge la funzione di
                           identificatore univoco dell'elemento LIBRO
                           all'intero del documento</xs:documentation>
                       </xs:annotation>
                       <xs:simpleType>
                          <xs:restriction base="xs:string">
                             <xs:whiteSpace value="preserve"/>
                             <xs:minLength value="1"/>
                             <xs:maxLength value="10"/>
                          </xs:restriction>
                       </xs:simpleType>
                    </xs:element>
                    <xs:element name="TITOLO">
                       <xs:simpleType>
                          <xs:restriction base="xs:string">
                             <xs:whiteSpace value="preserve"/>
                             <xs:maxLength value="150"/>
                             <xs:minLength value="1"/>
                          </xs:restriction>
                       </xs:simpleType>
                    </xs:element>
                    <xs:element name="AUTORE" maxOccurs="unbounded">
                       <xs:annotation>
                          <xs:documentation>L'elemento LIBRO contiene
                           l'elemento figlio AUTORE che può comparire
                           almeno una volta</xs:documentation>
                       </xs:annotation>
```

```
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="NOME">
                          <xs:simpleType>
                            <xs:restriction base="xs:string">
                              <xs:whiteSpace value="preserve"/>
                              <xs:minLength value="1"/>
                              <xs:maxLength value="20"/>
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:element>
                        <xs:element name="COGNOME">
                          <xs:simpleType>
                            <xs:restriction base="xs:string">
                              <xs:minLength value="1"/>
                              <xs:maxLength value="20"/>
                              <xs:whiteSpace value="preserve"/>
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:element>
                      </xs:sequence>
                      <xs:attribute name="sesso" use="optional">
                        <xs:simpleType>
                          <xs:restriction base="xs:string">
                            <xs:enumeration value="maschile"/>
                            <xs:enumeration value="femminile"/>
                          </xs:restriction>
                        </xs:simpleType>
                      </xs:attribute>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="ANNO">
                    <xs:annotation>
                      <xs:documentation>Elemento
                       tipo xs:gYear
                      </xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                      <xs:restriction base="xs:gYear">
                        <xs:whiteSpace value="collapse"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="CASA_ED">
```

```
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:whiteSpace value="preserve"/>
                        <xs:minLength value="1"/>
                        <xs:maxLength value="20"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="NUM_PAG">
                    <xs:annotation>
                      <xs:documentation>Il valore dell'elemento
                       deve essere necessariamente >=0
                    </xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                      <xs:restriction base="xs:positiveInteger">
                        <xs:whiteSpace value="collapse"/>
                        <xs:minInclusive value="1"/>
                        <xs:maxInclusive value="5000"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="lingua"
                 use="optional" default="italiano">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:whiteSpace value="collapse"/>
                      <xs:minLength value="1"/>
                      <xs:maxLength value="10"/>
                      <xs:enumeration value="italiano"/>
                      <xs:enumeration value="inglese"/>
                      <xs:enumeration value="francese"/>
                      <xs:enumeration value="spagnolo"/>
                      <xs:enumeration value="tedesco"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:attribute>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
</xs:schema>
```

Analisi del codice dello Schema XML

Gli Schemi dei documenti XML iniziano obbligatoriamente con una **dichiarazione**. Il compito di questa intestazione è indicare all'applicazione che il codice, compreso tra i tag di apertura e di chiusura, contiene informazioni sulla struttura del documento.

```
<xs:schema                xmlns:xs="http://www.wA1.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
```

Viene in particolare definito un **namespace**, ossia uno spazio dei nomi che ha il compito di far trattare in modo diverso dal parser gli elementi XML dello Schema rispetto agli altri.

Tutti i namespace che si riferiscono ad uno schema XML presuppongono l'utilizzo dell'attributo *xmlns:xs* che assume un valore fisso pari a *" http:// www.wA1.org /2001 / XMLSchema"*. In questo modo viene indicato al parser XML che tutte le informazioni contenute nello spazio dei nomi sono da considerarsi come elementi ed attributi dello schema. Osservando il comando si nota che il valore assegnato ha la stessa forma di un indirizzo internet (URI) e quindi in un primo momento sembrerebbe che lo schema effettui un collegamento ad una risorsa esterna. Tuttavia non viene effettuata alcuna operazione di connessione. La stringa di codice viene utilizzata unicamente come metodo standard (raccomandato dal W3C) per segnalare al parser e all'utente che il documento preso in considerazione è riconducibile a XML Schema (cioè è un'istanza dello schema) e che quindi va trattato in modo diverso dal documento che contiene i dati.

Inoltre se si presta attenzione al modo in cui lo spazio dei nomi viene dichiarato si può notare che viene assegnato anche un nome locale al namespace e che questo risultato si realizza con il comando *"xs:schema"*. In questo modo si può ottenere una particolare ridefinizione degli elementi che aiuta la lettura: tutti gli elementi appartenenti a XML Schema iniziano con il prefisso *"xs:"*, mentre quelli del documento XML vero e proprio non hanno alcun prefisso.

Nell'esempio proposto si è definito esclusivamente il namespace di default relativo alla tecnologia XML Schema ("xs:"). Tuttavia il progettista di uno Schema potrebbe prevedere la presenza di ulteriori dichiarazioni di namespace per rispondere a diverse necessità informative. In questi casi va tenuto presente che l'operazione di dichiarazione dello spazio dei nomi non è sufficiente per la corretta configurazione dello Schema. Occorre infatti effettuare un'ulteriore operazione attraverso la quale assegnare il namespace creato agli elementi esistenti. A tal scopo sono stati ideati gli attributi *"elementFormDefault"* e *"attributeFormDefault"* rispettivamente necessari per collegare gli elementi e per collegare gli attributi del documento al namespace. I due attributi possono assumere valore *"qualified"* nel caso in cui il progettista abbia l'intenzione di includere l'elemento/attributo nello spazio dei nomi o in alternativa il valore *"unqualified"* nel caso in cui si voglia escludere il valore dell'elemento/attributo dal namespace. Inoltre tramite l'uso dell'attributo "*targetNamespace*" si indica la posizione di rete dello Schema creato. Ad esempio in uno Schema si potrebbe dichiarare un namespace aggiuntivo oltre a quello di default:

```
.....
.....
<xs:schema xmlns:xs="http://www.wA1.org/2001/XMLSchema"
                    xmlns:es="http://www.esempio.it/esempio"
                     targetNamespace="http://www.esempio.it/esempio"
                     elementFormDefault="qualified"
                     attributeFormDefault="unqualified" >
.....
```

Nel documento XML relativo allo Schema di riferimento si avrebbe così una situazione in cui i nomi degli elementi sono qualificati con il prefisso del namespace aggiuntivo ("*es*:"), mentre il nome associato agli attributi resta non qualificato.

```
……
……
<es:città> …….</es:città>
<es:provincia stato="Italia">……</es:provincia>
……
……
```

Per quanto riguarda la dichiarazione degli elementi e degli attributi che compongono il documento XML, si osserva che XML Schema distingue fra due tipi base: tipi semplici e tipi complessi.

I tipi semplici

Gli **elementi tipo semplice** (*simple type element*) possono contenere esclusivamente testo o essere vuoti. Quindi sono usati come contenuto testuale di nodi elemento o nodi attributo, ma non possono in ogni caso contenere altri elementi o attributi. Esistono una serie di tipi semplici impostati per default da XML Schema ed a cui ci si può appoggiare nella costruzione del file.
I principali tipi semplici sono:

NOME DEL TIPO	DESCRIZIONE
xs:string	Consente agli elementi di contenere testo.
xs:integer	Consente agli elementi di contenere un qualsiasi numero positivo, negativo o nullo.
xs:positiveInteger	Consente agli elementi di contenere un qualsiasi numero positivo.
xs:negativeInteger	Consente agli elementi di contenere un qualsiasi numero negativo.
xs:decimal	Consente agli elementi di contenere solo numeri (positivi o negativi).
xs:float	Consente agli elementi di contenere solo numeri a virgola mobile.
xs:gYear	Consente agli elementi di contenere il valore di un anno in cifre (es. 2003)

Nell'elenco sono stati inclusi esclusivamente i tipi semplici più comuni, ma ne esistono molti altri e più precisamente è possibile scegliere fra 19 tipi di base e 25 tipi derivati da essi. Inoltre è possibile crearne di personalizzati a seconda delle esigenze.
XML Schema permette di associare delle restrizioni ai valori degli elementi, come determinare la lunghezza massima consentita per il contenuto del testo (*xs:maxLength*), la lunghezza minima (*xs:minLength*), oppure specificare come formattare gli spazi bianchi della

sequenza di caratteri digitati, quindi se preservarli, eliminarli o ignorarli (*xs:whiteSpace*). Inoltre si possono limitare i valori numerici contenuti nei nodi attraverso attributi quali *maxInclusive value=""* o *minInclusive value=""*.

Le **dichiarazioni di tipo semplice** iniziano sempre con i tag <*xs:simpleType*> a cui possono seguire parametri aggiuntivi rispetto a quelli descritti precedentemente . Uno dei più comuni è <*xs:restriction*> che consente di elencare i valori che è permesso assegnare a un determinato elemento e che verranno poi dichiarati singolarmente con <*xs:enumeration=""*>.

```
<xs:attribute name="lingua" use="optional" default="italiano">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:whiteSpace value="collapse"/>
                        <xs:minLength value="1"/>
                        <xs:maxLength value="10"/>
                        <xs:enumeration value="italiano"/>
                        <xs:enumeration value="inglese"/>
                        <xs:enumeration value="francese"/>
                        <xs:enumeration value="spagnolo"/>
                        <xs:enumeration value="tedesco"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:attribute>
```

Inoltre esistono attributi particolari che descrivono la priorità dell'elemento (es. *use=""* che assume valori optional, required e fixed) e la sua cardinalità. Per **cardinalità** di un elemento si intende il numero di volte che esso può comparire validamente nel documento. Essa è specificata attraverso gli attributi *maxOccurs=""* e *minOccurs=""*, che possono assumere valori precisi (es. 1, 2 , ecc..) o il valore "unbounded" se si permettono un numero indefinito di presenze.

```
<xs:element name="LIBRERIA">
            <xs:annotation>
                  <xs:documentation>Elemento
radice</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                  <xs:sequence maxOccurs="unbounded">
                        <xs:element name="LIBRO">
……
……
```

Oltre a poter creare tipi semplici personalizzati, è permesso combinare tipi semplici standard o riutilizzare tipi costruiti precedentemente per altri scopi.

I tipi complessi

Gli **elementi tipo complesso** (*complex type element*) possono invece contenere attributi, testo, oppure una qualsiasi combinazione di altri elementi. Questo modello è appropriato per definire nodi elemento che contengono al loro interno ulteriori definizioni di nodi elemento e attributi (analizzate precedentemente) e che quindi non siano formati soltanto da testo. Gli attributi standard impiegati per descrivere il tipo complesso sono sostanzialmente gli stessi usati nel contesto dei tipi semplici, quindi si ritrovano parametri quali *maxOccurs=””, minOccurs=””* e così via.

Il primo passo nella costruzione del codice è quello di dichiarare un elemento e di assegnarli un nome. La **definizione di tipo di elemento complesso** è aperta con il tag *<xs:complexType>* che consente al parser di capire che sono inclusi molteplici componenti in uno stesso elemento. Dopo la dichiarazione è necessario porre un tag speciale che indica l'ordine nel quale devono comparire nel documento XML le varie parti che si andranno ad elencare. In particolare si usa *<xs:sequence>* se si desidera rispettare l'ordine imposto nello Schema, *<xs:all>* se si consente di elencare nel documento gli elementi in un ordine qualsiasi e *<xs:choice>* nel caso in cui si hanno elenchi di elementi da cui si può scegliere. La sintassi rimane comunque la stessa.

Una volta conclusa la descrizione del tipo complesso si procede alla definizione dei tipi semplici in esso inclusi, operazione che avviene con le medesime modalità descritte nelle pagine precedenti. Il tipo complesso è dunque un contenitore all'interno del quale sono annidati tipi semplici standard o personalizzati, attributi o testo. Ecco una semplice dimostrazione:

```
<xs:element name="AUTORE" maxOccurs="unbounded">
                    <xs:annotation>
                       <xs:documentation>L'elemento LIBRO contiene
                         l'elemento figlio AUTORE che può comparire
                         almeno una volta
                    </xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                       <xs:sequence>
                          <xs:element name="NOME">
                             <xs:simpleType>
                                <xs:restriction base="xs:string">
                                   <xs:whiteSpace value="preserve"/>
                                   <xs:minLength value="1"/>
                                   <xs:maxLength value="20"/>
                                </xs:restriction>
                             </xs:simpleType>
                          </xs:element>
                          <xs:element name="COGNOME">
                             <xs:simpleType>
                                <xs:restriction base="xs:string">
                                   <xs:minLength value="1"/>
                                   <xs:maxLength value="20"/>
                                   <xs:whiteSpace value="preserve"/>
                                </xs:restriction>
```

```
                        </xs:simpleType>
                     </xs:element>
                  </xs:sequence>
                  <xs:attribute name="sesso" use="optional">
                     <xs:simpleType>
                        <xs:restriction base="xs:string">
                           <xs:enumeration value="maschile"/>
                           <xs:enumeration value="femminile"/>
                        </xs:restriction>
                     </xs:simpleType>
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
```

Nell'esempio è stato creato un tipo complesso (Autore) che contiene due tipi semplici formattati come stringhe (Nome e Cognome) e un attributo formattato anch'esso come stringa (Sesso). Nello specifico caso è stato inserito in Autore l'attributo maxOccurs con valore unbounded; questo permette all'elemento Autore di comparire più volte all'interno del suo nodo genitore Libro. L'opzione è stata attivata in quanto il caso di un libro scritto congiuntamente da più autori è facilmente riscontrabile nella realtà. Concludendo questa panoramica generale sui tipi complessi si ricorda che è possibile creare configurazioni personalizzate basate sui tipi esistenti.

Per **associare uno Schema ad un documento XML** occorre scrivere una particolare riga di codice nel documento stesso.

```
<LIBRERIA      xmlns:xsi="http://www.wA1.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\WINDOWS\Desktop\Xml\prova        di
XmlSchema.xsd">
```

In dettaglio è richiesto l'utilizzo dell'attributo *"xsi:noNamespaceSchemaLocation"* in cui è contenuta l'indicazione del file XML Schema da collegare (cioè il riferimento univoco allo Schema) e l'indicazione della posizione di rete in cui è conservato (comandi distinti ma posizionati sulla stessa riga di codice).

L'uso di questo attributo è particolarmente importante in quanto il documento XML senza il suo appoggio non disporrebbe di alcun riferimento per recuperare le informazioni relative alla struttura contenuta negli Schemi XML. Da ricordare ancora una volta che il percorso di accesso allo Schema va dichiarato nel documento XML per la funzione che ricopre e non nello Schema XML.

Un'ipotesi alternativa a quella presentata nell'esempio è quella in cui il file di XML Schema sia presente nel web e quindi non sia disponibile localmente sul computer dell'utente. In questi casi l'attributo *"xsi:noNamespaceSchemaLocation"* va sostituito con l'attributo *"xsi:schemaLocation"* che assume come valore l'indirizzo internet corrispondente al file dello schema.

Comunque in entrambi i casi deve essere presente la riga di codice *xmlns : xsi = "http:// www.wA1.org / 2001 / XMLSchema-instance"* utilizzata per indicare al parser che il documento XML analizzato è un'istanza di documento riferibile ad uno Schema XML.

Ogni elemento XML Schema può essere accompagnato da una documentazione che ne descrive le caratteristiche e le funzionalità (*xs:documentation*). Essa non è indispensabile, ma

agevola molto la lettura del codice da parte dei progettisti che dunque molto spesso ne prevedono la presenza.

```
<xs:documentation>Elemento tipo complesso</xs:documentation>
```

Per ulteriori approfondimenti relativi ai concetti esposti si consigliano le seguenti letture:

⇒ specifiche ufficiali del W3C:
- http://www.wA1.org/TR/xmlschema-0
- http://www.wA1.org/TR/xmlschema-1
- http://www.wA1.org/TR/xmlschema-2

⇒ alcuni testi informatici indicati nella bibliografia e facilmente reperibili in libreria o in biblioteca:
- Livingston D. (2002), "XML", Tecniche Nuove: Milano;
- Pialorsi P. (2002), "Xml: il nuovo linguaggio del web", Mondadori: Milano;

⇒ manuali tecnici in lingua inglese acquistabili sul sito www.amazon.com:
- Van Der Vlist E. (2002), "Xml Schema", O'Reilly & Associates;

1.2.2 XSL: XSLT e XPATH

Dall'analisi effettuata fino ad ora si è visto come la formattazione visiva dei dati contenuti in un documento XML abbia un'importanza secondaria, o per lo meno XML preso di per sé non consente di definire delle regole stilistiche specifiche per gli elementi. Tuttavia questo aspetto non è da sottovalutare o da dimenticare completamente; infatti potrebbe nascere l'esigenza di impaginare al meglio le informazioni, o comunque di predisporle in modo che abbiano un aspetto visivo gradevole e funzionale ai fini della lettura.

Come già accaduto in altre occasioni, vedi la nascita degli Schemi, la comunità informatica ha richiesto al W3C strumenti idonei per le specifiche esigenze. In particolare si avvertiva la necessità di disporre di meccanismi appositi per la formattazione del layout, che fossero compatibili con XML e che avessero un ruolo simile a quello assunto a suo tempo dai fogli di stile a cascata (CSS) nell'HTML.

Il W3C ha così sviluppato il linguaggio **XSL (eXtensible Style Language)** il cui scopo è quello di creare fogli di stile in XML per formattare graficamente i dati. Come XML Schema, anche XSL è un sotto-insieme proprio di XML, questo vuol dire che la sua sintassi prevede tag che descrivono elementi ed attributi. Il parser XML non è in grado di elaborare le informazioni provenienti dal nuovo linguaggio e così si è pensato di implementare un apposito **elaboratore XSL** per interpretare le istruzioni.

XSL si compone a sua volta di tre sotto-linguaggi: **XSLT, XPATH e XSL-FO**. Gran parte delle funzionalità di XSL sono comprese in XSLT che però si appoggia a XPATH per definire meglio le sue istruzioni. La causa principale di questa suddivisione del linguaggio principale in tre moduli è stata quella di fornire tempestivamente al web strumenti per l'impaginazione dei dati. Infatti il lavoro di studio e di implementazione di XSL nel suo complesso è risultato molto lungo e dettagliato e così si è preferito rilasciare le parti di codice che erano state terminate e testate. Attualmente la fase di progettazione tecnica di XSL è terminata e dunque tutti e tre i linguaggi (XSLT, XPATH e XSL-FO), sono diventati standard del web a tutti gli effetti. Si procederà di seguito ad una descrizione generale di questi tre linguaggi, con lo scopo di chiarire il loro ambito di applicazione e definire le loro caratteristiche principali, senza soffermarsi analiticamente sulla sintassi che li contraddistingue.

XPATH

Per comprendere al meglio le funzionalità offerte da XSLT occorre, seppur brevemente, introdurre il linguaggio XPATH. Il suo compito è quello di permettere all'applicazione di muoversi lungo la struttura logica di un documento XML.

XPATH è una raccomandazione del W3C e "definisce un linguaggio di query in modo specifico per consentire il filtraggio dei contenuti in un documento XML"[1]. Per linguaggio di query si intende un linguaggio attraverso il quale è possibile effettuare interrogazioni sui dati. Quindi il compito specifico del linguaggio consiste nel fornire uno strumento idoneo per individuare parti di un documento XML. In particolare è possibile selezionare uno specifico elemento o una serie di elementi che risponde ad un determinato pattern di ricerca (cioè ad un criterio di selezione). Il linguaggio consente di formulare operazioni chiamate "**espressioni**" (*Xpath expressions*) che, se applicate agli elementi del documento XML, restituiscono valori quali:

- un numero;
- una stringa di testo;
- un valore di tipo booleano (che può assumere valori "vero" o "falso");
- una raccolta di nodi.

XPATH gestisce il file XML come una struttura ad albero (la stessa vista nelle pagine precedenti), in cui ogni elemento rappresenta un nodo. I nodi riconosciuti sono del tipo:

NOME DEL NODO	DESCRIZIONE
Nodo radice *(root node)*	E' un nodo a cui non corrisponde alcun nodo genitore, ma solo nodi figli e che rappresenta il livello principale di un documento XML.
Nodo elemento *(element node)*	Identifica un classico elemento del documento XML che può avere sia nodi genitori che figli.
Nodo attributo *(attribute node)*	Sono gli attributi; devono essere necessariamente figli del nodo elemento.
Nodo di testo *(text node)*	E' una combinazione dei caratteri contenuti nei valori degli elementi; quindi comprende il testo contenuto negli elementi.
Nodi namespace	È lo spazio dei nomi *(namespace nodes)* dichiarato in XSLT
Nodi istruzione di elaborazione	Sono le istruzioni *(processing instructions nodes)* di elaborazione
Nodi commento	Sono i commenti (*comment nodes*) al documento

I nodi sono individuabili nella struttura ad albero analizzando la struttura logica del documento XML. L'elemento che contiene al suo interno un'istruzione XPATH è detto **nodo contesto** (*context-set*) e tutte le operazioni richiamate hanno come punto di riferimento questa

[1] Forlino (2003)

posizione. Da notare il fatto che le espressioni XPATH vengono dichiarate come attributi di questo nodo contesto (visto che devono riferirsi sempre e comunque ad un elemento di riferimento), mentre non è possibile inserirle in altre zone della sintassi. Dunque XPATH costruisce dei **percorsi di localizzazione (o di posizione) dell'informazione** (*location paths*), in modo da individuare particolari nodi della struttura.

Un percorso assume la forma di:

ASSE : NODE-SET

Dove ASSE indica la relazione tra il nodo contesto (punto di partenza) e il nodo da ricercare, mentre NODE-SET è un'espressione che seleziona un nodo o un insieme di nodi. I percorsi di posizione sono lo strumento attraverso il quale si procede a generare espressioni XPATH. In tal modo è possibile accedere ai nodi, individuandoli univocamente in base alla loro posizione relativa nella struttura ad albero del documento.

Esistono poi ulteriori criteri per specificare con maggiore dettaglio i parametri di ricerca degli elementi. Ad esempio si possono usare i **predicati** *(predicates)*, che consentono di applicare dei filtri per affinare la ricerca (ad esempio condizioni logiche da rispettare), gli **operatori** (*operators*), che aiutano a formalizzare l'espressione XPATH e gli **assi** (*axes*) che permettono di determinare la relazione tra il nodo test ed il nodo corrente. Concludendo la rassegna generale delle proprietà di XPATH si ricordano le **funzioni** (*Xpath functions*) che possono essere integrate ai predicati al fine di stabilire quali elementi devono essere selezionati. Esistono dunque funzioni relative ai nodi, alle stringhe, funzioni di tipo booleano e funzioni numeriche.

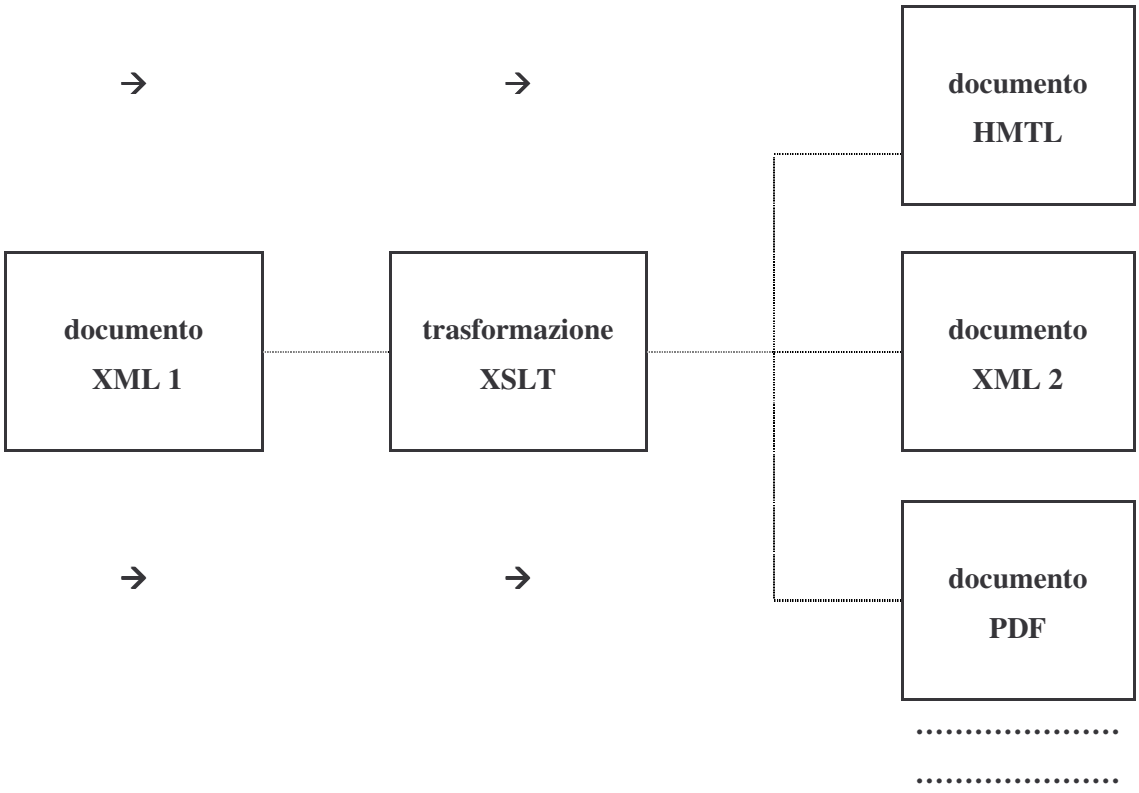
Per ulteriori approfondimenti relativi ai concetti esposti si consigliano le seguenti letture:

- ⇒ specifica ufficiale del W3C:
  - http://www.wA1.org/TR/1999/REC-xpath-19991116
- ⇒ alcuni testi informatici indicati nella bibliografia e facilmente reperibili in libreria o in biblioteca:
  - Forlino G. (2003), "Database & XML", Edizioni Master: Milano;
  - Livingston D. (2002), "XML", Tecniche Nuove: Milano;
  - Pialorsi P. (2002), "Xml: il nuovo linguaggio del web", Mondadori: Milano;
- ⇒ manuali tecnici in lingua inglese acquistabili sul sito www.amazon.com:
  - Watt (2002), "Xpath Essentials", John Wiley & Sons;
  - Holzner S. (2003), "Xpath kick start: Navigating XML with Xpath 1.0 and 2.0", SAMS;

XSLT

Il termine XSLT è l'acronimo di **eXtensible Stylesheet Language Transformation** ed è un linguaggio sviluppato dal W3C con l'intento di fornire ai progettisti gli strumenti idonei per consentire di modellare la presentazione grafica dei dati contenuti in un documento XML. La sintassi del linguaggio è ricavata direttamente dalle tradizionali specifiche XML e quindi lo sviluppatore non necessita di ulteriori conoscenze, se non un'integrazione dei contenuti appresi in XPATH. XSLT è un valido metodo per gestire il layout di un documento, ma il suo compito principale è quello di essere uno strumento standard universalmente riconosciuto. Quindi la sua funzione caratteristica è quella di appoggiare il lavoro di XML. Ad esempio si potrebbe costruire un file XML (che contiene i dati strutturati gerarchicamente) e modificarne la grafica semplicemente aggiustando le specifiche di trasformazione mediante l'uso di un foglio di stile scritto in XSLT. Al linguaggio è chiesto di trasformare il lavoro progettato in codice XML in un altro formato, generando documenti diversi a seconda delle necessità (ad esempio in formato HTML, PDF, ecc…). In particolare i tag di XSLT indicano dei comandi

all'applicazione che gestisce il documento XML relativi alle modalità di trattamento dei dati contenuti in particolari occorrenze di tag XML.



Esempio di trasformazione XSLT

Funzionamento di XSLT

Le istruzioni contenute in un file XSLT sono eseguite da un **parser XSLT** apposito che inizia la sua attività ricavando dal documento XML di partenza la struttura logica, cioè disegnando il classico albero d'origine. Questa operazione è effettuata sfruttando le funzionalità di XPATH descritte precedentemente.

Successivamente il parser XSLT individua le istruzioni XSLT contenute nel file XSLT e che sono relative al documento XML. Le istruzioni sono chiamate "**regole modello**" (*templates rules*) e i nodi sono individuati tramite espressioni XPATH. Ad ogni nodo viene associata una regola modello, cioè una serie di comandi che saranno applicati allo specifico nodo selezionato.

Il parser XSLT procede poi a costruire un ulteriore albero d'origine in base alle indicazioni contenute nelle istruzioni, generando così un nuovo file XML (o PDF, o HMTL, ecc…), che ha le caratteristiche desiderate.

Si può notare che linguaggio di trasformazione si compone di due parti principali: un linguaggio di trasformazione vero e proprio ed una specifica di formattazione dell'oggetto. Queste due componenti si integrano a vicenda e si combinano l'una con l'altra. Le operazioni relative trasformazioni XSLT sono identificate nel codice tramite l'uso di un namespace chiamato xsl, anche se non si tratta propriamente del linguaggio XSL.

Comandi fondamentali di XSLT

Le istruzioni: le regole modello contengono istruzioni (*XSLT instructions*) il cui compito è quello di prelevare specifiche informazioni dal documento XML di origine e di copiarle nel documento di destinazione, consentendo di aggiungere nuovi elementi e nuovi attributi. Le principali istruzioni sono:

ISTRUZIONE	DESCRIZIONE
<xsl:template>	E' un'istruzione che definisce una regola modello e che permette di ottenere un output dall'applicazione di un particolare template. E' un elemento che compare al livello più alto della gerarchia degli elementi XSLT.
<xsl:apply-templates>	Questa istruzione richiama la regola modello che si intende applicare al node-set analizzato. Quindi è un comando che richiama i modelli per i nodi selezionati.
<xsl:for-each>	L'istruzione consente di creare un ciclo iterativo. Il numero di ripetizioni del ciclo è pari al numero di nodi contenuti nel node-set corrente. In tal modo si estrapolano tutti i valori degli elementi appartenenti al node-set.
<xsl:if>	L'istruzione consente di effettuare delle condizioni logiche del tipo "se il test è verificato, allora…" e si accompagna solitamente all'istruzione xsl:otherwise che è il completamento della condizione logica (…"altrimenti…"). L'attributo test indica la condizione da verificare.
<xsl:value-of>	È un'istruzione attraverso la quale è possibile estrarre una stringa dal documento originario. In particolare il comando restituisce il valore contenuto nel nodo selezionato.
<xls:sort>	E' un'istruzione attraverso la quale si consente di ordinare i nodi estrapolati precedentemente dall'albero d'origine. E' un comando che va preceduto necessariamente dall'istruzione xsl:apply-templates.
<xsl:variable>	E' un'istruzione che crea una variabile dal valore di un nodo, cui è possibile accedere tramite ulteriori istruzioni XSLT. Dunque il contenuto di una variabile è riutilizzabile all'interno della struttura. Esiste inoltre l'istruzione <xsl:param> che consente di dichiarare parametri nel documento XSLT. Esistono parametri globali (che ricevono dati dall'esterno) e parametri locali (che ricevono invece dati dal modello stesso). La differenza sostanziale tra una variabile ed un parametro deriva dal fatto che il parametro è abilitato a ricevere informazioni da un modello esterno, mentre la variabile non consente questa opzione.
<xsl:copy>	E' un'istruzione che copia il nodo corrente nel file destinatario.
<xsl:copy-of>	Utilizzando l'istruzione <xsl:copy-of> è possibile copiare sia il nodo corrente che gli attributi e gli elementi discendenti dal nodo corrente nel file destinatario.
<xsl:import>	E' un comando usato per importare un foglio di stile esterno da applicare al file XML originario.
<xsl:element>	Consente di creare un elemento nel documento di destinazione.
<xsl:attribute>	Consente di creare un attributo nel documento di destinazione.
<xsl:text>	Consente di creare un nodo di testo nel documento di destinazione.
<xsl:processing-instruction>	Consente di creare un'istruzione di elaborazione nel documento di destinazione.
<xsl:comment>	Consente di creare un commento al codice nel documento di destinazione.

Inoltre esistono **funzioni** da integrare nelle istruzioni XSLT con l'obiettivo di aumentarne le capacità e permettendo così di raffinare la ricerca nell'albero d'origine. Le funzioni XSLT derivano direttamente dalle funzioni XPATH. Ad esempio si ha:

FUNZIONE	DESCRIZIONE
Document()	È possibile inserire informazioni provenienti da un documento XML diverso da quello di origine.
Format-number()	Consente di formattare un numero ed è simile alla funzione round().
Current ()	Restituisce il nodo corrente.
Element-available()	Se l'elemento ricercato nel documento XSLT è presente, allora la funzione restituisce il valore "true". La stessa operazione può essere fatta prendendo a riferimento le funzioni stesse con il comando function-available().
Positon()	Restituisce la posizione ordinale del nodo contesto e se usata combinandola con particolari istruzioni consente di estrapolare i valori degli elementi appartenenti ad un elenco.
Concat ()	E' usata per unire assieme due o più stringhe.
Contains()	E' usata per verificare se all'interno del contenuto di un elemento esiste una particolare stringa di testo.

Gli elementi letterali: gli elementi letterali (*literal result elements*) sono essenzialmente elementi non riconducibili ad una sintassi XSLT, che però vengono inclusi nelle istruzioni per agevolare la lettura del documento finale da parte dell'utente.

Il modello radice: le istruzioni relative al modello radice sono le uniche che vengono eseguite automaticamente dal parser XSLT. Per tutte le altre regole modello occorre procedere ad un loro richiamo mediante l'istruzione *<xsl:apply-templates select="node-set"/>*, la quale individua nell'albero logico il nodo, o i nodi corrispondenti e applica i comandi indicati. Quando un modello viene richiamato, le istruzioni hanno effetto solamente all'interno del node-set corrente, ossia la ricerca delle informazioni avviene solamente nel gruppo di elementi indicato.

Esempio di documento XSLT

A titolo dimostrativo delle potenzialità offerte dalle trasformazioni XSLT si presenta la trasformazione del documento XML preso come esempio di riferimento nel corso del capitolo. Ai dati di partenza sono state aggiunte alcune immagini ed alcune righe di testo per rendere più gradevole l'output. Naturalmente è possibile impaginare e formattare le informazioni in molteplici modalità a seconda delle esigenze operative dell'utente e quindi l'esempio offerto è puramente descrittivo di una delle soluzioni ottenibili applicando il linguaggio XSLT. Il codice che si propone è solo una porzione del testo complessivo del file XSLT, in quanto le

istruzioni si presentano in quantità abbondante (circa sette pagine) e quindi la lettura risulterebbe troppo complessa. Il dettaglio del codice fa riferimento alle dichiarazioni iniziali e alla creazione di una tabella nella quale si elencano i libri presenti in libreria.

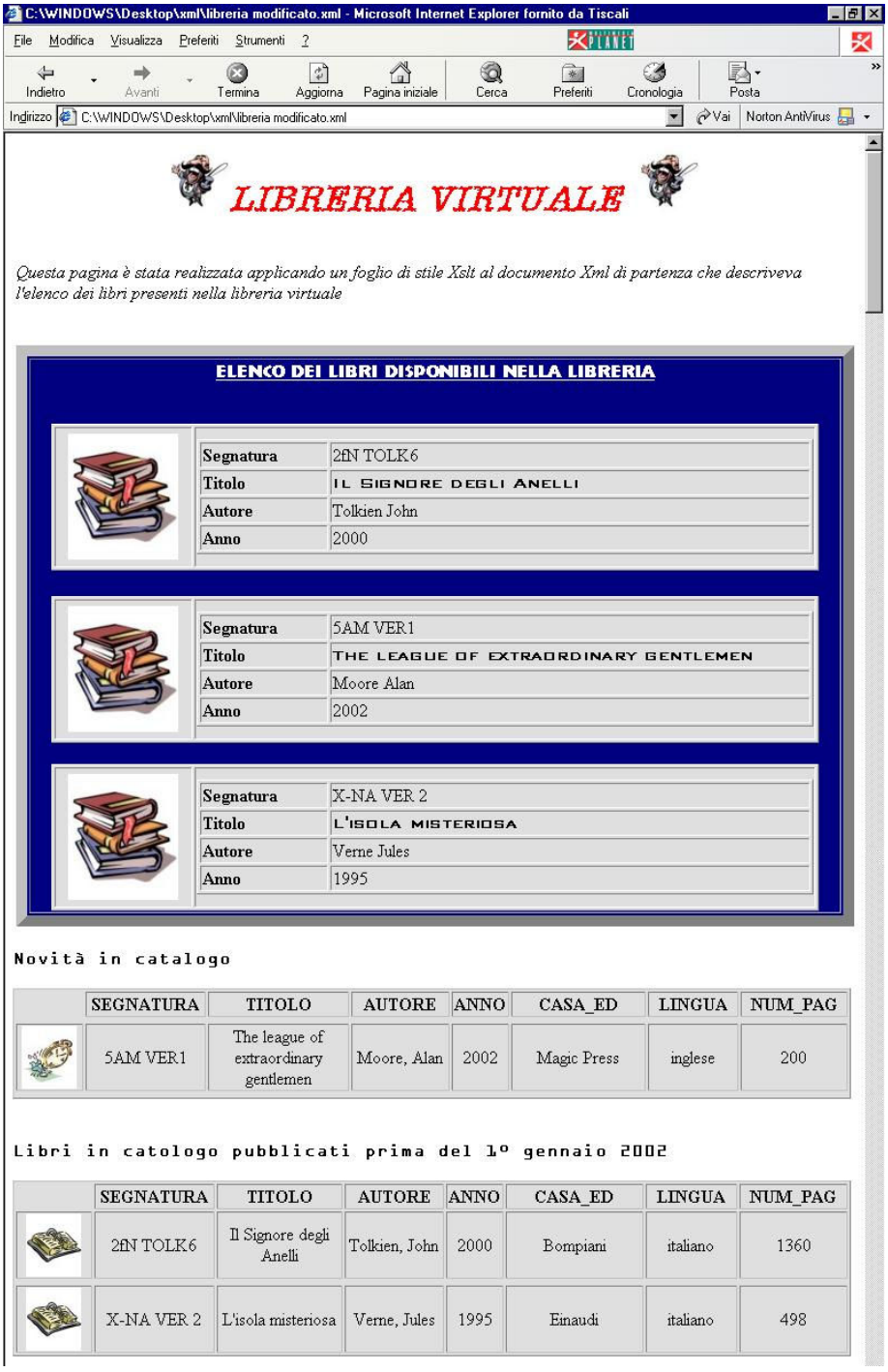
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.wA1.org/1999/XSL/Transform"
 xmlns:xsi="http://www.wA1.org/2001/XMLSchema-instance">
   <xsl:template match="/">
      <html>
         <head />
         <body>
……
……

<table bgcolor="navy" border="10">
                  <tbody>
                     <tr>
                        <td align="center" height="102" width="745">
                           <span style="color:white; font-family:Informal011 Blk BT; text-
                             decoration:underline; ">Elenco dei libri disponibili nella libreria</span>
                           <br />
                           <br />
                           <xsl:for-each select="LIBRERIA">
                              <xsl:for-each select="LIBRO">
                                 <br />
                                 <table bgcolor="#808040" border="1">
                                    <tbody>
                                       <tr>
                                          <td align="left" height="124" width="121">
                                             <img border="0">
                                                <xsl:attribute name="src"><xsl:text disable-output-
                                                  escaping="yes">file:///C:\WINDOWS\Desktop\xml\libro.jpg
                                                  </xsl:text></xsl:attribute>
                                             </img>
                                          </td>
                                          <td height="124" width="415">
                                             <table border="1">
                                                <tbody>
                                                   <tr>
                                                      <td width="112">
                                                         <span style="font-weight:bold; ">Segnatura</span>
                                                      </td>
                                                      <td width="431">
                                                         <xsl:for-each select="SEGNATURA">
                                                            <xsl:apply-templates />
                                                         </xsl:for-each>
                                                      </td>
                                                   </tr>
                                                   <tr>
                                                      <td width="112">
                                                         <span style="font-weight:bold; ">Titolo</span>
                                                      </td>
                                                      <td width="431">
                                                         <xsl:for-each select="TITOLO">
                                                            <span style="font-family:BankGothic Lt BT; font-
                                                              weight:bold; ">
                                                               <xsl:apply-templates />
                                                            </span>
                                                         </xsl:for-each>
```

```
                                        </td>
                                    </tr>
                                    <tr>
                                        <td width="112">
                                            <span style="font-weight:bold; ">Autore</span>
                                        </td>
                                        <td width="431">
                                            <xsl:for-each select="AUTORE">
                                                <xsl:for-each select="COGNOME">
                                                    <xsl:apply-templates />
                                                </xsl:for-each> <xsl:for-each
                                                 select="NOME">
                                                    <xsl:apply-templates />
                                                </xsl:for-each>
                                            </xsl:for-each>
                                        </td>
                                    </tr>
                                    <tr>
                                        <td height="1" width="112">
                                            <span style="font-weight:bold; ">Anno</span>
                                        </td>
                                        <td height="1" width="431">
                                            <xsl:for-each select="ANNO">
                                                <xsl:apply-templates />
                                            </xsl:for-each>
                                        </td>
                                    </tr>
                                </tbody>
                            </table>
                        </td>
                    </tr>
                </tbody>
            </table>
        </xsl:for-each>
    </xsl:for-each>
</td>
</tr>
</tbody>
</table>
<br />
</p>
</address>
```

Si può notare che nella prima riga del documento si indica che il file è di tipo XML, inoltre l'istruzione *<xsl : stylesheet version = "1.0" xmlns : xsl = "http: // www. wA1. org / 1999 / XSL / Transform">* che viene subito dopo, indica che tutte i comandi compresi al suo interno sono riferiti al foglio di stile. Nello specifico si crea un namespace che rappresenta l'elemento radice del foglio di stile. Successivamente si può notare l'istruzione *<xsl:template>* che definisce le regole di formattazione per l'elemento selezionato e a cui sarà applicata la regola modello. Da osservare inoltre che è indispensabile collegare il file XSLT al file XML di riferimento (il quale a sua volta è legato ad un file di XML Schema) e ciò avviene tramite una speciale istruzione di elaborazione che va inserita nel documento XML originario e che va localizzata subito dopo la dichiarazione XML (nello specifico si ha l'istruzione *<? xml:stylesheet type="text/xsl href="Nome_file"?>*.

Si presenta inoltre il risultato visivo della trasformazione. La modalità di rappresentazione del documento è quella che si ottiene usando il browser Microsoft Explorer 6.0, scaricabile gratuitamente al sito www.msd.microsoft.com.



Per ulteriori approfondimenti relativi ai concetti esposti si consigliano le seguenti letture:

⇒ specifica ufficiale del W3C:
- http://www.wA1.org/TR/1999/REC-xslt-19991116

⇒ alcuni testi informatici indicati nella bibliografia e facilmente reperibili in libreria o in biblioteca:
- Forlino G. (2003), “Database & XML”, Edizioni Master: Milano;

- Livingston D. (2002), “XML”, Tecniche Nuove: Milano;
- Pialorsi P. (2002), “Xml: il nuovo linguaggio del web”, Mondadori: Milano;

⇒ manuali tecnici in lingua inglese acquistabili sul sito www.amazon.com:

- Fitzgerald M. (2003), “Learning XSLT”, O’Reilly & Associates;
- Du Charme B. (2001), “*XSLT quickly*" , O’Reilly & Associates ;
- Mangano S. (2003), *"XSLT cookbook",* O’Reilly & Associates.

XSL:FO

XSL-FO è l’acronimo di **eXtensible Stylesheet Language Formatting Object** ed è la terza ed ultima parte che concorre a formare il linguaggio XSL.

Il linguaggio consente di gestire la stampa dei documenti XML, determinando a priori la formattazione della pagina. Quindi è possibile ad esempio stabilire la disposizione grafica delle informazioni sul foglio cartaceo, disegnare i margini e imporre la dimensione del testo. Sostanzialmente XLS-FO permette di ottenere una stampa da un comune documento XML, usato come input, generata nel formato desiderato (es. PDF). Prima di procedere all’utilizzo pratico del linguaggio sul file XML è necessario aver effettuato alcune operazioni intermedie quali:

- aver costruito un documento XML valido e ben formato;
- aver collegato il documento al relativo Schema;
- creare un foglio di stile e procedere alla trasformazione XSL tramite i comandi di XSLT e XPATH.

Non è necessario disporre di particolari software per la creazione di documenti XSL-FO; è sufficiente usare un semplice editor testuale, come del resto avviene per tutti gli altri linguaggi paralleli a XML. L’unica avvertenza è quella di disporre di un parser XSL-FO apposito.

Per ulteriori approfondimenti relativi ai concetti esposti si consigliano le seguenti letture:

⇒ specifica ufficiale del W3C:

- http://www.wA1.org/TR/2003/WD-xsl-20011015

⇒ alcuni testi informatici indicati nella bibliografia e facilmente reperibili in libreria o in biblioteca:

- Williamson H. (2002), “XML: la guida completa”, Mc Graw Hill: Milano;

⇒ manuali tecnici in lingua inglese acquistabili sul sito www.amazon.com:

- Pawson D. (2002), “XSL-FO”, O’Reilly & Associates;

1.3 *Rassegna delle principali tecnologie integrate in XML*

XLINK

Il sistema di collegamenti fra le risorse presenti in rete è senza dubbio uno dei principali fattori che ha determinato la fortuna dell’HTML e che ha consentito al World Wide Web di evolvere in modo così rapido. I collegamenti realizzati in HTML hanno fornito la base per collegare tra loro migliaia di documenti eterogenei, nonostante le limitate capacità offerte dal lato della flessibilità e dell’adattabilità al contesto. In particolare le restrizioni riscontrate nell’utilizzo di tali strumenti si riferiscono al fatto che:

1. i link costruiti in HTML consentono di puntare ad una sola risorsa per volta;
2. l’attraversamento è unidirezionale;

3. il progettista non è in grado di imporre caratteristiche univoche al link a seconda della risorsa a cui punta (quindi un collegamento ad un file di testo è costruito allo stesso modo di un collegamento che indirizza ad un'immagine);
4. non si ha la possibilità di puntare a singole porzioni del documento di destinazione, ma soltanto a posizioni specifiche del documento.

I progettisti XML, partendo dall'analisi dei limiti riscontrati nelle soluzioni precedenti, hanno sviluppato un nuovo linguaggio chiamato **XML LINKING LANGUAGE (XLINK)** che consente di gestire in modo ottimale le interconnessioni tra le risorse disponibili. La funzione di XMLINK è quella di definire le relazioni esistenti tra due o più risorse, permettendo all'utente di gestire il collegamento in un luogo separato rispetto alle risorse e di definire con maggiore attenzione il comportamento che ogni elemento deve assumere. Analogamente agli altri linguaggi derivati da XML, anche XMLINK necessita di un parser apposito (non ancora presente in molti casi, nei moduli del browser) che consente l'interpretazione del codice da parte della macchina. Attraverso l'applicazione di XMLINK è possibile progettare link multidirezionali, che puntano ad una pluralità di risorse e che sono costruiti ad hoc tenendo conto delle caratteristiche degli elementi da collegare. Lo standard Xlink non ha lo stesso grado di maturità e diffusione di quelli in precedenza illustrati. Esso riveste peraltro una funzione centrale nelle applicazioni basate su XBRL, considerate nelle parti successive.

Ogni collegamento XMLINK fa uso di un namespace ("*xlink*:") che viene utilizzato esclusivamente sugli attributi dei vari elementi del linguaggio.

Al fine di comprendere al meglio i concetti illustrati nel corso del paragrafo è opportuno fornire una panoramica delle definizioni relative ad i termini usati. Le definizioni sono estratte dal libro "XML" di Dan Livingston (2002) edito da Tecniche Nuove, Milano.

CONCETTO	DEFINZIONE
Risorsa	"Una risorsa tecnicamente è ogni tipo di informazione o servizio a cui è associato un indirizzo." Esistono dunque **risorse locali**, ossia interne al documento e **risorse remote**, cioè localizzate al di fuori del documento analizzato
Collegamento	"Un collegamento è una relazione esplicita tra risorse e parti di risorse."
Elemento XLINK	"Un elemento XLINK è un elemento XML i cui attributi sono conformi alle specifiche XLINK e che dichiarano l'esistenza di un collegamento."
Attraversare un collegamento	"Attraversare un collegamento significa seguire o utilizzare il collegamento stesso per una qualsiasi ragione. L'attraversamento avviene tra due risorse."

In XLINK esistono due modalità di definire un collegamento: il collegamento semplice ed il collegamento esteso. XML non elimina completamente le funzionalità offerte dal tag di collegamento "*<a>*" tipico di HTML, ma piuttosto le incorpora, mantenendone la struttura di base.

Collegamenti semplici in XLINK

I **collegamenti semplici** *(simple link)* costruiti in XLINK permettono di connettere due risorse distinte, chiamate rispettivamente risorsa locale di partenza e risorsa remota di destinazione. Da notare il fatto che i collegamenti semplici non possono interconnettere più di due risorse e non sono in grado di supportare modalità di collegamento che prevedono attraversamenti da una risorsa remota ad una locale o tra due risorse remote. Sostanzialmente i collegamenti semplici di XLINK sono simili a quelli di HTML visto che puntano a risorse singole e sono unidirezionali.

La dichiarazione di collegamento semplice prevede la sintassi:

```
<nome_link_semplice
      xmlns:xlink=”http://www.wA1.org/1999/xlink”
      xlink:type=”simple”
      xlink:href=”risorsa_destinazione”
      xlink:actuate=”onRequest|onLoad|embed|none”
      xlink:title=”Descrizione_collegamento”
      xlink:role=”Riferimento_documento_descrittivo”  />
```

Attraverso il comando *“xmlns:xlink”* si definisce lo spazio dei nomi relativo ai comandi di XLINK. Osservando il codice è possibile notare la presenza di alcuni attributi che concorrono a definire le caratteristiche del collegamento. Gli attributi principali presenti in questo tipo di collegamento sono:

ATTRIBUTO	DESCRIZIONE
type	è un attributo obbligatorio che identifica il tipo di elemento XLINK usato. E’ dunque utilizzato per definire se il collegamento è di tipo semplice (simple) o di tipo esteso (extended).
href	identifica l’indirizzo (URI) della risorsa a cui il link punta. Con il termine URI si intende “Uniform Resource Identifier”, cioè un metodo standard usato per localizzare le risorse nel web.
show	tramite questo attributo è possibile specificare la posizione in cui viene collocata la risorsa di destinazione nel momento in cui il link è attivato. Quindi a seconda dei valori assunti dall’attributo è possibile visualizzare la risorsa di destinazione in una nuova finestra del browser (new), sostituire la risorsa di partenza con quella di destinazione (replace), effettuare la stessa operazione di sostituzione escludendo dalle modifiche il resto del documento (embed) o lasciare libero il parser di agire autonomamente (none).
actuate	tramite la definizione dell’attributo “actuate” è possibile stabilire le modalità di attraversamento del collegamento e quindi definire il momento in cui il link è attivato. L’attivazione può avvenire dopo la richiesta dell’utente (onRequest), non appena viene caricata la risorsa di partenza (onLoad), oppure si può lasciare libero il parser di agire autonomamente (none).
title	è un attributo che definisce una stringa in cui è contenuta una descrizione utile all’uomo per comprendere il significato dell’elemento XLINK utilizzato.

ATTRIBUTO	DESCRIZIONE
role	indica un documento in cui è contenuta la descrizione della risorsa a cui ci si collega. E' simile all'attributo "title" ma consente l'uso di maggiore spazio per il commento.
arcrole	è un attributo usato per collegare un documento ad un insieme di link (linkbase).
label	label è un attributo che può assumere come valori esclusivamente nomi validi per XML (a tale proposito si rinvia al paragrafo in cui si definisce la sintassi di XML) e che consente di fare riferimento ad elementi di tipo risorsa ed elementi di tipo locator.

Ogni attributo va dichiarato all'interno di un elemento XLINK e deve essere preceduto dal prefisso *"xlink:"* che ne specifica l'appartenenza al namespace. Da notare che ogni elemento del documento XML può diventare un collegamento XLINK se al suo interno è presente l'attributo *"xlink:type"*.

Collegamenti estesi in XLINK

Un **collegamento XLINK di tipo esteso** *(extended link)* è un insieme di comandi che consente di interconnettere più risorse, sia di tipo locale che di tipo remoto. A differenza dei link semplici, i link estesi sono multidirezionali e quindi non ha più senso parlare di risorsa di destinazione e di risorsa di partenza. Inoltre sono collegamenti multipli nel senso che è possibile connettere un numero qualsiasi di risorse tramite un unico link.

La dichiarazione iniziale del collegamento esteso è simile a quella dei collegamenti semplici:

```
<nome_collegamento_esteso
      xmlns:xlink="http://www.wA1.org/TR/1999/xlink"
      xlink:type="extended"  />
```

E' possibile inoltre usare alcuni degli attributi descritti precedentemete quali *"xlink:href", "xlink:title", "xlink:role"* e *"xlink:label"*..

La differenza sostanziale dei link estesi rispetto ai link semplici si riscontra nella parte successiva del codice ed è riconducibile alla presenza di attributi, relativi al collegamento esteso, inseriti in elementi appartenenti al documento XML diversi da quello in cui è contenuta la dichiarazione del link. In dettaglio si possono avere:

- **Elementi di tipo risorsa**: gli elementi di tipo "risorsa" (*resource element*) sono elementi che identificano una risorsa locale, cioè presente nel documento XML. Gli unici attributi che sono obbligatori in questo contesto sono "*xlink:type"* e *"xlink:label"* . Un esempio di elemento risorsa è:

```
<nome_risorsa_locale
xlink:type="resource"
xlink:label="risorsa_locale"  />
```

- **Elementi di tipo locator:** un elemento di tipo "locator" individua una risorsa remota che è coinvolta nel collegamento XLINK. Gli attributi indispensabili per una sua dichiarazione sono *"xlink:type", "xlink:href"* e "*xlink:label"*. Quindi si può dire che gli elementi di tipo "risorsa" specificano le risorse locali, mentre gli elementi di tipo "locator" specificano le risorse remote (cioè esterne al documento XML). Tuttavia entrambi gli elementi si limitano ad indicare le risorse, senza definire le relazioni tra le stesse. Un esempio di elemento locator è:

```
<nome_risorsa_remota
xlink:type="locator"
xlink:href="indirizzo_uri_risorsa"
xlink:label="risorsa_remota"  />
```

- **Elementi di tipo arco:** l'elemento di tipo "arco" (*arc element*) consente di definire le relazioni esistenti tra le risorse selezionate precedentemente attraverso gli elementi "risorsa" e "locator". Un elemento di tipo "arco" stabilisce quali risorse connettere e impone le direzioni dell'attraversamento, definendo in sostanza quale sia la risorsa di partenza (*xlink:to*) e quale sia quella di destinazione (*xlink:from*). Gli attributi indispensabili per la costruzione dell'elemento sono *"xlink:type", "xlink:to"* e *"xlink:from"*. Alcuni esempi di elementi arco sono:

```
<nome_arco
xlink:type="arc"
xlink:from="risorsa_locale"
xlink:to="risorsa_remota"
xlink:show="replace"
xlink:actuate="onRequest" />
```

```
<nome_arco
xlink:type="arc"
xlink:from="risorsa_remota"
xlink:to="risorsa_locale"
xlink:show="replace"
xlink:actuate="onRequest" />
```

Gli elementi di tipo "arco" non ammettono l'uso dell'attributo *"xlink:role"*. Per questo particolare tipo di elemento XLINK è stato progettato l'attributo *"xlink:arcrole"*, il cui contenuto deve essere necessariamente un URI. L'indirizzo internet indicato nell'attributo punta ad una risorsa in cui è contenuta la descrizione della relazione esistente fra le risorse coinvolte nel collegamento. Quindi in sintesi attraverso l'uso dell'attributo *"xlink:arcrole"* si rinvia ad un documento esterno in cui si descrive il tipo di relazione esistente tra le risorse relazionate dall'elemento "arco" (ad es. relazione padre-figlio).

Da notare il fatto che l'attributo *"xlink:arcrole"* non è ritenuto obbligatorio nella sintassi XLINK.

I linkbase

Un ulteriore concetto molto importante nell'ambito di XLINK è quello di **linkbase** che definisce un documento contenente una pluralità di collegamenti. I link inclusi in questo file sono esclusivamente di due tipi: link verso l'interno (cioè collegamenti che connettono una risorsa remota con una locale) e link fra terze parti (cioè collegamenti che connettono tra loro risorse remote). Quindi si offre la possibilità di costruire link tra documenti su cui l'autore del linkbase non ha un controllo diretto.

In sintesi un linkbase è un documento/database separato dal documento XML iniziale e nel quale sono stati dichiarati link di tipo esteso. L'obiettivo di questo file è quello di permettere la modifica dei collegamenti evitando di manipolare direttamente il documento XML

contenente i dati veri e propri. In questi casi occorre fare uso dell'attributo *"xlink:arcrole"* nel documento XML per specificare il fatto che si sta puntando ad un linkbase.

La sintassi corretta prevede l'inserimento nell'attributo del codice *"arcrole=http:// www.wA1.org/ 1999/xlink/properties/linkbase"* all'interno di un elemento "arco", il cui compito è indicare al parser XLINK di puntare direttamente ad un collegamento inserito un linknbase specifico.

Trattandosi di una specifica complessa e articolata si preferisce rinviare la presentazione di esempi alle applicazioni relative a XBRL descritte nel capitolo successivo.

Per ulteriori approfondimenti relativi ai concetti esposti si consigliano le seguenti letture:

⇒ specifica ufficiale del W3C:
  - http://www.wA1.org/TR/2000/REC-xlink-20010627

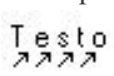
⇒ alcuni testi informatici indicati nella bibliografia e facilmente reperibili in libreria o in biblioteca:
  - Livingston D. (2002), "XML", Tecniche Nuove: Milano;
  - Pialorsi P. (2002), "Xml: il nuovo linguaggio del web", Mondadori: Milano;
  - Williamson H. (2002), "XML: la guida completa", Mc Graw Hill: Milano;

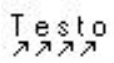
⇒ manuali tecnici in lingua inglese acquistabili sul sito www.amazon.com:
  - Watt A.H. (2002), "Xlink Essentials", John Wiley & Sons;

XPOINTER

Nella descrizione fornita a proposito di XLINK si è parlato della possibilità offerta dal linguaggio di puntare a singole porzioni del documento XML. Questa affermazione non è del tutto corretta in quanto XLINK di per sé non consente di focalizzare l'attenzione su singoli elementi del file XML. A questo proposito il W3C ha ideato un'ulteriore tecnologia chiamata **XML Pointing Language (XPOINTER)** attraverso la quale è possibile integrare le funzionalità di XLINK in questo ambito operativo.

Il linguaggio XPOINTER sostanzialmente è un'estensione di XPATH in XLINK, in quanto permette di analizzare la struttura logica del documento XML. Gli strumenti forniti consentono di analizzare con maggiore precisione il contenuto del file XML, superando i vincoli logici imposti dai concetti di nodo e di elemento XML. Tramite l'uso di XPOINTER è possibile infatti rilevare sezioni del documento XML che non rientrano nelle definizioni usuali ed in particolare si ha:

PORZIONE DI CODICE ANALIZZATA	DESCRIZIONE
Punto	Si analizzano i punti (*points*) esistenti tra i singoli caratteri del contenuto di un elemento. E' un concetto piuttosto astratto visto che i punti non hanno una larghezza vera e propria dal momento che si localizzano fra due caratteri privi di spazi. Ad esempio: 
Intervallo	Un intervallo (*range*) è un insieme di caratteri che sono racchiusi tra un punto di partenza ed un punto di arrivo. Un intervallo può includere una singola informazione, un nodo o altre porzioni del documento.

PORZIONE DI CODICE ANALIZZATA	DESCRIZIONE
Punto	Si analizzano i punti (*points*) esistenti tra i singoli caratteri del contenuto di un elemento. E' un concetto piuttosto astratto visto che i punti non hanno una larghezza vera e propria dal momento che si localizzano fra due caratteri privi di spazi. Ad esempio: 
Intervallo	Un intervallo (*range*) è un insieme di caratteri che sono racchiusi tra un punto di partenza ed un punto di arrivo. Un intervallo può includere una singola informazione, un nodo o altre porzioni del documento.
Frammento	Un frammento (*fragment*) identifica una porzione di codice del documento XML che è scritta in sequenza. Questa combinazione di codice non deve necessariamente includere completamente un elemento o un nodo della struttura ad albero.

La sintassi del linguaggio XPOINTER assume la forma:

```
#xpointer(espressione)
```

Le espressioni valide sono quelle relative ad XPATH (per una loro descrizione si rinvia al paragrafo inerente la descrizione di XPATH) a cui però se ne aggiungono altre particolari del tipo:

FUNZIONE	DESCRIZIONE
Start-point	Seleziona il primo punto di un intervallo analizzato.
End-point	Seleziona l'ultimo punto di un intervallo analizzato.
String-range	Seleziona una stringa da un intervallo analizzato.
Range	Seleziona un intervallo analizzato.
Range-inside	È simile alla funzione range; si differenzia nel caso in cui l'intervallo analizzato sia composto da un solo elemento e in questo caso fornisce come output il suo contenuto.
Range-to	Permette di creare un intervallo stabilendo un punto di partenza ed uno di arrivo.
Origin	Consente l'uso di XPOINTER per stabilire le posizioni relative delle risorse, nel caso in cui esse non risiedano all'interno dello stesso file (es. risorse remote).

FUNZIONE	DESCRIZIONE
Start-point	Seleziona il primo punto di un intervallo analizzato.
End-point	Seleziona l'ultimo punto di un intervallo analizzato.
Here	Seleziona il nodo del documento in cui è inclusa l'espressione XPOINTER.

Per ulteriori approfondimenti relativi ai concetti esposti si consigliano le seguenti letture:

- ⇒ specifica ufficiale del W3C:
  - http://www.wA1.org/TR/2001/WD-xptr-20010108
- ⇒ alcuni testi informatici indicati nella bibliografia e facilmente reperibili in libreria o in biblioteca:
  - Livingston D. (2002), "XML", Tecniche Nuove: Milano
  - Pialorsi P. (2002), "Xml: il nuovo linguaggio del web", Mondadori: Milano;
  - Williamson H. (2002), "XML: la guida completa", Mc Graw Hill: Milano
- ⇒ manuali tecnici in lingua inglese acquistabili sul sito www.amazon.com:
  - Simpson J.E. (2002), "Xpath and Xpointer", O'Reilly & Associates.

XQUERY

Dall'analisi effettuata nel corso del capitolo si è visto come XPATH sia una tecnologia attraverso la quale è possibile effettuare interrogazioni sui dati contenuti in un documento XML. Tuttavia XPATH non è l'unico linguaggio esistente che opera in questo ambito, ma esistono soluzioni alternative, quali XML QUERY o SQLXML, che svolgono sostanzialmente le stesse funzioni.

XML QUERY (Xquery) è un linguaggio creato per il recupero delle informazioni contenute in documenti XML ed il suo uso si rivela particolarmente efficiente nel caso in cui si abbia la necessità di analizzare grandi quantitativi di dati. XML QUERY pur non essendo una soluzione basata su XML risulta essere facilmente manipolabile e flessibile.

Dal punto di vista tecnico XQUERY si presenta come una serie di specifiche pubblicate nel febbraio 2001 e che attualmente hanno raggiunto solamente lo stadio di "bozza di lavoro". Quindi lo sviluppo del progetto non è stato ancora ultimato completamente e questo implica che potrebbe subire delle modifiche nel corso del tempo.

Le interrogazioni sui dati consentite dal linguaggio possono essere effettuate su uno specifico documento XML, su una porzione di esso (es. su un gruppo di nodi) o su una serie di documenti XML eterogenei. L'obiettivo di queste operazioni è quello di selezionare e filtrare i dati in base a dei parametri di ricerca. Una **query** (interrogazione) è formata da un'espressione che analizza il documento XML e restituisce come risultato una sequenza di nodi o dei valori specifici. Da notare il fatto che la tecnologia XQUERY si appoggia a XPATH nell'operazione di analisi della struttura logica del documento. Quindi riassumendo il processo si può dire che le informazioni sono individuate da XQUERY analizzando la struttura ad albero del documento XML e che, una volta trovate, esse vengono copiate in un nuovo file XML. Il fine del W3C è quello di rendere disponibile ai progettisti XML uno strumento con funzionalità simili a quelle che il linguaggio SQL offre nei confronti dei database relazionali.

Il linguaggio XML QUERY è composto attualmente da **quattro bozze di lavoro**, ognuna delle quali svolge compiti ben precisi. Nello specifico si individua:

1. **XML QUERY REQUIREMENTS:** è una specifica nella quale sono elencati i requisiti necessari che una query deve possedere per essere ritenuta valida. Ad esempio un vincolo imposto è che l'interrogazione sia scritta in una sintassi XML valida. L'elenco completo dei requisiti è piuttosto complesso e dettagliato ed è disponibile all'indirizzo http://www.wA1.org/TR/Xmlquery-req. Gli obiettivi che si intendono raggiungere mediante la definizione di questa specifica sono:
   - definire un modello per i dati per i documenti XML;
   - produrre un insieme di operatori di query per il modello dei dati;
   - ottenere un linguaggio di query basato su questi dati.[2]

2. **XML QUERY DATA MODEL:** la specifica completa è disponibile al sito http://www.wA1.org/TR/query-datamodel. Tramite l'uso di questa documentazione è possibile descrivere analiticamente le modalità di creazione dei dati in XML QUERY e le modalità di accesso agli stessi. Quindi in ultima analisi si identificano le categorie di dati e di informazioni accessibili mediante l'uso del linguaggio.

3. **XML QUERY ALGEBRA:** la specifica è disponibile al sito http://www.wA1.org/TR/query-algebra ed il suo compito è quello di "fornire una base formale per il linguaggio di query XML"[3]. In sostanza il linguaggio consente operazioni del tipo:

FUNZIONE	DESCRIZIONE
proiezione	È una funzione che svolge la ricerca di informazioni nel documento tramite l'uso delle query.
iterazione	Consiste nella selezione di elementi dal documento XML e nella riformattazione degli stessi in un formato diverso da quello di partenza.
selezione	Implica l'individuazione di elementi nel documento XML selezionandoli in base al valore assunto (ad esempio ricerca di tutti gli elementi con valore maggiore ad un numero stabilito).
quantificazione	Consiste nell'individuazione degli elementi che presentano un valore preciso all'interno del documento (ad esempio quelli che hanno un valore pari ad un numero fisso).
unione	Implica l'unione di due o più valori provenienti da uno o più documenti XML.
ordinamento	E' una funzione che svolge la classica operazione di ordinamento dei valori presenti in un documento XML in base ad un parametro chiave (es. ordinare gli elementi in modo crescente utilizzando come base l'identificatore univoco ID dell'elemento).

[2] H. Williamson (2002)
[3] H. Williamson (2002)

XML Query Algebra consente di estrarre stringhe, valori interi o valori di tipo booleano e permette inoltre di fare un confronto fra i risultati ottenuti dalla ricerca e i dati forniti come input. L'operazione di controllo è effettuata attraverso l'analisi degli Schemi XML e mediante un confronto della conformità del formato dei dati in entrata con quello dei dati in uscita. In tal modo si evitano eventuali errori di selezione delle informazioni. Quindi se la query ha come input di ricerca una stringa, il risultato dell'estrazione deve essere una stringa (e non ad esempio un valore numerico).

Da ricordare inoltre che il gruppo di lavoro del W3C sta attualmente implementando un linguaggio chiamato XQUERYX, ossia un linguaggio con funzionalità identiche a XQUERY, ma che è basato su una sintassi XML.

Per ulteriori approfondimenti relativi ai concetti esposti si consigliano le seguenti letture:

- ⇒ specifica ufficiale del W3C:
  - http://www.wA1.org/TR/2003/WD-xquery-20031112
- ⇒ alcuni testi informatici indicati nella bibliografia e facilmente reperibili in libreria o in biblioteca:
  - Williamson H. (2002), "XML: la guida completa", Mc Graw Hill: Milano
- ⇒ manuali tecnici in lingua inglese acquistabili sul sito www.amazon.com:
  - Katz, Chamberlin, Droper, Fernandez, Kay, Robie, Rys, Simeon, Tivy, Wadler (2003), "Xquery from the experts: a guide to the W3C XML query language", Addison-Wesley Pub Co.

1.3.2 XML e la protezione dei dati

I documenti XML possono contenere qualsiasi tipo di dati. In determinate circostanze è possibile che le informazioni incluse nei file abbiano carattere riservato e personale. Basta pensare ai dati medici relativi ad un paziente che sono conservati in una struttura ospedaliera. In altri casi le informazioni potrebbero richiedere livelli di sicurezza aggiuntivi, dal momento che potrebbero riferirsi ad esempio a transazioni finanziarie o commerciali.

XML non offre molte garanzie sotto questo aspetto. Infatti il linguaggio è stato creato in modo che il suo codice fosse completamente trasparente e facilmente manipolabile. Questo implica che se non si adottano strumenti informatici di difesa, XML, come ogni altra tecnologia, è potenzialmente vulnerabile da attacchi di hacker, virus o dalla semplice intrusione di curiosi nei documenti. Occorre adottare dei provvedimenti per mascherare i dati durante la trasmissione dei file XML ed una volta recapitati, consentire un loro accesso mediante viste personalizzate, a seconda dell'utente che le legge.

La prima forma di difesa è quella di conservare gli archivi in zone sicure. Successivamente si può ricorrere ad altri strumenti quali i router. I **router** sono dispositivi hardware che consentono di reindirizzare i dati dal web e di trasferirli in una rete informatica locale, come se fossero stati creati internamente. Quindi si evita che un certo tipo di traffico informatico indesiderato attraversi il sistema. Lo stesso procedimento di difesa avviene per le richieste in uscita dalla rete. Quindi sostanzialmente il router filtra i dati, selezionando gli accessi in base all'origine del dato stesso o in base alla porta di destinazione.

E' possibile adottare ulteriori meccanismi, come i **firewall di protezione**, cioè combinazioni di soluzioni hardware e software che consentono un accesso controllato ai dati e limitato ad un gruppo di persone selezionate. Altri metodi utilizzati sono la creazione di **password univoche** o la **crittografia**, usata per nascondere il significato reale dei dati durante la loro trasmissione.

Un esempio reale di problemi inerenti la sicurezza delle informazioni contenute in documenti XML è quello che recentemente ha interessato Microsoft. In particolare nell'ultima versione rilasciata di Microsoft Office, il produttore ha previsto l'opzione di salvataggio dei documenti

creati in formato XML. Questa è senza dubbio un'importante aggiunta che va ad integrarsi con le molteplici funzionalità offerte. Tuttavia si prospetta il verificarsi di problemi legati alle possibili infiltrazioni di virus nei file, soprattutto quelli che rientrano nella categoria di virus delle macro. Le difficoltà nel garantire la sicurezza dipendono dal fatto che i documenti XML non rispettano una localizzazione precisa dei dati nel testo. In dettaglio si nota che in un documento XML, una volta che i dati sono stati contrassegnati con la necessaria cura, l'ordine con il quale compaiono nel codice non ha importanza, mentre i tradizionali documenti Office posizionano il testo in specifici punti del file.

La carenza strutturale dell'applicazione è stata fatta notare al produttore da numerosi esperti. Tuttavia Microsoft ha replicato che il problema non è relativo alla progettazione del software, ma è riconducibile a XML stesso, scaricando in definitiva il problema al W3C.

Come si è potuto capire da questo esempio concreto, XML è attualmente uno standard in continua crescita e che non si può considerare ancora maturo per ogni ambito di applicazione. Si auspica tuttavia che con il tempo queste imperfezioni vengano rimosse, garantendo una sicurezza maggiore sui dati trasmessi.