UNIVERSITAT OBERTA DE CATALUNYA

Enginyeria superior en informàtica

XML / XQueryLanguages: Estat de l'Art.

Alumne: Eva Vilaginés Aguilà
Dirigit per: Alex Alfonso Minguillón

CURS: Primer quatrimestre 2003-2004

RESUM:

La present memòria pretén introduir al lector dins del camp de l'XML, fent referència al propi llenguatge XML així com a l'anomenada família XML, formada per tots aquelles tecnologies i estàndards que estan relacionats amb l'XML i que fan de l'XML un llenguatge realment potent i versàtil.

Per fer aquesta introducció a XML, s'ha recopilat informació procedent de diverses fonts, principalment del W3C (*World Wide Web Consortium*), organisme que promou i contribueix activament en l'elaboració dels estàndards i en l'evolució de l'XML. Posteriorment, s'ha procedit a fer una estructuració i classificació de tota aquesta informació, per tal de tenir clars quins són els aspectes més importants de l'XML i dels diferents estàndards. També, s'ha procedit a estudiar en més detall el llenguatge de consulta XQuery, així com veure quin és el tractament de l'XML dins el camp de les bases de dades.

Així doncs, en la present memòria podem diferenciar tres grans blocs temàtics:

1. L'XML i els seus estàndards.
2. XQuery: el llenguatge de consulta XML.
3. XML i les bases de dades.

En el primer bloc es descriu el llenguatge XML i els diferents estàndards i tecnologies que hi estan relacionats. En aquest bloc veurem entre d'altres aspectes: quin és l'origen i l'evolució de l'XML, com s'estructuren les dades d'un document XML, com mitjançant DTD i esquemes XML es defineix la gramàtica d'un document XML, etc. També, descriurem les tecnologies o estàndards que permeten definir la presentació, visualització i transformació dels documents XML. Ens referim a les fulles d'estils en cascada (CSS) i al llenguatge d'estils extensible (XSL), el qual consta d'un llenguatge de transformació (XSLT) i d'un llenguatge de formateig (XSL-FO) que descriu de forma precisa l'aparença de les dades d'un document XML.

També dins d'aquest primer bloc, parlarem de les tecnologies de navegació i accés que permeten navegar entre diferents documents XML i navegar dins l'estructura jeràrquica que constitueixen els documents XML. Concretament, veurem els llenguatges: XLink, que permet la creació d'enllaços entre dos o més documents XML; XPointer, que permet identificar i fer referències a punts o fragments que es troben dins d'un document XML i per últim XPath, que permet seleccionar i fer referències a qualsevol tipus d'informació que es trobi dins d'un document XML.

Per finalitzar aquest primer bloc, veurem una breu introducció a les tecnologies de suport per l'anàlisi i manipulació dels documents XML: DOM, interfície de programació orientada a objectes que crea una representació de l'estructura del document en memòria i SAX, interfície basada en events que permet processar els documents XML d'una forma més simple i ràpida.

El segon bloc de la memòria és una introducció al llenguatge XQuery. XQuery és el llenguatge de consulta més potent i robust que s'ha desenvolupat per poder interrogar, recuperar i interpretar la informació que contenen els documents XML. Així doncs, aquest bloc pretén ser una introducció a XQuery, descrivint el model de dades en que es basa i veient la sintaxi XQuery, així com les principals funcions i operadors que utilitza.

Per últim, en el tercer bloc es descriu el tractament de l'XML dins el camp de les bases de dades. S'analitzen les dues tendències existents: els sistemes gestors de bases de dades tradicionals i els sistemes gestors desenvolupats seguint totalment la tecnologia XML. Els primers, davant la forta evolució de l'XML, s'han vist obligats a desenvolupar interfícies i utilitats que s'adaptin als seus sistemes i que els habilitin per a l'ús de l'XML. Els segons, són sistemes natius XML en els que les dades es guarden directament en les estructures internes del propi sistema, mantenint el format jeràrquic de l'XML i sense necessitat de cap procés de transformació previ. Per presentar aquests dos tipus de tractament de l'XML, s'estudien i es comparen dos casos reals: el SGBD relacional d'Oracle i el SGBD XML natiu desenvolupat per Tamino.

Índex:

1 Introducció ____ *5*
1.1 Justificació del PFC ____ **5**
1.2 Objectius ____ **5**
1.3 Enfocament i mètode seguit ____ **5**
1.4 Planificació del projecte ____ **6**
1.5 Breu descripció dels propers capítols de la memòria ____ **7**
2 Introducció a l'XML ____ *9*
2.1 Que és l'XML ? ____ **9**
2.2 Origen i evolució de l'XML ____ **10**
2.3 Els estàndards XML i el W3C ____ **11**
3 Conceptes fonamentals d'XML ____ *14*
3.1 Documents XML ____ **14**
3.2 DTD (Document Type Definitions) ____ **17**
3.3 Esquemes XML ____ **21**
4 Tecnologies de transformació i presentació ____ *23*
4.1 CSS ____ **23**
4.2 XSL: ____ **24**
4.2.1 XSLT ____ 25
4.2.2 XSL-FO ____ 27
5 Tecnologies de navegació i accés ____ *28*
5.1 XLink ____ **28**
5.2 XPointer ____ **30**
5.3 XPath ____ **31**
5.4 XQuery ____ **34**
6 Tecnologies de suport per l'anàlisi i manipulació ____ *35*
6.1 DOM ____ **35**
6.2 SAX ____ **35**
7 XQuery ____ *37*
7.1 Que és XQuery? ____ **37**
7.2 Data Model ____ **38**
7.3 Treballant amb XQuery ____ **38**
7.3.1 Conceptes bàsics ____ 39
7.3.2 Localització de nodes (expressions i predicats) ____ 40
7.3.3 Constructors: creació de nodes ____ 40
7.3.4 Combinació i estructuració de nodes ____ 42

7.3.5 Operadors XQuery 46
7.3.6 Funcions 47

8 XML i les bases de dades 48

8.1 Bases de dades relacionals vs bases de dades XML natiu 48

8.2 Oracle 49
8.2.1 Oracle XDK 50
8.2.2 Oracle XML DB 51

8.3 Tamino 53

9 Conclusions 56

1 Introducció

1.1 Justificació del PFC

El treball consisteix en fer una recopilació i estudi sobre XML. Aquest estudi inclou tant els conceptes propis del XML, com els conceptes i aplicacions dels diferents estàndards i tecnologies existents relacionades amb XML (XSLT, Xlink, Xpath, Xquery, DOM, SAX, etc.).

S'ha de realitzar una recopilació de la nombrosa informació existent sobre XML per tal d'extreure'n la informació necessària per conèixer quina ha estat l'evolució del XML i dels diferents estàndards, així com dels llenguatges de consulta, quin és el seu estat actual i quina és la tendència a seguir.

El *World Wide Web Consortium* (W3C) juga un paper molt important en la definició de XML i dels diferents estàndards per tant, caldrà veure quines funcions realitza i com influeix en l'evolució de l'XML.

Cal aprofundir i parar especial atenció sobre els llenguatges de consulta XML i les diferents bases de dades que existeixen al mercat i que permeten utilitzar XML com font de dades, tant les bases de dades relacionals com les natives.

Entre els llenguatges de consulta XML existents, ens centrarem en l'estudi del llenguatge XQuery. Descriurem el model de dades en que es basa i veurem la sintaxi XQuery, així com les principals funcions i operadors que utilitza.

Finalment, es procedirà a estudiar com tracta l'XML un gestor de base de dades relacional (Oracle) i un gestor basat en XML (Tamino). D'aquesta manera podrem veure les diferencies entre tots dos i avaluar els avantatges i inconvenients que aporten cadascun d'ells.

1.2 Objectius

L'objectiu d'aquest projecte és realitzar un estudi sobre XML, els seus estàndards i la relació de l'XML amb el camp de les bases de dades. Més concretament, podem destacar els següents objectius:

- Conèixer l'origen, evolució i conceptes de l'XML.
- Conèixer l'origen, evolució i conceptes dels principals estàndards i tecnologies XML (XSLT, XLink, XPath, DOM, SAX, etc.).
- Recopilar informació sobre els llenguatge de consulta XML (XQuery).
- Realització d'un estudi per comparar el tractament de l'XML d'Oracle i de Tamino.

Després de la realització del projecte s'han d'haver assolit tots els objectius plantejats i per tant, s'han d'haver après i assimilat:

- Els conceptes fonamentals i aplicacions de l'XML i dels diferents estàndards.
- La sintaxi i les principals característiques del llenguatge de consulta XQuery.
- La relació de l'XML amb les bases de dades.
- Les diferències, avantatges i inconvenients del tractament de l'XML amb un gestor de base de dades relacional (com Oracle) i amb un gestor de base de dades XML (com Tamino).

1.3 Enfocament i mètode seguit

D'acord amb la finalitat del projecte i per tal d'assolir els objectius plantejats, s'ha procedit a fer un estudi i una recopilació d'aquella informació referent a XML i a tota la família d'estàndards i tecnologies que hi estan relacionats.

Justament al fer aquesta recopilació d'informació, un s'adona de l'abast real del projecte, ja que al ser XML una tecnologia relativament nova, existeix una gran quantitat d'informació relativa a XML i per

tant, cal procedir a fer una estructuració i classificació de tota aquesta informació, per tal de tenir clars quins són els aspectes més importants de l'XML i dels diferents estàndards.

Una de les fonts d'informació bàsiques per l'elaboració d'aquest projecte, ha estat la informació facilitada pel W3C (*World Wide Web Consortium*), organisme que promou i desenvolupa el futur de Internet, contribuint a l'estandardització de les tecnologies web mitjançant la publicació de les especificacions (o "recomanacions") que descriuen aquestes tecnologies. Tota la informació que s'ha extret del W3C es troba totalment actualitzada i descriu l'estat actual de l'XML i dels diferents estàndards.

També s'han utilitzat altres fonts d'informació però cal dir, que al ser XML una tecnologia en constant evolució, en alguns casos aquesta informació era poc exhaustiva i no estava actualitzada. D'aquí la importància de la informació facilitada pel W3C.

Així doncs, la metodologia que s'ha seguit per l'elaboració d'aquesta part del projecte ha estat:

- Cerca i recopilació d'informació de diverses fonts.
- Estudi, estructuració i classificació d'aquesta informació.
- Redacció de les principals característiques i propietats de l'XML i dels diferents estàndards (definicions, exemples, etc.).

Posteriorment, s'ha procedit a fer un estudi sobre el tractament de l'XML per part d'Oracle i Tamino. En aquest cas, la metodologia a seguir ha estat la mateixa però, les fonts principals d'informació per realitzar aquest estudi, han estat les dades facilitades pels propis actors implicats, Oracle i Tamino.

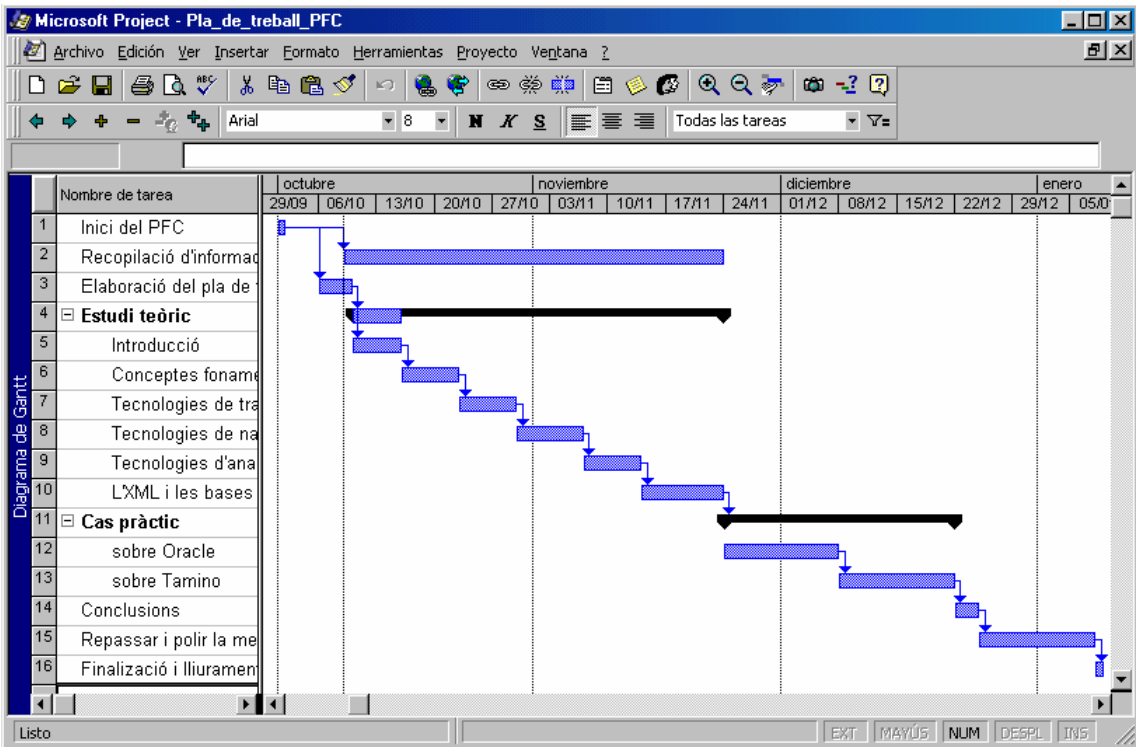
1.4 Planificació del projecte

La següent taula mostra els diferents apartats en que s'ha estructurat la memòria, així com les tasques a realitzar en cadascun d'ells:

Apartat	Tasques a realitzar
Introducció	• Recopilació d'informació. • Definir l'XML i explicar quin és el seu origen, els seus objectius, els estàndards relacionats i les funcions del W3C en la seva definició.
Conceptes fonamentals de l'XML	• Explicar els conceptes bàsics de l'XML: quina és l'estructura dels documents XML, que és un document vàlid, que és un DTD, que és un esquema XML, etc...
Tecnologies de tranformació i presentació	• Descriure els estàndards XML relacionats amb la presentació, format i transformació dels documents XML: la definició d'estils (CSS) i de plantilles de transformació (XSL, XSLT, XSL-FO).
Tecnologies de navegació i accés	• Descriure els estàndards relacionats amb els enllaços entre documents XML (Xlink, Xpointer). • Descriure els estàndards i llenguatges d'accés i consulta sobre els documents XML (Xpath, Xquery).
Tecnologies de suport per l'anàlisi i manipulació	• Explicar els conceptes fonamentals de les tecnologies que ens permeten manipular els documents XML i desenvolupar aplicacions (SAX i DOM).
XQuery	• Recopilació d'informació. • Descriure el model de dades en que es basa XQuery. • Fer una introducció a XQuery (sintaxi XQuery, funcions i operadors).

L'XML i les bases de dades	• Descriure les relacions de l'XML amb les bases de dades. • Tractament de l'XML per una SGBD relacional (Oracle). • Tractament de l'XML per una SGBD XML natiu (Tamino). • Comparació, avantatges i inconvenients entre les dues tecnologies.
Conclusions	• Conclusions sobre la potència de l'XML i els beneficis que aporta l'XML i la resta de tecnologies relacionades. • Conclusions sobre l'ús de l'XML en el camp de les bases de dades.

Tot seguit mostrem la planificació temporal d'aquestes tasques:



1.5 Breu descripció dels propers capítols de la memòria

- *Introducció a l'XML*

  En aquest capítol es fa una primera introducció a l'XML i als diversos estàndards i tecnologies que hi estan relacionats. Primer es defineix l'XML indicant quin és el seu origen, quins són els seus objectius de disseny, quin organisme és el responsable de la seva evolució i quines són les fites que es volen assolir mitjançant l'ús d'XML. Posteriorment, es presenten breument els principals estàndards i tecnologies que constitueixen la família XML: DTD, Esquemes XML, XSL, XSLT, XSL-FO, XPath, XPointer, XLink, DOM i SAX. En els posteriors capítols, veurem cadascun d'ells en més detall.

- *Conceptes fonamentals d'XML*

  A l'inici d'aquest capítol es defineixen els documents XML, descrivint la seva estructura lògica (formada bàsicament per elements i atributs) i la seva estructura física (formada pels diferents tipus d'entitats). Posteriorment, es descriuen els documents que serveixen per definir la gramàtica dels documents XML: els DTDs i els Esquemes XML. Mitjançant els DTDs s'especifiquen els tipus d'elements, atributs, entitats i notacions que poden aparèixer en un document XML (si el document XML indica un DTD i el satisfà es considera un document vàlid). Els Esquemes XML utilitzen la

pròpia sintaxi XML i milloren les limitacions dels DTDs permetent, entre d'altres punts, afegir certa lògica a les declaracions, disposar de diferents tipus de dades, utilitzar indicadors de concurrència, etc. En el capítol trobem diversos exemples de DTDs i d'Esquemes XML que ens mostren la seva aplicació.

- *Tecnologies de transformació i presentació*

  XML descriu l'estructura del document però no defineix les propietats de presentació i visualització de les dades que conté. Justament, en aquest capítol es presenten els mètodes que serveixen per definir i modificar la forma en que es visualitzen les dades dels documents XML: les fulles d'estil en cascada (CSS) i el llenguatge d'estils extensible (XSL). Mitjançant CSS es poden definir el conjunt de regles d'estil que s'hauran d'aplicar als diferents elements del document XML. XSL consta de dues parts: XSLT, un llenguatge de transformació mitjançant el qual un document XML és pot transformar en un altre document i XSL-FO, un llenguatge de formatge que serveix per descriure de forma precisa l'aparença de les dades d'un document XML.

- *Tecnologies de navegació i accés*

  En aquest capítol es presenten els estàndards i llenguatges XML definits que permeten navegar entre diferents documents XML i navegar dins l'estructura jeràrquica que formen els documents XML, fent referència a parts concretes (elements, atributs, etc.) d'aquesta estructura. Concretament, veurem els següents llenguatges: XLink, que permet la creació d'enllaços entre dos o més documents XML, XPointer, que permet identificar i fer referències a punts o fragments que es troben dins d'un document XML i per últim, XPath, que permet seleccionar i fer referència a text, elements, atributs i qualsevol altre tipus d'informació que es trobi dins d'un document XML. XQuery és el llenguatge més potent i robust definit per navegar, manipular i seleccionar les dades dels documents XML i per aquest motiu, es descriu en més detall en un altre capítol.

- *Tecnologies de suport per l'anàlisi i manipulació*

  En aquest capítol es presenten dues de les principals tecnologies de suport per l'anàlisi i manipulació dels documents XML: DOM i SAX. La primera, proporciona una interfície de programació per al desenvolupament d'aplicacions. Aquesta interfície és orientada a objectes i crea una representació de l'estructura del document en memòria. SAX de forma similar a DOM, també proporciona una interfície i un conjunt de classes per al processament dels documents XML però, a diferència de DOM, proporciona una plataforma més simple que està basada en events.

- *XQuery*

  XQuery és el llenguatge de consulta més potent i robust que s'ha desenvolupat per poder interrogar, recuperar i interpretar la informació que contenen els documents XML. Aquest capítol, pretén ser una introducció a XQuery, descrivint el model de dades en que es basa i veient la sintaxi XQuery, així com les principals funcions i operadors que utilitza.

- *XML i les bases de dades*

  En aquest capítol es descriu el tractament de l'XML dins el camp de les bases de dades. Existeixen dues tendències clares. Per una banda, hi ha els sistemes gestors de bases de dades tradicionals que, davant la forta evolució de l'XML, han adaptat els seus sistemes oferint interfícies i utilitats que permeten que les dades XML puguin ser emmagatzemades en les seves taules relacionals. Per altra banda, hi ha els nous sistemes de bases de dades natius XML, en els que les dades és guarden directament en les seves estructures internes mantenint el format jeràrquic de l'XML, sense necessitat de cap procés de transformació previ. En aquest sentit, es descriuen i es comparen les solucions proposades per Oracle (XDK i XML DB) i Tamino (sistema natiu XML).

2 Introducció a l'XML

2.1 Que és l'XML ?

L'XML (*eXtensible Markup Language*) és un llenguatge extensible de marques que defineix un format estàndard per a l'estructuració de dades i informació. XML no és un concepte nou, va ser desenvolupat pel Grup de Treball XML l'any 1996, sota la supervisió de l'organisme W3C[1] (*World Wide Web Consortium*) i va sorgir a partir d'un estàndard previ anomenat SGML (*Standard Generalized Markup Language*), molt més potent que XML però també molt més complex i difícil d'implementar.

> L'organisme W3C va ser creat l'Octubre de 1994 amb l'objectiu de treure el màxim rendiment de la Web mitjançant el desenvolupament de protocols (especificacions, guies d'estil, software i eines) que promouen la seva evolució i asseguren la seva interoperabilitat. Està format per més de 400 organitzacions de tot el món i està reconegut internacionalment per la seva contribució en el creixement de Internet.
>
> W3C promou i desenvolupa el futur de la Web identificant els requisits tècnics que s'han de satisfer per fer de la Web un espai veritablement universal de informació. W3C contribueix en l'estandardització de les tecnologies web desenvolupant les especificacions (anomenades "recomanacions") que descriuen aquestes tecnologies i les fan accessibles per a tothom.

Més que un llenguatge, XML és un metallenguatge que permet dissenyar llenguatges d'etiquetes per a diferents usos específics. Així doncs, XML no proporciona un conjunt fix d'etiquetes com l'HTML, sinó que permet als desenvolupadors crear les etiquetes que realment necessiten per estructurar la informació que els interessa (una agenda personal, un catàleg de productes, un expedient acadèmic, etc.). Qualsevol tipus de dades que puguin ser estructurades de forma jeràrquica és pot representar mitjançant un document XML.

En un document XML, les dades és representen mitjançant cadenes de text delimitades per etiquetes que descriuen la semàntica de les dades. L'especificació XML defineix la sintaxis que ha de complir un document XML per estar ben format (com s'han de delimitar els elements per etiquetes, quins noms es poden utilitzar, etc.) però no especifica com s'han de visualitzar aquestes dades. Aquesta, és precisament una de les característiques responsables de l'èxit de XML, la independència entre el contingut de les dades i la seva representació. Això, permet que les dades estructurades en un document puguin ser reutilitzades i mostrades en diferents formats, en diferents dispositius de sortida, adaptant-se amb flexibilitat a les necessitats del moment.

Una altra característica interessant de XML és que permet l'intercanvi de dades i informació de forma estàndard i totalment independent de les plataformes i aplicacions utilitzades. Això, converteix XML en una solució molt potent per realitzar l'intercanvi de documents per Internet..

El fet de que la informació en els documents XML estigui perfectament estructurada en funció del seu significat, fa que les dades puguin ser localitzades de forma precisa i eficient. Això, ha donat lloc a l'aparició de bases de dades especialment desenvolupades per a XML, com per exemple Tamino. Aquests sistemes de bases dades poden emmagatzemar documents XML mantenint la seva estructura original, permetent que els resultats de les cerques siguin molt ràpids i eficients.

Així doncs, XML és un estàndard que proporcionar una arquitectura oberta, capaç de separar la presentació de les dades del seu contingut i alhora ser capaç de donar als documents una estructura útil i lògica.

Tot seguit, veurem quina ha estat l'evolució de XML al llarg de la seva història i quins són la resta d'estàndards i tecnologies que converteixen XML en una eina realment potent i versàtil.

---

[1] *World Wide Web Consortium, http://www.w3c.org*

2.2 Origen i evolució de l'XML

En els darrers anys, l'HTML (*Hypertext Markup Language*) s'ha convertit en un llenguatge de marques de gran popularitat i ús per a la creació de pàgines web. La seva gran expansió en tan poc temps, s'ha degut a la seva senzillesa, al seu conjunt relativament reduït d'etiquetes i a la seva tolerància davant fallades, fent que tant usuaris professionals com aficionats siguin capaços d'utilitzar-lo.

Avui en dia però, les necessitats actuals, com per exemple, l'intercanvi de dades entre dos sistemes de bases de dades incompatibles, denoten les mancances i limitacions que presenta l'HTML. Algunes d'aquestes mancances s'han intentat corregir mitjançant l'ús d'scripts, javascripts, HTML dinàmic, fulles d'estil en cascada CSS, etc. però aquestes mesures, són insuficients per poder proporcionar una arquitectura client/servidor oberta, capaç de separar la presentació de les dades del seu contingut i alhora ser capaç de donar als documents una estructura útil i lògica.

Sorgeix per tant la necessitat de crear un nou estàndard que sigui capaç de proporcionar aquestes característiques, ordenant i estructurant el contingut dels documents: és el naixement de l'XML.

L'XML està basat en un subconjunt de l'estàndard SGML (*Standard Generalized Markup Language*). L'origen del SGML data dels anys 70, quan es va desenvolupar el SGML com un potent metallenguatge de marques per definir l'estructura i semàntica de documents de text. L'any 1986 va ser reconegut com l'estàndard ISO 8879.

SGML és extremadament potent, així per exemple, HTML és una aplicació concreta de SGML (la definició del conjunt d'etiquetes per descriure les pàgines web). SGML també és utilitzat per organismes militars i governamentals del EEUU, pel sector aerospacial, i per altres dominis que necessiten emmagatzemar de forma ordenada i eficient gran quantitat d'informació.

El principal problema de SGML és la seva complexitat, fent que cobreixi molts casos particulars en diferents tipus d'escenaris. És tant complex que la majoria de software no l'implementa completament i només un grup reduït d'especialistes coneixen el llenguatge en la seva totalitat. Aquesta complexitat va provocar que el 1996, el Grup de Treball XML (inicialment anomenat *SGML Editorial Review Board*) treballessin en una versió reduïda de SGML. Aquesta nova versió havia de mantenir la potencia del SGML però eliminant algunes característiques redundants, massa complicades d'implementar, confuses per als usuaris finals o que encara que simples no havien estat utilitzades durant els 20 anys anteriors d'experiència amb el SGML. El resultat d'això va ser l'especificació de l'estàndard XML 1.0, especificació ratificada per la W3C el 10 de Febrer de 1998.

Alguns dels objectius de disseny del XML seguits pel grup de treball XML i el W3C van ser:

- XML havia de ser directament utilitzable sobre Internet.
- XML havia de suportar una gran varietat d'aplicacions.
- XML havia de ser compatible amb SGML.
- Havia de ser senzill escriure programes que processessin l'XML.
- El número de capacitats opcionals de XML havia de ser mínim, idealment zero.
- Els documents XML havien de ser llegibles per a les persones i clars.
- El disseny de XML havia de finalitzar de forma ràpida.
- XML havia de ser simple però alhora perfectament formalitzat.
- Els documents XML havien de ser senzills de crear.
- La brevetat de les marques XML es de mínima importància.

Juntament amb el propi estàndard XML, el W3C treballa en l'especificació de la resta d'estàndards i tecnologies relacionades amb XML (XSL, XPath, XPointer, XLink, etc.) , que constitueixen l'anomenada "família XML". Tot seguit, descriurem breument aquests estàndards i veurem quin és el seu estat actual davant el W3C.

2.3 Els estàndards XML i el W3C

El futur de l'estàndard XML està fortament lligat amb el desenvolupament del propi XML però també de l'evolució de la resta d'estàndards que hi estan relacionats, és per això, que dins del W3C hi trobem diversos grups de treball responsables de l'activitat i seguiment de cadascun d'aquests estàndards.

Són aquests estàndards els que permeten que el potencial actual de XML pugui ser aplicat en aplicacions efectives. Tot seguit anem a descriure breument aquests estàndards i la seva situació davant del W3C. En concret veurem: DTD, Esquemes XML, XSL, XSLT, XSL-FO, XPath, XPointer, XLink, DOM i SAX. En els següents capítols ja entrarem més en detall en cadascun d'ells.

- **DTD:**

  Els DTD (*Document Type Definitions*) serveixen per definir la gramàtica dels documents XML, és a dir, els tipus d'elements, atributs, entitats i notacions que es podran utilitzar en el document XML, així com certes restriccions estructurals i de contingut, valors per defecte, etc. Si un document XML té indicat un DTD i el satisfà, llavors el document és considera vàlid. En cas contrari, és un document invàlid.

- **Esquemes XML:**

  Els esquemes XML (*XML Schemas*) s'utilitzen per definir l'estructura, el contingut i la semàntica dels documents XML. De forma similar als DTDs, els esquemes XML especifiquen quins elements pot contenir un document XML, com estan organitzats, quins atributs tenen i de quin tipus són. Les principals avantatges que presenten davant els DTDs són que utilitzen sintaxis XML, permeten especificar els tipus de dades i són extensibles.

  Estat davant el W3C:
  - Recomanació XML Schema 1.0, publicada el 2 de maig de 2001
    http://www.w3.org/TR/xmlschema-0/
    http://www.w3.org/TR/xmlschema-1/
    http://www.w3.org/TR/xmlschema-2/
  - Actualment treballant en el desenvolupament de XML Schema 1.1
    http://www.w3.org/XML/Schema/

- **XSL:**

  XSL (*Extensible Stylesheet Language*) és el llenguatge que s'utilitza per aplicar format als documents XML. Les fulles d'estil XSL afegeixen a un document XML una descripció de com s'ha de visualitzar i inclouen una certa capacitat de procés i transformació del contingut del document. Al llarg de la seva evolució, s'ha dividit donant lloc a dos nous estàndards: XSLT, llenguatge per transformar documents XML i XSL-FO, vocabulari XML per especificar el format dels documents XML.

  Estat davant el W3C:
  - Recomanació XSL 1.0, publicada el 15 d'octubre de 2001
    http://www.w3.org/TR/xsl

- **XSLT:**

  XSLT (*Extensible Stylesheet Language Tranformation*) és l'especificació que defineix el llenguatge per transformar un document XML en un altre document XML. Ha estat dissenyat per poder ser utilitzat independentment de XSL, tot i que el seu ús està fortament lligat a l'especificació XSL. Amb XSLT s'especifiquen les regles, filtres o patrons mitjançant els quals, un document XML és transformat en un altre document XML, substituint els patrons del document XML pel contingut indicat per aquell patró en el document XSLT.

Estat davant el W3C:
- Recomanació XSLT 1.0, publicada el 16 de novembre de 1999
  http://www.w3.org/TR/xslt
- Esborrany de treball XSLT Requeriments 2.0, publicat el 14 de febrer de 2001 1999
  http://www.w3.org/TR/xslt20req

- **XSLT-FO:**

XSLT-FO (*XML Stylesheet Language Formatting Objects*) és el llenguatge de formateig que descriu de forma precisa com s'ha de visualitzar el contingut dels documents XML. Mitjançant etiquetes XML s'especifiquen els elements que serveixen per definir pàgines, blocs de text, marges, gràfics, etc. Els documents de sortida poden ser documents XML però també documents d'altres tipus, com WML o PDF.

Estat davant el W3C:
- Recomanació XSL 1.0, publicada el 15 d'octubre de 2001
  http://www.w3.org/TR/xsl

- **XPath:**

XPath és un llenguatge que permet direccionar parts d'un document XML. Opera sobre l'estructura lògica abstracta del document XML més que en la seva sintaxis. XPath modela un document XML com un arbre de nodes i navega a través d'aquesta estructura jeràrquica. És utilitzat tant per XSL com per Xpointer, ja que els proporciona una sintaxis i semàntica comuna per identificar i fer referències als elements dels documents XML.

Estat davant el W3C:
- Recomanació XPath 1.0, publicada el 16 de novembre de 1999
  http://www.w3.org/TR/xpath
- Esborrany de treball XPath 2.0, publicat el 12 de novembre de 2002
  http://www.w3.org/TR/xpath20

- **XPointer:**

XPointer és l'estàndard que permet identificar i fer referències a parts que es troben dins d'un document XML. XPointer especifica com navegar dins d'un document XML, directament mitjançant apuntadors a paraules específiques o frases, o bé indirectament mitjançant l'especificació d'un camí.

XPointer ha estat dividit pel W3C en un marc de treball XPointer (*framework*) i tres esquemes: *element()*, *xmlns()* i *xpointer()*. El marc de treball i els dos primer esquemes ja han estat formalitzats pel W3C amb les seves corresponents recomanacions. L'esquema *xpointer()* es troba encara en fase d'estudi i existeix una proposta de recomanació pendent de ser aprovada.

Estat davant el W3C:
- Esborrany de treball, publicat el 16 d'agost de 2002
  http://www.w3.org/TR/2002/WD-xptr-20020816/
- Recomanació XPointer 1.0, publicada el 25 de març de 2003.
  http://www.w3c.org/TR/xptr-framework/
  http://www.w3c.org/TR/xptr-element/
  http://www.w3c.org/TR/xptr-xmlns/
- Proposta de recomanació per l'esquema xpointer() de XPointer, 19 de setembre de 2002.
  http://www.w3c.org/TR/xptr-xpointer/

- **XLink:**

XLink (*XML Linking Language*) és el llenguatge que permet introduir enllaços en els documents XML de manera que, podem relacionar uns documents amb d'altres. Aquests enllaços poden ser simples (similars als típics enllaços HTML) o bé ser estructures més complexes i potents.

Estat davant el W3C:

- Recomanació XLink 1.0, publicada el 27 de Juny de 2001
  http://www.w3.org/TR/xlink/

- **DOM:**

DOM (*Document Object Model*) és una API, és a dir, una interfície de programació per al desenvolupament d'aplicacions, orientada a objectes i dissenyada per a treballar amb documents XML. Per DOM, un document XML és un conjunt d'objectes que es relacionen entre sí mitjançant una jerarquia d'objectes en forma d'arbre, de manera que navegant pels nodes de l'arbre, es poden afegir, modificar o eliminar elements del document o crear nous documents.

Estat davant el W3C:

- Recomanació DOM Level 1.0, publicat l'1 d'octubre de 1998
  http://www.w3.org/TR/rec-dom-level-1/
- Candidat a recomanació DOM Level 2.0, publicat el 10 de maig de 2000
  http://www.w3.org/TR/dom-level-2/
- Esborrany de treball DOM Level 3.0, publicat l'1 de setembre de 2000
  http://www.w3.org/TR/dom-level-2-Core/

- **SAX:**

SAX (*Simple API for XML*), és una interfície simple i molt fàcil d'utilitzar per treballar amb documents XML. SAX és suportat per la majoria d'analitzadors sintàctics i s'utilitza per accedir als documents XML i processar-los. Al contrari que DOM, SAX és una API orientada a events, no es generen estructures internes i el tractament de les dades es realitza en el moment en que és produeixen els events.

Estat:

- Especificació SAX 2.0 - *XML-DEV Mailing List Member Specification, Megginson Technologies* , 5 de Maig de 2000
  http://www.megginson.com/SAX/index.html
- Especificació SAX 1.0 - *XML-DEV Mailing List Member Specification, Megginson Technologies* , 11 de Maig de 1998
  http://www.megginson.com/SAX/SAX1/index.html

3 Conceptes fonamentals d'XML

3.1 Documents XML

Un document XML està format únicament per text pla, no conté codi binari. Això fa que pugui ser llegit per qualsevol programa capaç de llegir un fitxer de text. L'exemple 2.1 mostra un document XML molt simple però que ja deixa entreveure com la informació està estructurada i delimitada mitjançant marques.

```
<autor>
       <nom>Isabel</nom>
       <cognom>Allende</cognom>
</autor>
```

Exemple 3.1 Document XML molt senzill

Tot document XML consta d'una *estructura física* i una *estructura lògica*:

- Físicament, el document està compost per una o més unitats d'emmagatzematge anomenades "*entitats*" que poden contenir tant dades analitzades com dades sense analitzar. Les entitats analitzades constitueixen el contingut del document (les dades). Les entitats no analitzades són recursos que poden contenir tant text com altres formats.
- Lògicament, el document està compost per declaracions, elements, atributs, comentaris, referències a caràcters i instruccions de processament. Tots ells, han d'estar explícitament delimitats mitjançant marques i han de seguir una sèrie de regles de marcatge per tal de que el document estigui "*ben format*".

Les marques d'un document XML tenen un aspecte molt similar a les marques HTML. La diferència es troba en que HTML proporciona un conjunt predefinit d'etiquetes que descriuen el format d'una pàgina web, en canvi, XML ens permet definir les nostres pròpies etiquetes d'acord amb la informació que volem estructurar. Així per exemple, *<autor>*, *<títol>* i *<editorial>* serien etiquetes que ens definiríem per estructurar les dades d'un catàleg de llibres mitjançant un document XML.

L'exemple 3.2 mostra com les etiquetes HTML no expressen el mateix que les etiquetes XML:

```
Document HTML:

<html>
<title>Exemple HTML</title>
<body>
       El llibre <b>La casa de los espiritus</b> de l'autora
       <em>Isabel Allende</em> té un preu de <b>28,90</b> euros.
</body>
</html>

Document XML:

<?xml version="1.0"?>
<llibre>
  <titol>La casa de los espíritus</titol>
  <autor>
         <nom>Isabel</nom>
         <cognom>Allende</cognom>
  </autor>
  <categoria>novel·la</categoria>
  <preu quantitat="28,90" moneda="euro"/>
</llibre>
```

Exemple 3.2 Diferent significat de les etiquetes HTML i XML. Document XML que estructura la informació d'un llibre.

Amb HTML la informació no està estructurada semànticament i les etiquetes serveixen sobretot per definir el format del document. Les marques XML en canvi, afegeixen informació sobre el contingut del document, descriuen la semàntica del document i la seva estructura. Existeix també un grup reduït de marques XML predefinides, que afegeixen informació sobre el propi document (meta-informació), com per exemple, la versió XML utilitzada al escriure el document XML.

Elements i atributs

Els components clau que formen l'estructura lògica d'un document XML són els elements i els atributs.

- L'element és la unitat bàsica amb capacitat per representar l'estructura lògica i la semàntica d'un document XML, en definitiva, el seu contingut. Són les diferents parts d'informació en que podem dividir un document XML.

- Els atributs són components dels elements que actuen com a modificadors d'aquests, afegint informació addicional a l'element al que estan associats.

Els elements poden tenir associats un o més atributs i, poden contenir text o bé altres elements com a fills. El document XML de l'exemple 3.2, que estructura la informació d'un llibre, està format pels elements `llibre`, `títol`, `autor`, `nom`, `cognom`, `categoria` i `preu`. L'element `títol` conté text mentre que l'element `autor` conté els elements `nom` i `cognom`. L'element `preu` conté els atributs `quantitat` i `moneda`.

Tots els documents XML han de tenir obligatòriament un **element arrel**. És el primer element del document i conté tots els altres elements. En l'exemple anterior, l'element `alumne` és l'element arrel del document, també anomenat **element document**. Un document XML ben format té un i només un element arrel.

Donat que els element no es poden sobreposar entre ells i que tots els elements tenen un element pare (excepte l'element arrel), els documents XML es poden representar com una jerarquia estrictament aniuada d'elements. La figura 3.1, ens mostra l'arbre que representa el document XML de l'exemple 3.2.

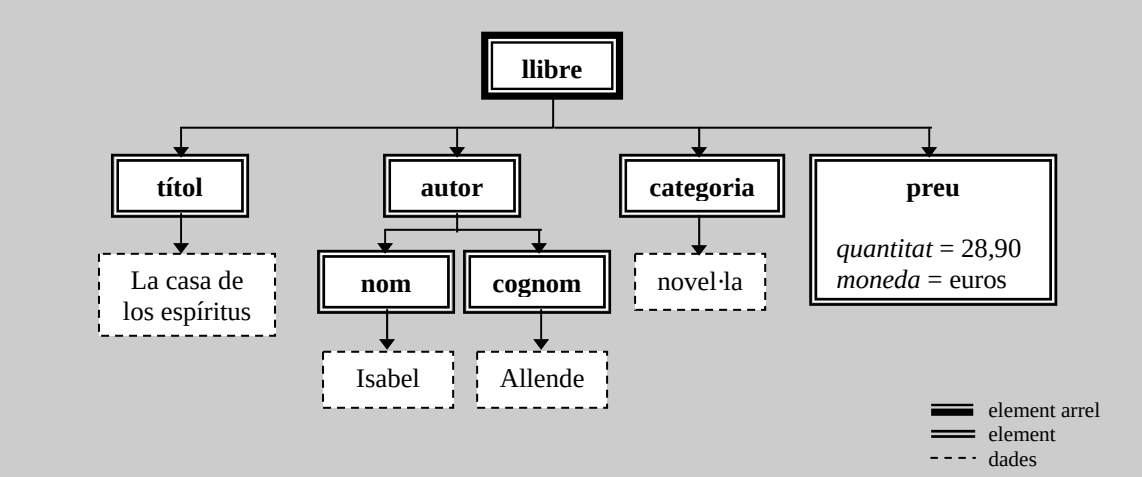


Figura 3.1. Estructura jeràrquica en forma d'arbre del document XML de l'exemple 3.2.

Els documents XML han de seguir estrictament una sèrie de regles bàsiques de marcatge per tal d'assegurar un mínim de coherència i precisió alhora d'analitzar o llegir els documents XML. Aquest fet també ajuda a simplificar els programes que han de processar els documents XML.

Algunes de les principal regles de marcatge són:

- Tots els elements tenen una marca d'inici i una marca de fi:
  *<identificador>*element*</identificador>*.
- Els element buits (que no tenen contingut) poden tenir una única marca d'inici i fi:
  <identificador/>

- Totes les marques comencen per '<' i acaben amb '>'.
- Les marques no es poden suprimir.
- Es fa sempre distinció entre majúscules i minúscules. Per exemple *<persona>*, *<Persona>* i *<PERSONA>* són marques de diferents elements.
- Un document XML conté un i només un element arrel, de forma que cap part del mateix pot aparèixer com a contingut d'un altre element.
- Les marques dels element aniuats no poden solapar-se, s'han de tancar seguint l'ordre d'obertura. Per exemple, això és incorrecte: *<a>*text**.
- Els valors dels atributs només es poden especificar dins de la marca d'inici dels elements.
- Un element no pot tenir dos atributs amb el mateix nom.
- Els valors dels atributs s'han d'indicar entre cometes simples o dobles.
- Els comentaris estan formats per la cadena d'inici "<!--" i finalitzen amb la cadena "-->".

Direm que un document és un document ben format si compleix amb les regles descrites anteriorment.

Entitats

XML anomena **entitats** a les unitats físiques que componen un document (fitxers externs, recursos accessibles via URL, fragments de text, etc.). Les entitats ajuden a organitzar el document, independentment de la seva estructura lògica. El propi fitxer que conté el document XML és una entitat, l'anomenada **entitat document**.

La recomanació XML, diferencia clarament dos tipus d'entitats:

- Les entitats generals
- Les entitats paràmetre

Les **entitats generals**, bàsicament s'utilitzen per posar nom a cadenes de text que posteriorment s'utilitzaran dins del document XML. També s'utilitzen per incloure dins del document XML, dades que, en principi, no són XML.

Les **entitats paràmetre** també s'utilitzen per a posar nom a una cadena de text, però s'utilitzen exclusivament dins la definició del tipus de document (anomenada DTD, que descriurem en el següent punt) i contenen text que té sentit per aquell entorn, principalment declaracions de marcatge.

Podríem dir que existeix un tercer tipus d'entitats, les **entitats predefinides**, que s'utilitzen per escapar els caràcters especials que intervenen en el marcatge. Per exemple, s'utilitza l'entitat "&amp" per escapar el caràcter "&", les entitats "<" i ">" per escapar els caràcters "<" i ">" respectivament, etc.

D'altra banda, podem classificar les entitats com:

- Entitats **analitzables**, són aquelles que contenen informació que el processador XML entén i és capaç de processar.
- Entitats **no analitzables**, són aquelles que contenen informació que per al processador XML no és XML, com per exemple, imatges o sons.
- Entitats **internes**, aquelles en les que el seu contingut es troba totalment dins del document XML.
- Entitats **externes**, aquelles en les que el seu contingut està definit total o parcialment fóra del document XML.

3.2 DTD (Document Type Definitions)

Com hem vist en el punt anterior, una de les principals característiques de l'XML és la seva flexibilitat per definir i estructurar qualsevol tipus d'informació. D'altra banda però, les aplicacions que treballen amb XML només necessiten llegir determinats tipus de documents XML i per tant, resultaria interessant poder restringir d'alguna manera el contingut dels elements i atributs que poden aparèixer en un document XML. Això s'aconsegueix mitjançant les definicions de tipus de documents, els DTD. Els DTD faciliten les tasques dels processadors XML, donant lloc a resultats més flexibles i a més, permeten una major senzillesa a l'hora de programar les aplicacions que s'encarregaran de processar els documents XML

Així doncs, l'objectiu dels DTD és definir la gramàtica dels documents XML, és a dir, els tipus d'elements, atributs, entitats i notacions que es podran utilitzar en el document XML, així com certes restriccions estructurals i de contingut.

Direm que un document XML és un **document vàlid** si inclou una declaració de tipus de document que identifica un DTD i el satisfà. En cas contrari, direm que és un **document invàlid**. La validació és opcional, és a dir, donat un document XML no és obligatori que aquest indiqui quin és el seu DTD. Els parsejadors XML mitjançant els DTD detecten les violacions que es poden produir en el document XML, ja que entre d'altres coses, el DTD indica:

- quin és l'element arrel del document.
- quantes ocurrències de cada classe d'element poden aparèixer en el document.
- quin tipus de dades conté cada element (si conté altres elements o dades).
- quins atributs pot tenir un determinat element.
- quin tipus de dades pot contenir cada atribut.
- quins són els valors per defecte dels atributs, etc.

Cal dir però que tot i els avantatges dels DTD, aquests són poc flexibles i no permeten especificar propietats com la longitud, l'estructura, els valors permesos o d'altres aspectes del contingut dels elements.

Tot seguit, veurem en més detall quins tipus de declaracions poden aparèixer en un DTD però, en el següent exemple, ja podem veure quin és l'aspecte del DTD que defineix el document XML de l'exemple 3.2, corresponent a la descripció d'un llibre.

```
<!ELEMENT llibre    (titol, autor, categoria*, preu)>
<!ELEMENT titol     (#PDATA)>
<!ELEMENT autor     (nom, cognom)>
<!ELEMENT nom       (#PCDATA)>
<!ELEMENT cognom    (#PCDATA)>
<!ELEMENT categoria (#PCDATA)>
<!ELEMENT preu      (EMPTY)>
<!ATTLIST preu      quantitat  CDATA  #REQUIRED
                    moneda     CDATA  #REQUIRED>
```

Exemple 3.3. DTD corresponent al document XML de l'exemple 3.2.

El DTD ens indica que l'element `llibre` té com a fills una ocurrència dels elements `titol`, `autor`, `preu` i cap, una o més ocurrències de l'element `categoria`. D'altra banda, els elements `nom` i `cognom` són fills de l'element `autor`. També ens indica que els elements `autor, nom, cognom` i `categoria` contenen text i que l'element `preu`, no conté dades però té associats els atributs `quantitat` i `moneda`, que contenen text i són obligatoris.

La definició del tipus de document és pot incloure en el propi document XML o bé, es pot utilitzar un fitxer extern. En aquest darrer cas, en la declaració del tipus del document (instrucció `<DOCTYPE>`), és fa

referència al fitxer extern que conté el DTD (fitxer amb extensió .dtd). Els exemples 3.4 i 3.5 mostren aquestes dues formes d'incloure el contingut del DTD dins del document XML.

```
<?xml version="1.0"?>
<!DOCTYPE persona[
     <!ELEMENT persona   (nom, cognom, professio*)>
     <!ELEMENT nom       (#PCDATA)>
     <!ELEMENT cognom    (#PCDATA)>
     <!ELEMENT professio (#PCDATA)>
]>
<persona>
   <nom>Jordi</nom>
   <cognom>Morell</cognom>
   <professio>Professor</professio>
   <professio>Escriptor</professio>
</persona>
```

Exemple 3.4. Document XML amb el DTD intern.

```
Document XML:

<?xml version="1.0"?>
<!DOCTYPE persona SYSTEM "/dtds/persona.dtd">
<persona>
   <nom>Jordi</nom>
   <cognom>Morell</cognom>
   <professio>Professor</professio>
   <professio>Escriptor</professio>
</persona>


Fitxer "persona.dtd":

<!ELEMENT persona   (nom, cognom, professio*)>
<!ELEMENT nom       (#PCDATA)>
<!ELEMENT cognom    (#PCDATA)>
<!ELEMENT professio (#PCDATA)>
```

Exemple 3.5. Document XML amb el DTD extern.

Dins d'un DTD ens podem trobar amb aquests tipus de declaracions:

- Declaracions de tipus d'elements
- Declaracions de llistes d'atributs per als tipus d'elements
- Declaracions d'entitats
- Declaracions de notacions

Tot seguit descriurem cadascun d'aquests tipus de declaracions i posteriorment, veurem un DTD que inclogui declaracions dels diferents tipus (exemple 3.6).

Declaracions de tipus d'elements

Les declaracions de tipus d'elements permeten definir:

- Els tipus d'elements que poden aparèixer en el document XML.
- El contingut dels tipus d'elements definits.

D'aquesta manera, els processadors XML al validar els documents, poden conèixer quins elements estan permesos i poden comprovar si el seu contingut és correcte o no.

Totes les declaracions de tipus d'element tenen la forma:

```
<!ELEMENT nom_del_element (tipus_del_contingut)>
```

La següent taula resumeix breument els possibles tipus de contingut que pot tenir un element:

Tipus	**Descripció**	**Exemple**
EMPTY	Element buit, sense contingut. Possiblement la informació estarà en els seus atributs.	`<!ELEMENT llista EMPTY>`
#PCDATA	L'element només pot contenir caràcters de text.	`<!ELEMENT nom (#PCDATA)>`
ANY	L'element pot contenir tant text com altres elements declarats prèviament.	`<!ELEMENT paràgraf ANY>`

Altres elements	L'element conté un o més elements. Poden ser seqüències d'elements, una alternativa d'elements o repetició d'elements ('?': pot aparèixer un cop o no aparèixer, '+': un o més cops, '*': zero o més cops).	`<!ELEMENT persona (nom, cognom)>`
		`<!ELEMENT unitat (metre | gram)>`
		`<!ELEMENT persona (nom, edat?, professio*)>`

Combinació de diferents tipus	L'element pot contenir tant caràcters de text com combinacions d'altres elements.	`<!ELEMENT paràgraf (#PCDATA, nom, edat?, professio)*>`

Declaracions de llistes d'atributs

Les declaracions de llistes d'atributs pels elements, defineixen:

- El conjunt d'atributs que pertanyen a cada tipus d'element. Cada atribut està lligat sempre a un element.
- Estableixen restriccions en el contingut d'aquests atributs, a través de tipus predefinits.
- Aporten informació sobre la naturalesa dels atributs (atributs obligatoris, voluntaris, de valor fix o bé amb valors per defecte).

Les declaracions de llistes d'atributs tenen la forma:

```
<!ATTLIST  nom_del_element  nom_del_atribut  tipus  valor_per_defecte>
```

La següent taula descriu breument els tipus d'atributs existents:

Tipus d'atribut	Descripció
CDATA	L'atribut és del tipus cadena de text.
ID	L'atribut d'aquest tipus serveix per donar un nom a l'element, que l'identifica de forma única.
IDREF, IDREFS	Aquest tipus d'atribut s'utilitza per indicar que l'atribut és una referència a un element identificat mitjançant un atribut del tipus ID.
ENTITY, ENTITIES	El valor de l'atribut és una o més entitats declarades anteriorment en el DTD.
NMTOKEN, NMTOKENS	El valor de l'atribut és un o més tokens XML (cadenes de text que poden començar per qualsevol caràcter).
NOTATION	El valor de l'atribut és una notació declarada anteriorment en el DTD.
ENUMERATION	Indica una llista de possibles valors per l'atribut.

Cada declaració d'atribut inclou una declaració de valors per defecte. Existeixen quatre possibilitats:

Declaració per defecte	Descripció
`#REQUIRED`	S'ha d'indicar obligatòriament un valor per l'atribut.
`#IMPLIED`	L'atribut és opcional. L'element pot indicar un valor per l'atribut o no.
"valor per defecte"	L'atribut s'inicialitza amb el valor per defecte indicat. Si no s'indica cap valor, l'atribut tindrà aquest valor.
`#FIXED` "valor per defecte"	L'atribut té el valor fixat i no és pot modificar.

Declaracions d'entitats

Com hem vist al final de l'apartat 3.1, les entitats constitueixen les unitats físiques del document XML i poden ser de dos tipus: entitats generals o entitats de paràmetre.

Les declaracions de les entitats generals tenen la forma:

```
<!ENTITY nom_entitat "valor_entitat">    (entitat general internes)
<!ENTITY nom_entitat SYSTEM "uri"?>      (entitat general externa)
```

Les referències a les entitats generals són de la forma: `&nom_entitat`, on `nom_entitat` és el nom de l'entitat declarada de manera que, el processador, substituirà la referència pel text entre cometes de la declaració (entitats internes) o bé, pel contingut remot indicat en les entitats generals externes.

Les declaracions de les entitats externes no analitzables, com són fitxers d'imatges o sons, són de la forma:

```
<!ENTITY nom_entitat SYSTEM "uri" NDATA nom_de_notacio ?>
```

on `uri` indica on és troba el recurs i `NDATA` serveix per indicar el nom de notació que se li assigna a aquest tipus d'entitat. Això servirà per poder indicar al processador quina aplicació podrà utilitzar per tractar la entitat.

Declaracions de notacions

Per últim, en els DTD també es poden declarar notacions. Les notacions bàsicament, identifiquen un nom XML amb una aplicació auxiliar capaç de processar certes dades i que podrà ser utilitzada pel processador XML per tal de realitzar alguna tasca. També s'utilitzen per realitzar negociacions de contingut.

La declaració de les notacions té aquesta forma:

```
<!NOTATION nom_de_la_notació SYSTEM "uri_aplicacio">
<!NOTATION nom_de_la_notació PUBLIC "id_publica"  "uri_aplicacio">
```

Per exemple, si hem definit com a entitat externa un fitxer que conté una presentació amb PowerPoint, podríem declarar una notació de nom "ppt" que indiqui al processador XML que utilitzi l'aplicació "Microsoft PowerPoint" per visualitzar-lo.

En l'exemple 3.6 podem veure un DTD complert que conté declaracions dels quatre tipus que acabem de veure.

```
<?xml version="1.0"?>
<!DOCTYPE llista_pacients[
      <!NOTATION jpg  SYSTEM "programa_fotos.exe">
      <!ENTITY   F100 SYSTEM "fotos/P100.jpg" NDATA jpg">
      <!ELEMENT  llista_pacients (pacient)*>
      <!ELEMENT  pacient         (nom,cognom)>
      <!ELEMENT  nom             (#PCDATA)>
      <!ELEMENT  cognom          (#PCDATA)>
      <!ATTLIST  pacient  id       ID          #REQUIRED
```

```
                          id_pare  IDREF       #IMPLIED
                          id_mare  IDREF       #IMPLIED
                          sexe     (home|dona) #REQUIRED
                          d_naix   NMTOKEN     #REQUIRED
                          poblacio CDATA       #IMPLIED
                          fotos    ENTITIES    "">
]>
<llista_pacients>
  <pacient id="P90" id_pare="P67" id_mare="P75" sexe="dona"
           d_naix="30-04-1975">
       <nom>Maria</nom>
       <cognom>Calderó</cognom>
  </pacient>
  <pacient id="P100" sexe="home" d_naix="30-04-1975" poblacio="Lleida"
           fotos="F100">
       <nom>Josep</nom>
       <cognom>Lopez</cognom>
  </pacient>
</llista_pacients>
```

Exemple 3.6. Document XML vàlid amb un DTD que inclou declaracions d'elements, atributs, notacions i entitats.

3.3 Esquemes XML

Els esquemes XML (*XML Schema Definition Language*) és un llenguatge XML per descriure i restringir el contingut dels documents XML. Aquest llenguatge va ser definit per W3C com una recomanació formal per la descripció del tipus de document. De vegades, els esquemes XML s'anomenen com la nova generació dels DTD. Han sorgit de l'experiència d'utilitzar els DTD que, com hem vist en el punt anterior, no utilitzen la sintaxi pròpia de XML i tenen poca capacitat per a restringir els documents XML.

Per tant, la funció dels esquemes XML és definir i restringir el contingut i l'estructura de document XML en XML, i per tant, substituir els DTD per esquemes XML.

Les principals característiques dels esquemes XML són les següents:

- Facilita el "mapatge" automàtic amb estructures de dades d'un programa, ja que permet la definició dels tipus de dades.
- Afegeix certa lògica a les declaracions.
- Permet la integració de l'espai de noms.
- Al basar-se en sintaxi XML, el codi de llibreries o parses per a processar els documents XML pot ser més compacte.

Altres característiques que s'han incorporat a partir d'experimentar les limitacions dels DTD:

- Tipus de dades definits pel propi XML Schema (*string, boolean, real, decimal, integer, date, time, dateTime, binary*, etc.).
- Tipus de dades definits per l'usuari.
- Possibilitat d'utilitzar indicadors d'ocurrència (*minOccurs, maxOccurs*).
- Permet la importació/exportació d'elements.
- Refinament: herència entre els elements de l'esquema (per exemple, persona => treballador).
- Control d'extensions (*open, refinable, closed*).

Les restriccions que es poden introduir en un esquema XML són les següents:

- Control sobre les dades: longitud de la cadena (*length*), màxima longitud permesa (*maxlength*), representacions possibles (per exemple, "*dd-mm-yyyy*"), enumeracions (*enumeration*), màxim possible (*maxInclusive*), mínim possible (*minInclusive*), màxim exclusiu (*maxExclusive*), mínim exclusiu (*minExclusive*), dígits amb part decimal (*scale*), etc.
- Control sobre el refinament: mecanismes d'herència i extensió.

- Control sobre l'extensibilitat: oberta, refinable, tancada.

Tot seguit mostrem un document XML i els seus corresponents DTD i XML esquema, per poder veure la diferència entre ells:

Document XML:

```
<?xml version="1.0">
   <llibre isbn="123456">
   <títol>Laia</títol>
   <autor>Salvador Espriu</autor>
   <any>1968</any>
   <preu>8,90</preu>
</llibre>
```

DTD:

```
<!ELEMENT llibre (títol, autor+, any?, preu)>
<!ELEMENT títol  (#PCDATA)>
<!ELEMENT autor  (#PCDATA)>
<!ELEMENT any    (#PCDATA)>
<!ELEMENT preu   (#PCDATA)>
<!ATTLIST llibre isbn ID #REQUIRED>
```

Esquema XML:

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
<simpleType name='linia'>
   <restriction base='string'><maxLength value='50'/></restriction>
</simpleType>
<simpleType name='segleXX'>
   <restriction base='integer'>
       <minInclusive value='1900'/>
       <maxInclusive value='1999'/>
   </restriction>
</simpleType>
<element name='llibre' type='TipusLlibre'/>
<complexType name='TipusLlibre'>
  <sequence>
     <element name='titol' type='linia'/>
     <element name='autor' type='linia'/>
     <element name='any' type='segleXX'/>
     <element name='preu' type='divisa'>
       <complexType><attribute name='divisa' type='string'/></complexType>
     </element>
  </sequence>
  <attribute name='isbn' type='ID'/>
</complexType>
</schema>
```

Exemple 3.7. Comparació Esquema XML i DTD d'un document XML.

L'esquema anterior és un document XML amb un únic element esquema (*schema*) on s'han definit els tipus de dades específiques per aquest esquema: el nou tipus de dades "*linia*" com una cadena de llargada limitada a 50 caràcters, el nou tipus "*segleXX*" com un enter restringit entre els valors 1900 i 1999 inclosos tots dos, i el nou tipus complex "*TipusLlibre*" com a seqüència de diversos elements. Finalment, s'afirma que els documents conformats en base a aquest esquema tindran l'element "*llibre*" de tipus "*TipusLlibre*" com a element arrel.

Com veiem, l'esquema XML resulta més llarg que un DTD però, en canvi, la verificació de les dades és més rigorosa i, per tant, l'exportació i importació de dades en format XML és més senzilla per a qualsevol que faci servir un processador de documents XML, que prèviament validi el document contra un esquema.

4 Tecnologies de transformació i presentació

XML descriu l'estructura del document però no defineix les propietats de presentació i visualització de les dades que conté. Això és deu a que, justament un dels requisits de l'especificació XML va ser no incloure capacitat de format dins la sintaxis de les marques XML, per tal de separar el contingut de la presentació en els documents XML. Això va significar la necessitat d'utilitzar altres mètodes per poder definir i modificar la forma en que es visualitzen les dades.

Les fulles d'estil ens proporcionen aquest mètode, doncs contenen les descripcions de com s'han de mostrar els elements d'un document XML. Mitjançant les fulles d'estil es pot variar la forma en que es visualitzaran les dades sense necessitat de modificar el contingut, permetent així adaptar la seva visualització en múltiples formats.

Actualment, existeixen principalment dos tipus de fulles d'estil disponibles per poder definir el format dels documents XML:

- Les fulles d'estil en cascada CSS.
- El llenguatge d'estil extensible XSL.

4.1 CSS

Les fulles d'estil CSS són un estàndard desenvolupat pel W3C amb l'objectiu de proporcionar un mecanisme per modificar la visualització de les pàgines web, sense necessitat d'haver d'afegir a l'HTML noves etiquetes de format. Posteriorment, l'estàndard CSS també ha estat adaptat com a solució inicial per tal d'especificar la presentació dels documents XML.

Un fulla d'estil CSS consisteix en un conjunt de regles d'estil que s'apliquen als diferents elements del document HTML o XML que es vol visualitzar. Per cada element s'especifica una llista amb els diferents estils que es volen aplicar sobre l'element. Els estils són aplicats en cascada, és a dir, podem utilitzar múltiples estils i el navegador segueix les regles en cascada per determinar la prioritat i resoldre els conflictes.

La definició d'una regla consisteix en:

- U selector que indica l'element o elements als que s'aplicarà l'estil.
- Una declaració que descriu les propietats que conformen l'estil. Cada propietat s'identifica mitjançant el nom de la propietat i el valor que pren.

Així doncs, l'estructura bàsica d'una regla d'estil CSS és de la forma:

Etiqueta {
nom_propietat1: valor1;
nom_propietat2: valor2;
...
nom_propietatn: valorn;
}

Algunes de les propietats que es poden especificar dins dels estils són:

- de posició: definició de la posició dels elements, com s'ha de mostra, etc.
- de tipus de lletra: definició de la font de text, de la mida, del seu estil, etc.
- de color i fons: definició del color del text i del color de fons.
- de text: alineació dels elements de text, espai entre caràcters, etc.
- d'enquadrament: definició dels marges, de mida de l'element, etc.

Per indicar en un document XML quina és la fulla d'estil CSS que ha d'utilitzar, s'ha d'afegir-hi la següent instrucció de processament:

```
<?xml-stylesheet type="text/css" href="nom_del_document.css"?>
```

L'exemple 4.1 ens mostra la fulla d'estil CSS (agenda.css) que defineix com s'ha de visualitzar el document XML corresponent a una agenda (agenda.xml). Com podem veure, es defineix l'estil que s'ha d'aplicar a l'element agenda i posteriorment,

```
Document "agenda.css":

   agenda
    {
      display: block;
      border: 2px solid black;
      padding: 1em;
      background-color: #555555;
      color: #FFFFDD;
      font-family: Arial, Helvetica, sans-serif;
      font-style: italic;
      text-align: center;
     }

   nom
    {
      display: block;
      font-size: normal;
      font-style: bold;
      width: 30em;
     }

Document "agenda.xml":

<?xml versio="1.0"?>
<?xml-stylesheet type="text/css" href="agenda.css"?>
<agenda>
    <contacte>
        <nom>Montse Canals</nom>
        <telef>669988456</telef>
        <domicili>Prat de la Riba, 20</domicili>
    </contacte>
</agenda>
</xml>
```

Exemple 4.1. Definició de la fulla d'estil CSS "agenda.css" pel document XML "agenda.xml".

4.2 XSL:

XSL és l'estàndard definit per W3C que té per objectiu aplicar format als documents XML. Les fulles d'estil XSL afegeixen a un document XML una descripció de com s'ha de visualitzar el document i inclouen una certa capacitat de procés i transformació del contingut del document.

Mitjançant les fulles d'estil XSL el contingut d'un document font XML pot ser mostrat de diferents formes o adoptar diferents formats en funció del dispositiu de sortida (telèfon mòbil, dispositiu PDA, pàgina web, etc.).

XSL consta de dos parts:

- Un llenguatge de transformació: XSLT. mitjançant el qual un document XML és pot transformar en un altre document XML.

- Un llenguatge de formateig, XSL-FO, un vocabulari XML que defineix un conjunt d'elements anomenats Objectes de formatge i atributs.

4.2.1 XSLT

XSLT (*Extensible Stylesheet Language Transformations*) és un llenguatge de programació que permet especificar com un document XML és transformat en un altre document de text que pot ser o no, un altre document XML.

Un processador XSLT llegeix com a entrada un document XML i la plantilla XSLT que té associada (que és tracta també d'un document XML, ja que XSLT és una aplicació XML) i produeix un arbre resultant com a sortida. La clara separació entre el document font i l'arbre resultant, compleix amb l'objectiu bàsic de XML, de separar el contingut del document de la seva presentació, permetent que tant la gramàtica com l'estructura del document XML font, romanguin independents de l'estructura i del llenguatge de presentació.

L'estructura bàsica d'una plantilla XSLT és la següent:

```
<xsl:stylesheet version="1.0" xmlns:xls=http://www.w3.org/TR/WD-xsl">

   <xls:template match="expressio">
       <!—acció de la plantilla -->
   </xsl:template>

</xsl:stylesheet>
```

L'element `<xsl:stylesheet>` contindrà un conjunt d'elements dels tipus `<xsl:template>` que defineixen les diverses plantilles de sortida utilitzades en la fulla d'estil. Mitjançant l'expressió indicada en l'atribut *match* es selecciona la part de l'arbre a la que s'haurà d'aplicar l'acció de la plantilla. Normalment, per especificar aquesta expressió, s'utilitza el llenguatge XPath, que descriurem més endavant en l'apartat 5.3.

XSLT proporciona diferents elements que permeten definir les regles, patrons i accions que s'han d'aplicar sobre els elements del document XML origen. Cada element del document XML és comparat amb els patrons definits en la plantilla XSLT, i en el cas que coincideixi, es incorporat en l'arbre resultant d'acord amb l'acció associada al patró corresponent.

Tot seguit, expliquem molt breument alguns dels elements XSL i posteriorment, veurem la seva aplicació mitjançant un exemple:

- *<xsl:stylesheet>*: Aquest és el primer element XSL de tota la fulla d'estils i només hi pot aparèixer un cop. Mitjançant els seus atributs podem expressar la versió XSLT utilitzada en la fulla d'estils, les extensions dels espais de nom a utilitzar, etc.

- *<xsl:output>*: Defineix el tipus de sortida associada a la fulla d'estils. Una fulla pot contenir múltiples elements d'aquest tipus. Així per exemple, mitjançant el seu atribut "*method*" podem indicar si la sortida generada serà "xml", "html" o "text".

- *<xsl:include>*: Permet incloure una fulla d'estils dins de la fulla d'estils actual.

- *<xsl:template>*: Permet definir les plantilles que s'aplicaran sobre les parts del document XML que compleixin amb el patró definit.

- *<xls:apply-templates>*: Selecciona un conjunt de nodes a partir de la consulta indicada en el seu atribut "*select*". Si aquest atribut no existeix, llavors es seleccionen tots els fills del node actual. Si aquest element apareix dins del cos de l'element *<xsl:template>*, estem indicant al processador XSL que s'han de processar la resta de patrons que existeixen en la fulla d'estil. Només s'aplicaran aquells patrons que s'ajustin al node destí, tenint en compte el node amb el que s'està treballant.

- *<xsl:value-of>*: Aquest element no manipula els nodes sinó que ens permet obtenir el seu contingut per tal de mostrar-lo.

- *<xls:for-each>*: Permet fer iteracions per tal de recórrer tots els nodes resultants d'una consulta.

- *<xls:sort>*: Permet ordenar els resultats obtinguts al processar el document original mitjançant els elements *<xls:for-each>* o *<xls:apply-templates>*.

- *<xls:element>*, *<xls:attribute>*, *<xls:text>*, *<xls:comment>*: Elements que permeten crear nous nodes de tipus element, atribut, de text o comentaris, respectivament.

- *<xls:if>*, *<xls:choose>*, *<xls:when>*, *<xls:otherwise>*: Elements XSL condicionals que permeten seleccionar els nodes que compleixen determinades condicions.

Seguidament, farem un exemple per veure com és defineix una plantilla XSLT i quin és el resultat d'aplicar aquesta plantilla sobre un document XML. Abans però de mostrar la plantilla XSLT, mostrem el document XML sobre el que s'aplicarà:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<notes>
  <alumne>
    <nom>Marc</nom>
    <cognom>Gallardo</cognom>
    <nota-teoria>7,5</nota-teoria>
    <nota-practica>6,4</nota-practica>
  </alumne>
  <alumne>
    <nom>Sílvia</nom>
    <cognom>Coutiño</cognom>
    <nota-teoria>7,5</nota-teoria>
    <nota-practica>7,9</nota-practica>
  </alumne>
  <alumne>
    <nom>Maria</nom>
    <cognom>Rovira</cognom>
    <nota-teoria>7,5</nota-teoria>
    <nota-practica>8,8</nota-practica>
  </alumne>
</notes>
```

Plantilla XSLT:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl=http://www.w3.org/TR/WD-xsl>
<xsl:ouput method="html"/>
  <xsl:template match="/">
     <HTML>
        <BODY>
          <H2>Patró aplicat al node arrel</H2>
          <xsl:apply-templates select="notes/alumne" />
        </BODY>
     </HTML>
  </xsl:template>
  <xsl:template match="alumne">
     <P><B>Nom i Cognoms:</B>
        <xsl:value-of select="nom" />,
        <xsl:value-of select="cognom" /></P>
     <P><B>Notes:</B>
        <xsl:value-of select="nota-teoria" />,
        <xsl:value-of select="nota-practica" /></P>
  </xsl:template>
</xsl:stylesheet>
```

Exemple 4.2. Plantilla XSLT que aplica patrons al document XML d'entrada.

El resultat que obtindrem si apliquem aquesta fulla d'estil XSLT sobre el document XML, serà una pàgina HTML que contindrà un llistat d'alumnes, indicant per cadascun d'ells, les notes que ha obtingut. Com podem veure, la plantilla XSLT defineix dos patrons: el primer patró, s'aplica sobre el node arrel del document, genera la pàgina HTML i amb l'element *<xsl:apply-templates>* indica que s'apliquin la resta de patrons (en aquest cas, el segon patró) sobre els nodes *notes/alumne*. El segon patró, s'aplica sobre els nodes "*alumne*" i utilitza l'element XSL *<xsl:value-of>* per obtenir els valors dels nodes fills de l'alumne (*nom*, *cognom*, *nota-teoria*, *nota-practica*), per tal d'incloure'ls en la pàgina HTML resultant.

4.2.2 XSL-FO

A diferència de les fulles d'estil CSS, XSL-FO és una complexa aplicació XML que serveix per descriure de forma precisa l'aparença de les dades d'un document XML. Normalment, no s'escriu XSL-FO directament, si no que s'utilitzen les plantilles XSLT per transformar un document d'entrada XML al format XSL-FO.

Un document XSL-FO descriu el comportament d'una sèrie de caixes o àrees aniuades que es van situant al llarg d'una pàgina i que contenen text i ocasionalment, imatges o d'altres continguts externs. Existeixen diferents tipus d'àrees i mitjançant els elements proporcionats per XSL-FO, es poden especificar les propietats d'aquestes àrees: la seva posició relativa o absoluta dins la pàgina, l'espaiat entre diferents àrees o regions de text, la seva mida, etc.

Mitjançant d'altres elements definits per XSL-FO, es poden especificar propietats referents al text, com per exemple, et tipus de font a utilitzar, el seu estil, la mida de la font, l'alineació del text, etc.

Tot seguit, mostrem una plantilla XSLT que genera un document XML formatejat mitjançant XSL-FO:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <xsl:template match="/">
     <fo:root>
       <fo:layout-master-set>
         <fo:simple-page-master margin-right="1in"  margin-left="1in"
                                margin-bottom="1in" margin-top="1in"
                                page-width="8.5in"  page-height="11in"
                                master-name="pagina">
            <fo:region-body/>
         </fo:simple-page-master>
       </fo:layout-master-set>
     <fo:root>
  </xsl:template>

  <xsl:template match="alumne">
      <fo:block
          font-size="16pt"
          color="gree">
         <xsl:value-of select="nom"/>,
         <xsl:value-of select="cognom"/>
      </fo:block>
  </xsl:template>

</xsl:stylesheet>
```

Exemple 4.3. Plantilla XSLT que utilitza XSL-FO per formatejar el document XML d'entrada.

Mitjançant l'element `<fo:simple-page-master>` és defineix el format de la pàgina principal indicant els seus marges. L'element `<fo:block>` és un dels element FO més importants ja que defineix el mode

en que les dades són dividides en blocs seqüencials de text. Així doncs, aquest element és utilitzat per formatejar paràgrafs, títols, taules, etc.

5 Tecnologies de navegació i accés

5.1 XLink

XLink (*XML Linking Language*) és el llenguatge que defineix la sintaxis que, basada en atributs XML, permet la creació d'enllaços entre dos o més documents. Aquests documents no han de ser obligatòriament documents XML, ja que un enllaç situat en un document XML pot enllaçar amb un altre document XML o bé, a d'altres tipus de documents no XML (fitxers JPG, documents PDF, fitxers de text, etc.).

Els enllaços poden ser *simples*, enllaços unidireccionals que són similars als típics enllaços HTML, o bé, poden ser *estesos*, enllaços més complexes i multidireccionals que mitjançant múltiples connexions, permeten relacionar diferents recursos entre sí.

Per afegir un enllaç xlink sobre un element XML, cal afegir a l'element una sèrie d'atributs que es troben definits dins l'espai de nom xlink, d'acord amb la norma "http://www.w3.org/XML/XLink/1.0" o superiors. Tot seguit, descriurem breument aquests atributs i com s'utilitzen per definir els enllaços simples i estesos.

Enllaços simples

Un enllaç simple defineix una connexió unidireccional entre dos recursos, el recurs origen i el recurs destí. El recurs origen conté el propi enllaç i és tracta sempre d'un element XML. El recurs destí és identificat per una URI i pot ser un document XML, un element concret d'un document XML, un grup d'elements o bé, altres tipus de documents com una imatge, un fitxer de so, un document PDF, etc.

Els atributs que s'han de posar en un element XML per definir-hi un enllaç simple són:

- `xlink:type`, atribut que indica el tipus d'enllaç. En aquest cas, el seu valor ha de ser "simple".
- `xlink:href`, atribut que conté la URI o URL que indica on es troba el destí de l'enllaç.

En l'exemple 5.1, es defineix un enllaç simple sobre l'element `llibre`, que enllaça a un fitxer de text que conté un resum del llibre indicat. Per fer-ho, s'afegeix l'atribut `xlink:type` amb el valor "simple" i l'atribut `xlink:href` amb la localització del fitxer. L'atribut `xmlns:xlink` s'utilitza per associar el prefix `xlink` amb l'espai de noms que es troba a l'adreça "http://www.w3.org/1999/xlink".

```
<llibre xmlns:xlink  =  "http://www.w3.org/1999/xlink"
        xlink:type   =  "simple"
        xlink:href   =  "/docs/resum_casa_espiritus.txt">
<titol>La casa de los espíritus</titol>
<autor>Isabel Allende</autor>
</llibre>
```

Exemple 5.1. Element XML amb un enllaç simple

Enllaços estesos

Els enllaços estesos descriuen un conjunt de recursos i estableixen camins de connexió entre aquests recursos. Cada camí connecta exactament dos recursos. Un recurs es pot enllaçar cap a un altre recurs o cap a un conjunt de recursos.

Els enllaçs estesos estan formats per:

- L'element que defineix l'enllaç, mitjançant l'atribut `xlink:type` amb el valor "`extended`".

- L'element o elements que defineixen els recursos (remots o locals), mitjançant l'atribut `xlink:type` amb el valor "`locator`" pels recursos remots i el valor "`resource`" pels recursos locals. Mitjançant l'atribut `xlink:label` s'identifica cada recurs. Els recursos remots també han d'incloure l'atribut `xlink:href` per indicar la URI o URL del recurs destí.
- L'element o elements que defineixen els camins entre els diferents recursos, mitjançant l'atribut `xlink:type` amb el valor "`arc`".

L'exemple 5.2 ens mostra la definició d'un enllaç estes on l'element `coleccio` és l'element d'enllaç, cada element `llibre` és un recurs remot i els elements `seguent` i `anterior` descriuen els arcs de connexió.

```
<coleccio xmlns:xlink  =  "http://www.w3.org/1999/xlink"
          xlink:type   =  "extended">

  <llibre xlink:type   =  "locator"  xlink:label= "llibre_01"
          xlink:href   =  "http://www.mybooks.com/info.jsp?id=01">
     <titol>La casa de los espíritus</titol>
     <autor>Isabel Allende</autor>
  </llibre>

  <llibre xlink:type   =  "locator"  xlink:label= "llibre_02"
          xlink:href   =  "http://www.mybooks.com/info.jsp?id=02">
     <titol></titol>
     <autor> </autor>
  </llibre>

<seguent  xlink:type="arc" xlink:from="llibre_01" xlink:to="llibre_02" />
<anterior xlink:type="arc" xlink:from="llibre_02" xlink:to="llibre_01" />

</coleccio>
```

Exemple 5.2. XML amb un enllaç de tipus estes, recursos remots i arcs de connexió.

Altres atributs

Existeixen diversos atributs opcionals que serveixen per definir el comportament de l'enllaç:

- `xlink:show`, permet indicar com es veurà el resultat obtingut al seguir l'enllaç. Els seu possibles valors són: `new` (el resultat és mostra en un finestra nova), `replace` (el resultat és mostra en la finestra activa, substituint el document actual), `embed` (el resultat s'inclou en el document origen i es processa com si fos part del propi document) i `none` (no s'especifica cap comportament).

- `xlink:actuate`, permet indicar en quin moment s'ha d'anar a buscar el destí indicat per l'enllaç. Els seus possibles valors són: `onLoad` (l'enllaç es segueix automàticament), `onRequets` (es segueix quan l'usuari l'activa) i `none` (no s'especifica cap comportament).

- `xlink:title`, permet indicar una breu descripció del contingut de l'enllaç.

- `xlink:role`, permet especificar una URI que conté una descripció més extensa sobre el contingut de l'enllaç.

En l'exemple 5.1 vist anteriorment, podríem afegir a l'element llibre els atributs `title` i `role`, i també els atributs `actuate` i `show` per indicar que el fitxer del resum del llibre és mostri quan l'usuari activi l'enllaç i, que ho faci en una finestra nova. Això ho podem veure en l'exemple 5.3.

```
<llibre xmlns:xlink   =  "http://www.w3.org/1999/xlink"
        xlink:type    =  "simple"
        xlink:href    =  "http://www.mybooks.com/info.jsp?id=01"
```

```
        xlink:title   =  "Llibre Allende"
        xlink:role    =  "http://www.mybooks.com/IsabelAllende"
        xlink:actuate =  "onRequest"
        xlink:show    =  "new">
<titol>La casa de los espíritus</titol>
<autor>Isabel Allende</autor>
</llibre>
```

Exemple 5.3. Enllaç de tipus simple amb diferents atributs de comportament.

5.2 XPointer

XPointer especifica una sintaxis no XML que permet identificar i fer referències a punts o fragments que es troben dins d'un document XML. La sintaxis de XPointer està basada en la sintaxis del XPath, que permet navegar per l'estructura lògica que forma un document XML.

Conceptualment, l'objectiu que persegueix XPointer és similar al de l'etiqueta <a> de l'HTML, que ens permet identificar un cert punt dins d'un document HTML i posteriorment, fer referència a aquest punt, afegint el fragment "#nom_etiqueta" a la URL del document d'aquesta manera:
http://www.docs.com/document.html#nom_eitiqueta.

Amb HTML, per poder enllaçar amb un determinat punt, prèviament s'ha de modificar el document per tal d'afegir una marca en aquell punt. XPointer elimina aquesta restricció permeten especificar els punts als que es vol accedir mitjançant expressions XPath com a identificadors. A més, XPointer amplia XPath proporcionant mètodes per poder seleccionar punts o fragments del document XML, que no tenen per que coincidir amb els nodes element del document, ja que també poden fer referència a atributs, contingut de text o bé, a punts o a rangs, dos nous tipus d'expressions que proporciona XPointer i que més endavant descriurem.

Així doncs, XPointer és una expressió XPath tancada entre els parèntesis del mètode *xpointer()*. En l'exemple 5.4 podem veure l'aplicació de XPointer en una URL. Al seleccionar l'enllaç, probablement el navegador mostrarà el document, situant-se en la primera ocurrència de l'element llibre. Cal dir però, que no tots els navegadors suporten XPointer i que encara no està dels tot definit el tractament que se n'ha de fer.

```
<a href = "http://www.biblio.com/llibres.xml#xpointer(//llibre[position()=1])"  >
   Primer llibre
</a>
```

Exemple 5.4. Expressió XPointer en una URL que fa referència a la primera ocurrència de l'element `llibre` *dins del document 'llibres.xml'.*

A més de les expressions pròpies de XPaht, la sintaxis de XPointer permet fer referències mitjançant:

- Seqüències de fills
- Punts
- Rangs

Seqüències de fills

XPointer permet seleccionar els elements per la posició relativa que ocupen dins l'estructura d'arbre dels document XML. Així per exemple, l'expressió: `document.xml#1/3/4`, fa referència al quart fill, del tercer fill, del primer fill del node arrel del document.

En cas de que els elements tinguin identificadors, aquests també es poden utilitzar, de manera que, l'expressió: `document.xml#p1/4`, fa referència al quart fill de l'element amb identificador "p1".

Punts i rangs

XPointer considera que entre qualsevol parell de caràcters consecutius i entre cada parell d'elements consecutius existeix un punt. El fragment de text que existeix entre dos punts és un rang. Mitjançant els punts i els rangs, es poden fer referències com per exemple, la tercera paraula del primer paràgraf , l'any d'una data, etc.

Per exemple, l'element llibre conté 8 punts, un immediatament després i un altre immediatament abans de cada tag:

```
          <llibre>* ←————————————————————— punt 1
punt 2 ——→ *<titol>Eva Luna</titol>* ←————————  punt 3
punt 4 ——→ *<autor>Isabel Allende</titol>* ←——— punt 5
punt 6 ——→ *<any>1987</any>*←————————————————— punt 7
punt 8 → *</llibre>
```

Dins de l'element *any* hi ha definits 5 punts: el primer punt entre `<any>` i `1`, el segon punt entre `1` i `9`, el tercer punt entre `9` i `8`, el quart punt entre `8` i `7` i el cinquè punt entre `7` i `</any>`. Així per exemple, l'expressió XPointer : `xpointer(//titol[position()=1]/text()/point[position()=3])`, inicialment selecciona el primer element `titol`, després agafa d'aquest node el seu contingut (node text) i finalment, selecciona del node text el punt entre el tercer i quart caràcter. Per tant, si apliquem l'expressió sobre l'exemple anterior, ens hauríem situat sobre l'espai que apareix després de `Eva`.

Com ja hem dit anteriorment, un rang és el fragment de text que està delimitat per dos punts. Un rang pot ser o no una expressió XML ben formada. Per exemple, un rang pot incloure una etiqueta d'inici d'element però no la seva corresponent etiqueta de fi. XPointer defineix quatre funcions que permeten crear rangs i obtenir el contingut dels rangs: *range*(), *range-inside*(), *range-to*() i *string-range*(). Totes elles retornen rangs generats a partir de les expressions que se'ls hi passen com a paràmetre. Així per exemple, `range(//llibre/autor)` retorna un rang format per tot l'element `autor` (incloent els tags d'inici i fi), en canvi `range-inside(//llibre/autor)` retorna un rang format només pel contingut de l'element autor (sense incloure els tags d'inici i fi d'element).

5.3 XPath

XPath (XML Path Language) és l'estàndard XML que ens permet seleccionar i fer referència a text, elements, atributs i qualsevol altre tipus d'informació que es trobi dins d'un document XML. XPath és un llenguatge molt compacte tot i que, les seves expressions són representades amb una sintaxis que no compleix amb l'estàndard XML.

XPath és un estàndard desenvolupat amb l'objectiu d'aplicar una sintaxis i un significat comú a totes les funcionalitats mútues dels estàndards XSL, XPointer i Xlink. Així doncs, per exemple, XSLT utilitza XPath per identificar o localitzar elements del document d'entrada mitjançant patrons de cerca especificats. XPointer i Xlink utilitzen expressions XPath per identificar un punt o parts especifiques d'un document XML.

XPath treballa al nivell de les estructures lògiques dels documents XML i defineix el camí de navegació a través de l'arbre de nodes que forma el document XML. Aquest arbre comença amb el node arrel, del que pengen diferents elements, i acaba amb els nodes fulla, que contenen text, comentaris, instruccions de processament o fins i tot, poden estar buits i tenir només atributs.

L'exemple 5.5 ens mostra el document XML que utilitzarem per mostrar el funcionament de XPath.

```
<?xml version="1.0"?>
<biblioteca>
  <llibre id="1">
    <titol>La casa de los espíritus</titol>
    <autor>
```

```
        <nom>Isabel</nom>
        <cognom>Allende</cognom>
    </autor>
    <categoria>novel·la</categoria>
    <preu quantitat="28,90" moneda="euro"/>
  </llibre>
  <llibre id=”2” prestec=”no”>
    <titol>Laia</titol>
    <autor>
        <nom>Salvador</nom>
        <cognom>Espriu</cognom>
    </autor>
    <categoria>novel·la</categoria>
    <preu quantitat="11,50" moneda="euro"/>
  </llibre>
</biblioteca>
```

Exemple 5.5. Document XML d'exemple per mostrar les el funcionament de XPath.

XPath diferencia els següents tipus de nodes:

- *node arrel*: s'identifica per /. No s'ha de confondre el node arrel amb l'element arrel del document. Així, en el document de l'exemple, l'element arrel és `biblioteca` que és el primer i únic node que penja del node arrel de l'arbre, que és "/".

- *node element*: qualsevol element d'un document XML es converteix en un node element de l'arbre. Tot element té el seu node pare i pot tenir nodes fills que seran altres nodes. Així doncs, per exemple, l'element `autor` és un node element que penja del node element `llibre` i que té com a nodes fills els nodes `nom` i `cognom`. Els nodes element també poden tenir associats atributs. Una propietat interessant dels nodes elements es que poden tenir identificadors únics (aquesta propietat ha d'estar indicada en el corresponent DTD) i això permet que es pugui fer referència a aquests elements d'una forma molt directa.

- node de text: Per text entenem tots els caràcters del document XML que no formen part d'una etiqueta. Un node de text no té fills, és a dir, els diferents caràcters que el formen no es consideren fills seus. En l'exemple, "`La casa de los Espitirus`", "`Isabel`", "`Allende`", etc. són nodes text.

- node atribut: qualsevol atribut d'un document XML es converteix en un node atribut de l'arbre. Els nodes atributs no es consideren fills del node element que els conté sinó, que són etiquetes afegides al node element que el descriuen. Cada node atribut consta d'un nom, un valor i un possible espai de noms. En l'exemple, `quantitat` i `moneda` són nodes atributs del node element `preu`.

- node comentari: qualsevol comentari d'un document XML es converteix en un node comentari.

- node instrucció de processament: corresponen a les instruccions de processament del document XML.

- node d'espais de noms: qualsevol instrucció d'espai de noms es converteix en un node d'aquest tipus.

XPath pot especificar aquests nodes per posició, per posició relativa, per tipus, per contingut, per determinades condicions que el node ha de complir o per d'altres criteris. D'aquesta manera, es possible direccionar nodes dins d'un arbre XML de forma dirigida. També es poden formular expressions XPath més complexes utilitzant operadors aritmètics i lògics.

Una instrucció en el llenguatge XPath s'anomena una expressió. Els dos tipus bàsics d'expressions són les crides a funcions i els camins de localització (location paths). En la seva construcció hi intervenen altres elements bàsics com són el node de context, els atributs i d'altres funcions i operadors.

- Camins de localització:
  Ens permeten navegar i moure'ns a través de l'estructura jeràrquica de l'arbre. La formació d'aquestes expressions és similar a la sintaxis utilitzada pel sistema Unix per navegar pel sistema de fitxers. Cal dir però, que tot i tenir una sintaxis similar, el significat és totalment diferent. Així doncs, el camí de localització:

  ```
  /biblioteca/llibre/autor
  ```

  en Unix fa referència a un únic directori (autor) el qual es troba dins del directori /biblioteca/llibre. En canvi, amb XPath indica tots els elements `autor` que pengen directament de qualsevol element `llibre` que a la vegada, penja de qualsevol element `biblioteca` que finalment, penja directament del node arrel /.

  Cal tenir en compte, que una expressió XPath no retorna els elements que compleixen amb el patró que representa l'expressió, sinó que retorna una llista de referències a aquests elements. Aquesta llista pot estar buida o contenir una o més referències a nodes.

- Node de context:
  Un camí de localització sempre té un punt de partida anomenat node de context (seguint amb l'analogia del sistema Unix, seria similar al directori actual). Així, a menys que s'indiqui explícitament en contrari, s'entén que el camí de localització parteix del node que s'està processant en aquell moment.

- Predicats:
  Mitjançant els predicats podem seleccionar només els nodes que compleixen una determinada condició. Els predicats s'inclouen dins d'un camí de localització entre els símbols [ ]. Així per exemple, mitjançant el predicat:

  ```
  /biblioteca/llibre[@id="2"]/titol
  ```

  només es seleccionen els elements `titol` de tots els elements `llibre` que tenen un atribut anomenat `id` i que aquest té assignat el valor 2. En el nostre exemple, només es seleccionaria el node títol del segon llibre.

- Atributs:
  Com hem vist en l'exemple dels predicats, els atributs es representen mitjançant el símbol @. Mitjançant aquest operador podem seleccionar tots o alguns dels atributs d'un determinat element i utilitzar-los per seleccionar els elements que compleixen certes condicions. Per exemple, mitjançant el predicat:

  ```
  /biblioteca/llibre[@prestec]/*
  ```

  es seleccionen tots els nodes fills dels llibres que contenen l'atribut `prestec`. Per seleccionar tots els elements fills dels llibres que tinguin l'atribut prestec amb el valor igual a "si":

  ```
  /biblioteca/llibre[@prestec="si"]
  ```

- Altres funcions i operadors:
  - comment(), text(), processing_instruction(): serveixen per obtenir o fer referència a qualsevol node comentari, node de text o node d'instrucció de processament que siguin nodes fills del node de context.
  - node(): retorna tots els nodes fills del node actual.
  - @*: retorna tots els atributs del node de context
  - position(), last(), id(): restringeixen el conjunt de nodes seleccionats basant-se en la posició que ocupa el node o bé, en el cas de la funció id(), en l'identificador únic assignat al node.
  - operadors lògics and, or i not.

5.4 XQuery

De tots els llenguatges i estàndards desenvolupats pel W3C que tenen com a objectiu navegar, manipular i seleccionar les dades dels documents XML, XQuery és sens dubte, el llenguatge més potent i robust de tots. Per aquest motiu, dedicarem tot un capítol (concretament, el capítol 7) per fer una introducció al llenguatge XQuery, descrivint el model de dades en que es basa i veient la seva sintaxis mitjançant diversos exemples.

6 Tecnologies de suport per l'anàlisi i manipulació

6.1 DOM

DOM (Document Object Model) és una API, és a dir, una interfície de programació per al desenvolupament d'aplicacions, orientada a objectes i dissenyada per a treballar amb document HTML vàlids i documents XML ben formats.

Per a DOM, un document és un conjunt d'objectes que es relacionen entre sí mitjançant una jerarquia d'objectes en forma d'arbre. DOM permet l'accés tant als objectes com a les relacions entre ells. Els objectes es construeixen a partir d'elements, atributs, entitats, instruccions de procés, etc. i les relacions venen definides en el propi document donant lloc a elements pare, elements fills, etc.

Les principals utilitats que proporciona DOM són:

- Construcció i modificació de nous documents: la construcció de documents es pot fer partint de zero o bé, a partir d'altres document ja que DOM permet incloure fragments d'un document dins d'un altre document.
- Navegació i modificació de la jerarquia del document: és possible extreure informació de les relacions entre objectes (quins són els elements fills d'un element, qui és el pare, etc.). També és possible modificar aquesta estructura, eliminant nodes de l'arbre, afegint-ne de nous, movent-los de lloc, etc.
- Gestió dels objectes: els objectes representen certes qualitats, com per exemple, el valor d'un atribut, que es poden consultar i modificar.

DOM crea una representació del document en la memòria. Això presenta l'avantatge de poder modificar els elements directament, eliminant o inserint elements o el seu contingut, però d'altra banda, presenta el l'inconvenient de consumir gran quantitat de recursos, de tems i de màquina.

DOM és una especificació del W3C i està reconegut com a estàndard internacional. En l'actualitat existeixen dos nivells o versions del DOM i s'està treballant en una tercera versió:

- DOM nivell 1, recomanació Octubre de 1998: http://www.w3.org/TR/rec-dom-level-1/
- DOM nivell 2, recomanació Novembre de 2000: http://www.w3.org/TR/dom-level-2/
- DOM nivell 3, en desenvolupament: http://www.w3.org/TR/dom-level-2-Core/

DOM defineix, a través d'interfícies, els tipus d'objectes que modelaran els documents, els seus mètodes, els seus atributs i les relacions entre ells, però no proporciona cap implementació concreta. DOM s'ha dissenyat de manera que pugui ser utilitzat per una gran varietat de llenguatges de programació, el que facilita que hi hagi implementacions en Java, Perl, C++, JavaScript, etc.

6.2 SAX

SAX és un conjunt d'interfícies i classes que ofereixen una eina molt útil per al processament de documents XML. De la mateixa forma que DOM, SAX és independent de plataforma i del llenguatge de programació utilitzat però, a diferència de DOM, és una API basada en *events*. Entenem com a events, per exemple, el fet de trobar l'etiqueta d'inici d'un element, el valor d'un atribut, etc.

El parssejador que implementa SAX genera events cada cop que es troba una nova marca XML, un error perquè el document no està ben format, etc. Cada vegada que al llegir el document XML, es troba un determinat event s'invoca un mètode capaç de tractar-lo.

SAX no requereix la generació d'estructures internes, això implica que, al no tenir que emmagatzemar en la memòria els events que succeeixen, no és pot retrocedir i per tant, el tractament dels events s'ha de realitzar en el moment en que tenen lloc. Generalment, es necessari tenir algun coneixement sobre el passat i per fer-ho, és sol utilitzar alguna variable o estructura per guardar aquesta informació.

SAX és útil en aplicacions en les que s'ha de comptar les vegades que succeeix un element, en les que és vol obtenir el valor d'un element o atribut concret, en les que és vol buscar una certa informació, etc., i en general, per aquelles aplicacions en les que no es necessari conèixer, navegar o modificar l'estructura d'un document XML.

Existeixen dos versions de SAX:

- Especificació SAX 1.0
  XML-DEV Mailing List Member Specification, Megginson Technologies , 11 de Maig de 1998
  http://www.megginson.com/SAX/SAX1/index.html

- Especificació SAX 2.0
  XML-DEV Mailing List Member Specification, Megginson Technologies , 5 de Maig de 2000
  http://www.megginson.com/SAX/index.html

Un dels principals avantatges de SAX comparant amb una solució DOM, és que amb SAX es poden analitzar sintàcticament documents molt més extensos que la memòria disponible de la màquina. A més, SAX proporciona els resultats molt més ràpid en qualsevol situació en la que elements individuals s'hagin de llegir, ja que no es necessari remodelar el document cada vegada.

La següent figura mostra un document XML i la seva corresponent representació en SAX i en DOM:

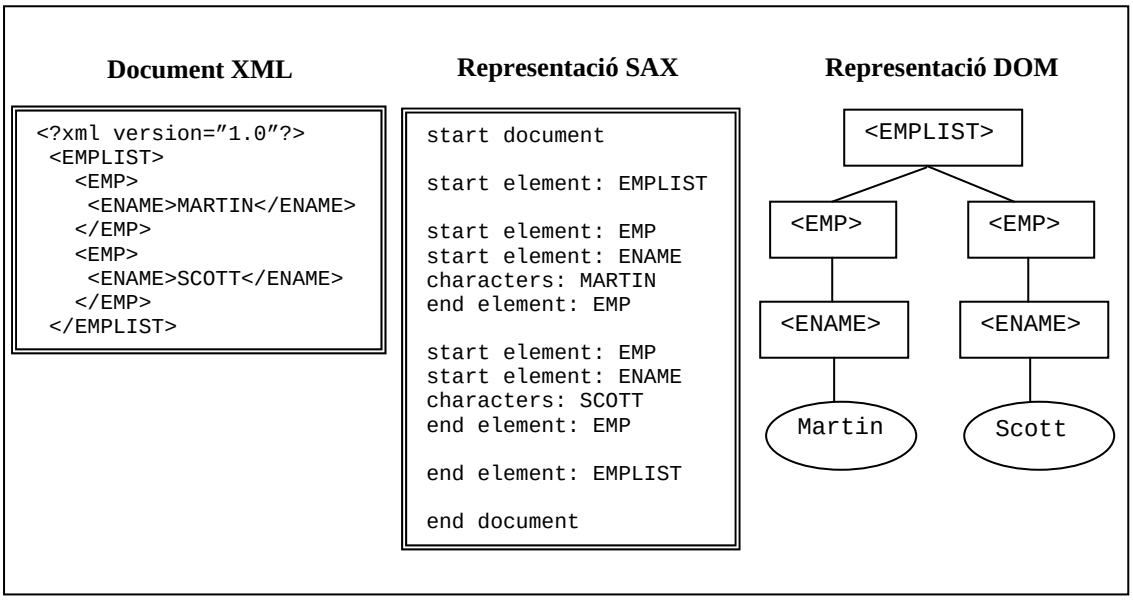


Figura 6.1. Representació SAX i DOM d'un document XML.

7 XQuery

7.1 Que és XQuery?

Com hem vist en els apartats anteriors, XML és un format extremadament versàtil que permet representar diferents tipus d'informació de fonts molt diverses. Tot i la possible complexitat de la informació emmagatzemada en XML, l'estructura bàsica de XML és molt simple de manera que, un document XML és pot representar com una estructura jeràrquica i ordenada en forma d'arbre. El llenguatge de consulta XQuery, basant-se en aquesta estructura, proporciona els mecanismes necessaris per poder interrogar, recuperar i interpretar la informació.

XQuery és un llenguatge funcional que, en lloc d'executar comandes com fan els llenguatges procedurals, es basa en expressions de manera que cada consulta és una expressió que ha de ser avaluada. Aquestes expressions poden combinar-se entre elles per tal de crear noves expressions més complexes i potents.

XQuery neix fruit de l'esforç realitzat pel Grup de Treball XML Query (creat l'octubre de 1999 pel W3C) per tal de desenvolupar un model de dades per als documents XML, un conjunt d'operadors d'interrogació per al model de dades i un llenguatge d'interrogació basat en aquest conjunt d'operadors.

Entre les característiques principals de XQuery en destaquem les següents:

- És un llenguatge funcional on les consultes es representen mitjançant expressions.
- Les expressions és poden aniuar formant expressions que donen lloc a consultes més complexes.
- Proporciona una sintaxi que és comprensible i fàcil d'escriure i entendre pels humans, en front de la sintaxis expressada en termes XML (XQueryX).
- Permet realitzar interrogacions concises i fàcils d'entendre.
- És suficientment flexible per poder interrogar una gran varietat de fonts d'informació XML, incloent documents i bases de dades.
- El disseny de XQuery està basat i influenciat per altres estàndards W3C, incloent XML Schema, XPath, XSLT i el propi XML.

És important destacar que el disseny de XQuery està encara en procés i en contínua evolució. El Grup de Treball XML Query ha publicat diversos documents que descriuen l'estat actual de disseny i que és van actualitzant a mesura que el disseny avança. Entre aquests documents destaquem:

- *"XQuery 1.0: An XML Query Language"*, conté la sintaxi de XQuery i una descripció informal del llenguatge.
- *"XQuery 1.0 and XPath 2.0 Data Model*", defineix el model de dades sobre el que es basen totes les expressions XQuery i XPath.
- "*XQuery 1.0 and XPath 2.0 Formal Semantics*", defineix formalment la semàntica de XQuery 1.0 i XPath 2.0.
- "*XQuery 1.0 and XPath 2.0 Functions and Operators*", conté la llibreria de funcions i operadors suportats per XQuery.
- "*XML Query Use Cases*", conjunt de casos d'ús que mostren aplicacions del llenguatge XQuery.

Com veiem, XQuery està íntimament relacionat amb XPath. De fet, XQuery 1.0 és una extensió de XPath 2.0. Qualsevol expressió que sigui sintàcticament vàlida i s'executi correctament amb XPath 2.0 i amb XQuery 1.0, haurà de retornar exactament el mateix resultat en els dos llenguatges. Donada aquesta gran proximitat entre els dos llenguatges, els seus respectius grups de treball W3C, treballen conjuntament en les seves especificacions.

En els propers apartats, presentarem una introducció al llenguatge XQuery, descrivint breument el model de dades (data model) sobre el que es basa XQuery i veient la sintaxi XQuery, així com les principals funcions i operadors que utilitza. Tot això, ho veurem amb diversos exemples. Recomanem però, que per un estudi exhaustiu de XQuery, s'hauria de recórrer als documents citats anteriorment.

7.2 Data Model

XQuery està definit en termes d'un model de dades formal (*Data Model*), no en termes de text XML. Cada entrada d'una consulta és una instància al model de dades, i la sortida de tota consulta és també una instància al model de dades. L'objectiu del model de dades XQuery és proporcionar una representació abstracta d'un o més documents XML o fragments de document.

En el model de dades XQuery cada document es representa com un *arbre de nodes*. Com ja havíem indicat quan parlàvem de XPath, els tipus de nodes possibles són: node document, element, atribut, text, espai de noms, instrucció de processament i comentaris. Els nodes tenen identitat pròpia, és a dir, cada node té un identificador únic que el distingeix de la resta de nodes, incloent aquells nodes que aparentment són iguals.

A més dels nodes, el model de dades defineix també *valors atòmics* (*atomic values*), els quals són instàncies dels tipus de dades simples definits per XML Schema, com són cadenes de caràctes, booleans, decimals, enters i dates.

Un *item* és un node o un valor atòmic. Una sèrie de items formen una *seqüència*. Per XQuery, cada valor es una seqüència i no hi ha distinció entre un únic item i una seqüència de longitud 1, és a dir, un node o valor atòmic és equivalent a una seqüència de longitud 1 que conté aquell node o valor atòmic. Les seqüències només poden contenir nodes o valors atòmics, no poden contenir altres seqüències.

Les seqüències de longitud zero són vàlides i sovint s'utilitzen per representar informació desconeguda (el seu ús és similar al del valor *null* utilitzat en els sistemes relacionals). El model de dades defineix també un valor especial anomenat *error value*, el qual és el resultat d'avaluar una expressió que conté un error. Un *error value* no és pot combinar en una seqüència amb d'altres valors.

El primer node de qualsevol document és el node document, el qual conté el document sencer. El node document no correspon a cap part visible del document sinó que representa al document en sí mateix. Els nodes element, els nodes comentari i els nodes d'instruccions de processament van apareixent seguint l'ordre en que es troben en el document XML. Els nodes element apareixen sempre abans dels seus nodes fills, és a dir, abans dels nodes element, nodes de text, nodes comentari o nodes d'instruccions de processament que contenen. Els atributs no es consideren fills d'un element, però també segueixen la posició definida d'acord amb l'ordre del document: els atributs apareixen després de l'element en que es troben i abans dels nodes fills de l'element.

Els documents XML d'entrada són transformats al model de dades XQuery mitjançant un procés anomenat *schema validation*. Aquest procés s'ocupa de parsejar el document i validar-lo d'acord amb el seu XML esquema. En cas de que el document no tingui un esquema definit, el procés assigna tipus genèrics als nodes i als valors atòmics (*anyType* per als nodes i *anySimpleType* per als valors atòmics).

El resultat d'una consulta pot ser transformat del model de dades XQuery a una representació XML mitjançant un procés anomenat *serialització*. Cal dir però, que no sempre el resultat d'una consulta és un document XML ben format, així per exemple, una consulta pot retornar un valor atòmic o un conjunt d'elements que no pertanyen al mateix pare.

7.3 Treballant amb XQuery

En aquest apartat, presentarem les principals característiques del llenguatge XQuery i mostrarem com treballa mitjançant un sèrie d'exemples. Així, entre d'altres aspectes, veurem com localitzar els nodes mitjançant expressions XPath, com crear noves estructures XML mitjançant constructors i com combinar i reestructurar la informació dels documents XML mitjançant les expressions FLWOR, les ordenacions, les expressions condicionals i els quantificadors. Després, també descriurem els operadors i les funcions de XQuery.

Per mostrar aquestes característiques de XQuery, utilitzarem el següent document XML que estructura les dades d'una biblioteca.

```
<bib>
  <llibre any="2001">
     <titol>XML in a Nutshell</titol>
     <autor><nom>Elliotte</nom><cognom>Rusty</cognom></autor>
     <autor><nom>William</nom><cognom>Scott</cognom></autor>
     <editorial>O'Reilly</editorial>
     <preu>65,95</preu>
  </llibre>

  <llibre any="2001">
     <titol>XML a través de ejemplos</titol>
     <autor><nom>Abraham</nom><cognom>Gutiérrez</cognom></autor>
     <autor><nom>Raúl</nom><cognom>Martínez</cognom></autor>
     <editorial>Ra-Ma</editorial>
     <preu>45</preu>
  </llibre>

  <llibre any="1994">
     <titol>TCP/IP</titol>
     <autor><nom>William</nom><cognom>Stevens</cognom></autor>
     <editorial>Addison-Wesley</editorial>
     <preu>48,95</preu>
  </llibre>

  <llibre any="1999">
     <titol>Aprendiendo XML</titol>
     <autor><nom>Erik</nom><cognom>Ray</cognom></autor>
     <editorial>O'Reilly</editorial>
     <preu>58,90</preu>
  </llibre>
</bib>
```

El DTD que descriu el document anterior:

```
<!ELEMENT bib (llibre*)>
<!ELEMENT llibre (titol, autor+, editorial+, preu)>
<!ATTLIST llibre any CDATA #REQUIRED>
<!ELEMENT autor  (nom, cognom)>
<!ELEMENT titol  (#PCDATA)>
<!ELEMENT nom    (#PCDATA)>
<!ELEMENT cognom (#PCDATA)>
<!ELEMENT editorial (#PCDATA)>
<!ELEMENT preu (#PCDATA)>
```

7.3.1 Conceptes bàsics

Literals i comentaris

XQuery és un llenguatge funcional *case-sensitive* basat en expressions que retornen valors i que no tenen efectes colaterals. Les expressions més simples són els **literals** i els **comentaris**. Els comentaris s'indiquen mitjançant els símbols '(:' i ':)', per exemple:

```
  (: això és un comentari :)
```

L'altre tipus d'expressions simples són els literals que representen valors atòmics. Els literals poden ser cadenes de caràcters o literals numèrics. Els literals cadena de caràcters es delimiten per cometes simples o dobles. Els literals numèrics poden ser de tres tipus: *integer*, *decimal* o *double*.

```
  "una cadena de caràcters"    (: literal cadena :)
   256                         (: un literal numèric enter :)
  -1.23                        (: un literal numèric decimal :)
```

Un literal cadena pot contenir referències a les entitats predefinides per XQuery: < (<), &gt (>), &amp (&), &quot ("") i ' ('').

Variables

Un variable XQuery és un nom que comença pel signe $. Un variable té associat un valor i és utilitzada com una expressió que representa aquest valor. Una forma de relacionar un variable amb un valor és mitjançant l'expressió `let`. Més endavant, quan parlem de les expressions FLWOR, descriurem amb més detall el funcionament de `let`. De moment, en el següent exemple, veiem com assignem el valor 1 a la variable `$inici`:

```
let $inici := 1
```

Funcions d'entrada

XQuery defineix dos funcions d'entrada per poder identificar l'origen de les dades que podran ser consultades:

- *doc()*, retorna un document sencer, identificant el document mitjançant una URI. Per ser més precisos, *doc()* retorna el node document.
- *collection()*, retorna una col·lecció que és una seqüència de nodes que estan associats a una URI. Sovint s'utilitza per identificar una base de dades que serà utilitzada en un consulta.

En el cas del document exemple (books.xml), la crida a la funció doc() retorna el document sencer:

```
doc("llibres.xml")
```

7.3.2 Localització de nodes (expressions i predicats)

La localització dels nodes és realitza mitjançant les **expressions XQuery** que estan basades en la sintaxi XPath 2.0. Com ja hem descrit en l'apart 5.3, XPath localitza els nodes identificant-los dins l'estructura d'arbre que representa el document. Una expressió XPath consisteix en una sèrie de passos separats pels caràcters '/' o '//' de manera que, el resultat de cada pas és una seqüència de nodes. El valor d'una expressió XPath és el resultat que s'obté al avaluar l'últim pas de l'expressió. Cada pas és avaluat en el context d'un node particular, anomenat node de context.

Per exemple, veiem com funciona la següent expressió XPath:

```
doc("llibres.xml")/bib/llibre
```

Aquesta expressió obra el document "llibres.xml" utilitzant la funció *doc()* i retorna el seu node document, amb */bib* es selecciona l'element *bib* (element document) i finalment amb */llibre* es seleccionen tots els elements *llibre* que pengen de l'element *bib*. Aquesta expressió consta doncs de tres passos.

Els **predicats** són condicions booleanes que seleccionen un subconjunt dels nodes obtinguts per una expressió. XQuery utilitza els caràcters '[' i ']' per indicar els predicats.

Per exemple, el següent predicat retorna només els autors pels que *nom="William"* és cert:

```
doc("llibres.xml")/bib/llibre/autor[nom="William"]
```

7.3.3 Constructors: creació de nodes

Les expressions XPath que hem vist anteriorment són molt potents, però tenen una limitació important: només poden seleccionar nodes existents. Un llenguatge necessita alguna eina que li permeti construir nous nodes element, atribut, de text, etc. i especificar els seus continguts i relacions. XQuery ofereix

aquesta propietat mitjançant diversos constructors que permeten crear nodes dels diferents tipus. Existeixen dos classes de constructors: **constructors directes**, que es basen en la sintaxi XML, i **constructors computats**, que utilitzen una notació basada en expressions tancades entre claus {}.

Constructors directes

Un constructor d'elements crea un element XML. Si el nom, els atributs i el contingut de l'element són constants, el constructor està basat en sintaxi XML estàndard i s'anomena constructor directe d'elements. Per exemple, aquí mostrem un constructor directe d'elements que crea un llibre:

```
<llibre any="1999">
   <titol>Java and XML</titol>
   <autor><nom>Brett</nom><cognom>McLaughlin</cognom></autor>
   <editorial>O'Reilly</editorial>
   <preu>68,50</preu>
</llibre>
```

En aquest exemple, els valors dels elements i dels atributs són literals de text. No obstant, en molts casos és necessari crear elements o atributs que prenen com a valors el resultat d'avaluar alguna expressió. En aquest cas, l'expressió està tancada entre claus {} per tal d'indicar que ha de ser avaluada en lloc de ser tractada com un literal de text. En el constructor, l'expressió és avaluada i reemplaçada pel seu valor resultant i pot aparèixer tant en el contingut dels elements com en els valors dels atributs. Tot seguit, mostrem un exemple que mostra l'ús de les expressions tancades dins d'un constructor d'elements:

```
<exemple>
   <p> Aquí hi ha una consulta: </p>
   <eg> doc("llibres.xml")//llibre[1]/titol </eg>
   <p> Aquí hi ha el resultat de la consulta: </p>
   <eg>{doc("llibres.xml")//llibre[1]/titol}</eg>
</exemple>
```

Aquest és el resultat que obtindríem al executar la consulta anterior:

```
<exemple>
   <p> Aquí hi ha una consulta: </p>
   <eg> doc("llibres.xml")//llibre[1]/titol </eg>
   <p> Aquí hi ha el resultat de la consulta: </p>
   <eg><titol>XML in a Nutshell</titol></eg>
</exemple>
```

El resultat d'un constructor d'elements és un nou node de tipus element, el qual té la seva pròpia identitat. Tots els atributs i nodes fills del nou node element tenen també la seva pròpia identitat, encara que aquests siguin còpies de nodes ja existents.

Constructors computats (*computed constructors*)

Un altra forma de crear nodes és utilitzar els constructors computats. Un constructor computat comença per una paraula que identifica el tipus de node que es crearà: *element* per crear nodes de tipus element, *attribute* (nodes atribut), *document* (nodes document), *text* (nodes de text), *processing-instruction* (nodes d'instruccións de processament), *comment* (nodes comentaris) i *namespace* (nodes d'espai de noms). Després de la paraula que indica el tipus de node, en el cas dels nodes que tenen nom (elements, atributs, instruccions de processament i nodes d'espai de noms), s'indica el nom que s'assignarà al nou node. Posteriorment, s'indica el contingut del node mitjançant una expressió tancada entre claus {}.

A continuació, mostrem un exemple de constructors computats que creen nodes elements i atributs.

```
element llibre {
   attribute any {"1999"}
   element titol {"Java and XML"}
   element autor {
      element nom {"Brett"}
      element cognom {"McLaughlin"}
    }
```

```
    element editorial {"O'Reilly"}
    element preu {"68,50"}
}
```

Com veiem, en l'exemple anterior el nom dels nodes està fixat. De vegades però, és necessari construir els nodes indicant-hi dinàmicament el seu nom. Per fer-ho, els constructors computats permeten indicar el nom del node mitjançant una expressió tancada entre claus. El següent exemple, crea un nou node element que té el mateix nom que l'element associat a la variable $e i que conté el doble del valor del node origen:

```
element
     { name($e)}
     { data($e)*2}
```

El constructor d'atributs pot aparèixer en aquells llocs on un atribut és vàlid, per exemple, dins d'un constructor d'elements. El següent constructor d'atribut, crea un atribut a l'element *persona* al qual fa referència la variable *$p*, de manera que, el nou atribut prendrà la forma *pare="Jordi"* o *mare="Maria"* en funció del sexe de la persona:

```
attribute
     {if $p/sexe="M" then "pare" else "mare"}
     {$p/nom}
```

El constructor de documents *document{ }* crea un node document buit de manera que, junt amb la resta de constructors, permet crear un document XML sencer.

7.3.4 Combinació i estructuració de nodes

Expressions FLWOR

Les expressions FLWOR són unes de les expressions més potents i comuns de XQuery. Són similars a les construccions SELECT-FROM-WHERE del llenguatge SQL però a diferència d'aquestes, les expressions FLWOR no es defineixen en termes de taules, files i columnes, sinó que són expressions que permeten fer iteracions i que relacionen variables amb determinats valors. Sovint, aquesta classe d'expressions s'utilitzen per relacionar les dades de dos o més documents i per reestructurar les dades. El nom FLWOR prové de les paraules *For*, *Let*, *Where*, *Ordre by* i *Return*.

En una expressió FLWOR, les sentències *for* i *let* generen una seqüència de tuples de variables que constitueixen un flux de tuples. La sentència *where* permet filtrar aquestes tuples, descartant aquelles que no compleixen la condició indicada. La sentència *order by* permet establir l'ordre a seguir per les tuples generades i finalment, mitjançant la sentència *return* es retorna el resultat obtingut. El resultat d'una expressió FLWOR és una seqüència ordenada formada a partir de la concatenació dels resultats obtinguts al avaluar cadascuna de les tuples generades.

La següent expressió FLWOR retorna el títol de tots els llibres publicats l'any 2001:

```
for $b in doc("llibres.xml")//llibre
where $b/@any = "2001"
return $b/titol
```

Aquest és el resultat obtingut:

```
<titol>XML in a Nutshell</titol>
<titol>XML a través de ejemplos</titol>
```

Sentències *for* i *let*

Com hem vist les expressions FLWOR es basen en tuples i les sentències *for* i *let* són les encarregades de generar aquestes tuples. Així doncs, una expressió FLWOR ha de tenir com a mínim una sentència *for* o *let*. En els següents exemples podem veure la diferència entre aquestes dues sentències.

Primer, utilitzem la sentència *for* per assignar valors a la variable `$i`:

```
for $i in (1, 2, 3)
return
       <tupla><i>{$i}</i></tupla>
```

Com a resultat, obtenim tres tuples, una per cada valor que pren la variable:

```
<tupla><i>1</i></tupla>
<tupla><i>2</i></tupla>
<tupla><i>3</i></tupla>
```

En canvi, farem ara el mateix exemple però utilitzant la sentència `let`:

```
let $i := (1, 2, 3)
return
       <tupla><i>{$i}</i></tupla>
```

Ara, el resultat obtingut és una única tupla formada pels tres valors:

```
<tupla><i>1 2 3</i></tupla>
```

En el següent exemple, podem veure una consulta que combina les sentències *for* i *let* per tal d'obtenir el títol i el número d'autors de cada llibre:

```
for $b in doc("llibres.xml")//llibre
let $c := $b/autor
return <llibre>{ $b/titol, <num>{ count($c) }</num>}</llibre>
```

Aquest és el resultat de la consulta:

```
<llibre>
   <titol>XML in a Nutshell</titol>
   <num>2</num >
</llibre>
<llibre>
   <titol>XML a través de ejemplos</titol>
   <num>2</num >
</llibre>
<llibre>
   <titol>TCP/IP</titol>
   <num>1</num >
</llibre>
<llibre>
   <titol>Aprendiendo XML</titol>
   <num>1</num >
</llibre>
```

Sentència *where*

La sentència *where* serveix per filtrar les tuples generades per les sentències *for* i *let*, conservant només aquelles que compleixen una determinada condició i eliminant la resta. La sentència *return* només és fa efectiva per aquelles tuples acceptades per la sentència *where*. Per exemple, per obtenir només els llibres amb un preu superior als 50 euros, executaríem la següent consulta:

```
for $b in doc("llibres.xml")//llibre
where $b/preu > 50
return $b/titol
```

Sentència *order by*

La sentència *order by* ordena les tuples abans de que la sentècia *return* sigui executada, amb l'objectiu de canviar l'ordre de sortida dels resultats. La següent consulta, obté els títols dels llibres i els ordena alfabèticament:

```
for $t in doc("llibres.xml")//titol
order by $t
return $t
```

En la sentència *order by* podem especificar:

- Un o més criteris d'ordenació, mitjançant una o més expressions.
- El sentit de l'ordenació: ascendent (*ascending*) o descendent (*descending*).
- La forma d'ordenar els valors buits: tractant-los com si fossin els valors més grans possibles (*empty greatest*) o com si fossin els valors més petits (*empty least*).

La consulta del següent exemple, ordena els autors dels llibres per nom en ordre ascendent i per cognom en ordre descendent, en cas d'haver-hi noms amb valors buits, aquests serien els primers:

```
for $a in doc("llibres.xml")//autor
order by $a/nom ascending empty greatest, $a/cognom descending
return $t
```

Sentència *return*

La sentència *return* retorna el resultat de la consulta després de ser avaluada. Pot estar formada per qualsevol expressió XQuery però molt sovint, conté elements constructors per tal de retornar nous nodes, modificar la jerarquia del documents, etc.

Per exemple, la següent consulta retorna els noms i cognoms dels autors que han escrit llibres de l'editorial "Ra-Ma" i a més, converteix els dos elements originals (*nom* i *cognom*) en un únic element (*autor*) que conté la concatenació del nom i del cognom:

```
for $b in doc("llibres.xml")//llibre
where $b/editorial = "Ra-Ma"
return <autor>{string($b/autor/nom), " ", string($b/autor/cognom)}</autor>
```

Aquest és el resultat de la consulta anterior:

```
<autor>Abraham Gutiérrez</autor>
<autor>Raúl Martínez</autor>
```

Eliminació de valors duplicats amb *distinct-values()*

Les dades sovint contenen valors duplicats. Per aquesta raó, les expressions FLWOR és poden combinar amb la funció *distinct-values()* per tal d'eliminar-los.

Així doncs, la funció *distinct-values()* extreu els valors de la seqüència de nodes resultant d'avaluar l'expressió FLWOR i crea una nova seqüència que conté només valors únics, eliminant els duplicats. Per exemple, per obtenir tots els noms dels autors però eliminant els duplicats, executaríem la consulta:

```
distinct-values(doc("llibres.xml"//autor/nom)
```

El resultat obtingut seria:

```
  Elliotte William Abraham Raúl Erik
```

Joins*: Combinació de fonts de dades amb *for* i *where

Les expressions FLWOR permeten incloure diverses sentències *for* per tal de poder combinar les dades de diferents fonts d'informació. El funcionament d'aquestes expressions és el mateix que segueixen les consultes SQL que combinen dues o més taules (incloses al *from* del *select*) per tal de combinar les dades de les diferents taules i extreure'n només aquelles que satisfan una determinada condició.

Per exemple, suposem que tenim un document XML anomenat "*critiques.xml*" que conté valoracions de llibres.

```
<valoracions>
  <entrada>
    <titol>XML in a Nutshell</titol>
    <puntuacio>5</puntuacio>
    <comentari>Explica clarament els principis bàsics del XML</comentari>
  </entrada>
</valoracions>
```

El següent exemple, obté els comentaris dels llibres que apareixen en el document "llibres.xml":

```
for $t in doc("llibres.xml")//titol,
    $e in doc("critiques.xml")//entrada
where $t = $e/titol
return <critica>{ $t, $e/comentari }</critica>
```

Aquest és el resultat:

```
<critica>
  <titol>XML in a Nutshell</titol>
  <comentari>Explica clarament els principis bàsics del XML</comentari>
</critica>
```

Expressions condicionals

Les expressions condicionals XQuery s'utilitzen de la mateixa manera que en la majoria dels llenguatges de programació (*if ... then ... else*), permetent executar unes instruccions o unes altres en funció del resultat de l'expressió de testeig.

El següent exemple, en funció del preu del llibre, mostra el títol o l'editorial del llibre:

```
for $b in doc("llibres.xml")//llibre
return
  <llibre>
    { if ($b/preu > 50)
      then $b/titol
      else $b/editorial
    }
  </llibre>
```

Tot seguit mostrem el resultat de la consulta anterior:

```
<llibre>
   <titol>XML in a Nutshell</titol>
</llibre>
<llibre>
   <editorial>Ra-Ma</editorial>
</llibre>
<llibre>
```

```
    <editorial>Addison-Wesley</editorial>
  </llibre>
  <llibre>
    <autor><nom>Erik</nom><cognom>Ray</cognom></autor>
  </llibre>
```

7.3.5 Operadors XQuery

Com hem vist en els exemples anteriors, XQuery proporciona un conjunt d'operadors. Com en la majoria de llenguatges, XQuery té operadors aritmètics i operadors de comparació. A més, degut a que les seqüències d'elements són un tipus bàsic en XQuery, aquest també proporciona un conjunt d'operadors per poder manipular seqüències de nodes. La següent taula mostra els diferents tipus d'operadors XQuery:

Operadors aritmètics	Operadors de comparació	Operadors de seqüències
+	eq, =	union
-	ne, !=	intersect
*	lt, <	except
div	le, <=	
idiv	gt, >	
mod	ge, >=	
	is, <<, >>	

Operadors aritmètics

Els operadors aritmètics treballen sobre valors de tipus numèric (*integer, decimal, float, double*) o tipus derivats d'aquests. Si l'operand d'un operador aritmètic és un node, s'obté el tipus del valor que conté. En el cas de que un dels operands sigui la seqüència buida, el resultat de l'operació és també la seqüència buida (similar al comportament del valor *null* en el llenguatge SQL).

Operadors de comparació

Els operadors de comparació permeten comparar dos valors. Els podem dividir en tres grups: operadors comparatius per valor (eq, ne, lt, le, gt, ge), operadors comparatius generals (=, !=, <, <=, >, >=) i operadors comparatius per nodes (is, <<, >>).

Els operadors comparatius per valor serveixen per comparar dos valors simples. El resultat de la comparació serà cert (*true*) si el valor del primer operand és (igual, no igual, menor, menor o igual, major o major igual) que el valor del segon operand. Altrament, el resulta de la comparació serà fals (*false*). Si un dels operands conté més d'un item o si els seus tipus no són comparables, la comparació retorna un error.

Els operadors comparatius generals permeten realitzar comparacions sobre seqüències de qualsevol longitud de manera que, cada operand és una seqüència de items. El resultat de la comparació és cert si es troba un item en el primer operand i un item en el segon operand, pels quals el resultat de la comparació que s'està aplicant, és cert. Altrament, el resultat serà fals. Si és produeix algun error comparant individualment alguns dels items, la comparació retorna l'error.

Els operadors comparatius per nodes permeten comparar dos nodes, per identitat o per la posició que ocupen dins del document. En el cas de l'operador *is*, el resultat de l'operació serà cert si els dos nodes comparats tenen la mateixa identitat (els operands fan referència al mateix node), altrament el resultat serà fals. Una comparació amb l'operador << retorna cert si el node del primer operand precedeix al node del segon operand d'acord amb l'ordre del document, altrament retorna fals. De forma similar, una comparació amb l'operador >> retorna cert si el node del primer operand segueix al node del segon operand d'acord amb l'ordre del document, altrament retorna fals.

Operadors de seqüències

Els operadors de seqüències tenen com a entrada dues seqüències de nodes (anomenem-les A i B), de manera que:
- l'operador *union* retorna una seqüència formada per tots els nodes que apareixen en la seqüència A i tots els nodes que apareixen en la seqüència B.
- l'operador *intersect* retorna una seqüència formada només per aquells nodes que apareixen tant en la seqüència A com en la seqüència B.
- l'operador *except* retorna una seqüència formada només per aquells nodes que apareixen en la seqüència A i que no apareixen en la seqüència B.

7.3.6 Funcions

XQuery proporciona a l'usuari un conjunt de funcions predefinides. Per veure una llista completa amb totes aquestes funcions, recomanem la lectura del document indicat a l'inici d'aquest capítol: "*XQuery 1.0 and XPath 2.0 Functions and Operators*".

Moltes d'aquestes funcions ja ens resulten familiars per haver-les utilitzat en altres llenguatges de programació. Així doncs, de forma similar al llenguatge SQL, ens trobem amb funcions del tipus: *min*(), *max*(), *count*(), *sum*(), *avg*(), etc. També ens trobem amb funcions per treballar amb cadenes de caràcters: *concat*(), *string-length*(), *starts-with*(), *ends-with*(), *substring*(), *upper-case*(), *lower-case*(), etc.

Dues funcions molt utilitzades són: *not*() i *empty*(). La funció *not*() permet negar una condició boolena. La funció *empty*() ens permet saber si una seqüència està buida o no. La funció *exists*() és just el contrari de la funció *empty*(), és a dir, ens diu si una seqüència conté un o més items.

XQuery també proporciona funcions que serveixen per obtenir informació associada als nodes. Per exemple, la funció *string*() retorna el valor d'un node i la funció *data*() retorna el tipus del node.

A més d'aquestes funcions predefinides, XQuery permet a l'usuari definir-se les seves pròpies funcions. Cada funció està identificada pel seu nom i en la seva definició, s'hi inclou una descripció del tipus de dades de cadascun dels seus paràmetres, així com del valor que retorna. Si la funció no té paràmetres, s'indica amb parèntesis buits. Així doncs, l'estructura bàsica d'una definició de funció és la següent:

```
 define function nom_funcio ($nom_parametre1 as tipus_parametre1, ....)
    as tipus_valor_retorn
  {
     (: -- cos de la funció -- :)
  }
```

Com a exemple, mostrem la definició d'una funció recursiva que retorna la profunditat de l'arbre que representa l'element *e* passat com a paràmetre:

```
define function profunditat($e as element())
    as integer
  {
    if (empty($e/*)) then 1
    else 1 + max(for $c in $e/* return profunditat($c))
  }
```

8 XML i les bases de dades

8.1 Bases de dades relacionals vs bases de dades XML natiu

A conseqüència de la forta evolució que ha experiment l'XML en els darrers anys, actualment trobem en el mercat nombrosos productes anomenats "*habilitats per a XML*" que suporten el format XML com a format d'entrada/sortida. Aquests productes ofereixen avantatges evidents sobre els altres productes que no suporten XML. D'altra banda però, ha sorgit un altre tipus de productes anomenats "*XML natius*" que ofereixen nombroses avantatges addicionals. Aquests productes, suporten XML en les seves estructures internes i són més escalables, confiables i més interoperables que aquells productes que només utilitzen XML com un format per a l'intercanvi de dades. A més, a mesura que el volum i la complexitat de les operacions s'incrementa, la sobrecàrrega requerida per la conversió d'XML a una altra forma de representació de dades (i viceversa) afecta seriosament la velocitat, fiabilitat i funcionalitat del sistemes habilitats per a XML. En canvi, els sistemes XML natius, ofereixen no només l'aparença XML, sinó també la realitat d'una arquitectura XML, executant les accions amb més rapidesa, més fiablement i requerint menys tasques d'administració.

Centrant-nos més en el tema de les bases de dades, tot seguit compararem les arquitectures que proveeixen una visió XML d'una base de dades relacional, amb aquelles que emmagatzemen i indexen les dades en un format intern XML natiu.

Definim una "*base de dades XML nativa*" , com aquella base de dades en la que les seves estructures internes de dades es projecten directament al format jeràrquic de XML. Els usuaris d'una base de dades XML nativa no estan obligats a diferenciar entre algun format extern "d'intercanvi" i un format intern "eficient". Tampoc s'han de dissenyar aplicacions que diferenciïn entre "dades de negoci" i "contingut de documents". En una base de dades XML nativa, aquestes distincions no són necessàries.

La majoria dels proveïdors de RDBMS (Sistemes Manipuladors de Bases de Dades Relacionals) ja proporcionen interfícies i utilitats que permeten que les dades XML puguin ser emmagatzemades en els seus sistemes relacionals de la forma menys dolorosa possible, fent ús d'interfícies que realitzen les conversions d'un format a l'altre.

A continuació enumerem les principals característiques que diferencien els sistemes de base de dades XML basats en el model relacional, amb els nous sistemes de base de dades XML natius.

- Base de dades relacionals:

  - Les dades són emmagatzemades en files i columnes, amb valors atòmics per cada cel·la.
  - S'han de definir múltiples taules enllaçades (joins) per representar relacions jeràrquiques.
  - L'ordre de les files no està definit.
  - Normalment es necessari un esquema per descriure l'estructura de les dades.
  - Normalment es necessària la unió (join) de diverses taules per poder recuperar els documents de negoci (encara que siguin senzills, no formen un sol objecte).
  - Les consultes es fan amb SQL amb ampliacions propietàries per a XML.

- Bases de dades XML natives:

  - Jerarquies aniuades d'elements.
  - Els elements són ordenats.
  - No es necessari un esquema formal.
  - Els documents ordinaris de negoci poden ser representats, emmagatzemats i recuperats com un sol objecte.
  - L'estàndard XPath proporciona un llenguatge de consulta comú (encara que limitat) per la ubicació de les dades.
  - El llenguatge de consulta XQuery proporciona un potent llenguatge de consulta basat en altres estàndards XML (XPath, Schema XML, etc.)

Els principals proveïdors de bases de dades relacions han treballat dur per oferir els mecanismes necessaris que permeten emmagatzemar XML en un RDBMS. No obstant, aquests mecanismes i

interfícies poden resultar massa complexes, lentes i poc eficients, sobretot quan es vol emmagatzemar documents i missatges XML més complexes i a mesura que s'incrementa el volum de les transaccions. La complexitat de les taules i el número de unions (joins) necessàries per adaptar a XML és poden amagar al programador, però suposen una càrrega constant per als administradors de bases de dades.

De la mateixa manera, a mesura que la visió XML de les dades i els seus estàndards siguin més àmpliament coneguts i assimilats pels usuaris finals, aquests utilitzaran consultes cada cop més complexes que seran fàcilment expressables mitjançant XPath i altres llenguatges de consulta XML (XQuery), però molt difícils d'expressar i descompondre mitjançant les consultes SQL tradicionals.

Així doncs, sembla que el sistemes de base dades XML natius, jugaran un paper molt important en les tasques de gestió de la informació estructurada dels documents XML. No obstant, el sistemes RDBMS (com Oracle) estan realitzant nombrosos esforços per adaptar el seus sistemes a les noves tendències basades en els estàndards XML.

8.2 Oracle

A mesura que l'XML i els diversos estàndards han anat evolucionant, la tecnologia Oracle XML encarregada d'integrar l'XML amb els sistemes de bases de dades relacionals Oracle, també ha anat evolucionant i madurant per tal d'adaptar el SGBD d'Oracle als nous requeriments.

Tot seguit, descriurem breument la primera proposta d'Oracle per tal de donar suport a XML en les seves bases de dades: l'**Oracle XDK** (Oracle XML Developer's Kit). Posteriorment, veurem en més detall la solució adoptada actualment per Oracle, molt més potent i que permet la integració real de l'XML a nivell intern: l'**Oracle XML DB**.

La següent figura mostra l'evolució de la tecnologia XML d'Oracle, des de Oracle 8i a Oracle 9i Rel. 2.

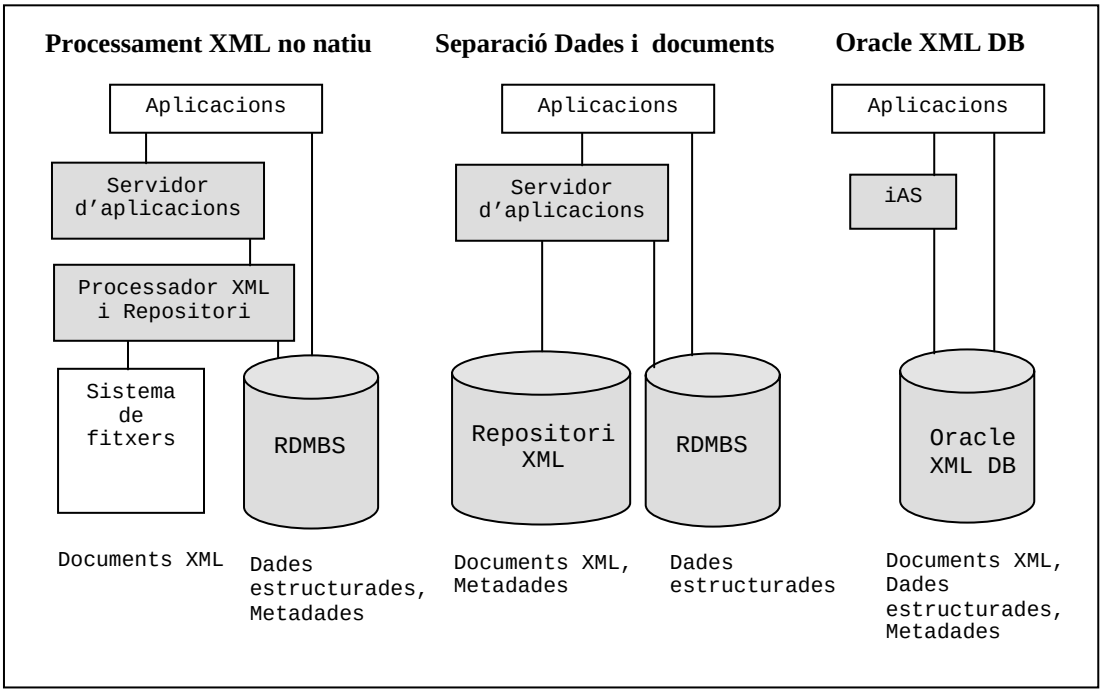


Figura 8.1. Evolució de la tecnologia XML d'Oracle.

8.2.1 Oracle XDK

Oracle va introduir el primer suport a XML en la versió Oracle8i (1999). Era una integració relativament pobra i dissenyada principalment per satisfer els requeriments d'intercanvi d'informació en format XML. Amb Oracle9i (2001), Oracle ja proporciona suport XML directament a la base de dades. El XDK (XML Developers Kit, proporciona un conjunt d'eines per al processament d'XML tant dins com fóra de la base de dades:

- *Parsejadors XML:*

  Oracle proporciona parsejadors XML per a Java, C, C++ i PL/SQL. Aquests components parsejen els documents XML per tal de que puguin ser processats per les aplicacions. Els parsejadors es basen en les interfícies proporcionades per DOM i SAX.

- *Processador XSLT:*

  Eina que permet la transformació dels documents XML a partir de fulles d'estil XSL. Utilitzant el processador XSLT, es poden transformar els documents XML a XML, HTML o qualsevol altre format basat en text.

- *Processador XML Schema*:

  El processador XML Schema parseja i valida els fitxers XML, oferint funcionalitats i característiques que no poden oferir els DTDs.

- *Generador classes XML*:

  Eina capaç de generar automàticament classes Java a partir dels esquemes XML o dels DTDs. Es especialment útil quan una aplicació vol enviar dades XML als formularis web o d'altres aplicacions.

- *XSQL Servlet*:

  El Servelt XSQL processa consultes SQL i genera com a sortida els resultats en format XML. És utilitzat pel Parsejador XML de Java i per l'XML SQL Utility per realitzar moltes de les seves operacions.

- *XML SQL Utility*:

  Eina que consisteix en un conjunt de classes Java que permeten:
  - Passar una consulta a la base de dades i generar un document XML a partir dels resultats obtinguts per la consulta.
  - Escriure dades XML a les taules de la base de dades.

  Per exemple, executant la següent consulta utilitzant l'eina XML SQL Utility:

```
    SELECT EMPNO, ENAME FROM EMP WHERE EMPNO = 7654;
```

  es genera el següent document XML:

```
<?xml version="1.0"?>
<ROWSET>
   <ROW id="1">
       <EMPNO>7654</EMPNO>
       <ENAME>Martin</ENAME>
   </ROW>
</ROWSET>
```

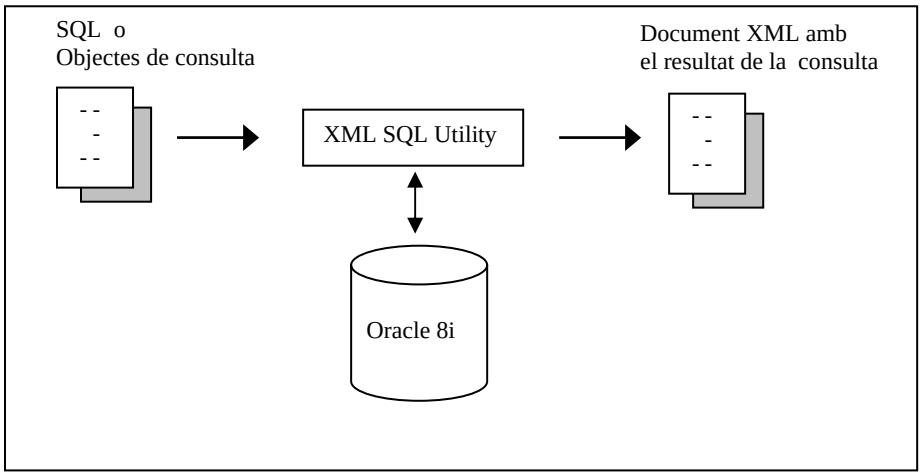


Figura 8.2. Generació d'un document XML a partir d'una consulta SQL mitjançant l'eina XML SQL Utility.

Les possibles formes d'emmagatzemar els documents XML en la base de dades són:

- Guardar el document XML com un sol objecte intacte amb els seus tags mitjançant un camp de tipus CLOB o BLOB. Aquesta és una bona opció en el cas de documents XML que contenen informació estàtica i que només s'actualitzaran substituint el document sencer, ja que són guardats en la base de dades com un tot.

- Guardar el document XML com a dades distribuïdes (sense tags) mitjançant les diferents taules relacionals. Un cop guardades, les dades poden ser fàcilment modificades i consultades mitjançant SQL.

- Combinar els documents XML i les dades guardades en taules mitjançant vistes. Amb les vistes es poden construir objectes combinant les dades XML estructurades o no estructurades de diferents formes.

8.2.2 Oracle XML DB

Oracle XML DB és la tecnologia d'alt rendiment per l'emmagatzematge d'XML disponible amb el servidor Oracle9i Rel 2. XML DB absorbeix completament el model de dades XML definit per W3C XML i proporciona nous mètodes d'accés estàndards per navegar i consultar XML. Mitjançant XML DB podem gaudir dels avantatges de la tecnologia de bases de dades relacionals més els avantatges de la tecnologia XML.

El model de dades XML contempla tant continguts no estructurats (per exemple, documents XML que són articles, notícies, etc.) com dades estructurades (per exemple, informació d'una biblioteca, una agenda, etc.).

L'arquitectura d'Oracle XML DB es pot presentar de dues formes, en funció de si les dades emmagatzemades són dades estructurades i l'accés és orientat a dades (figura 8.3) o bé, si les dades emmagatzemades són dades no estructurades i l'accés és orientat a contingut (figura 8.4).

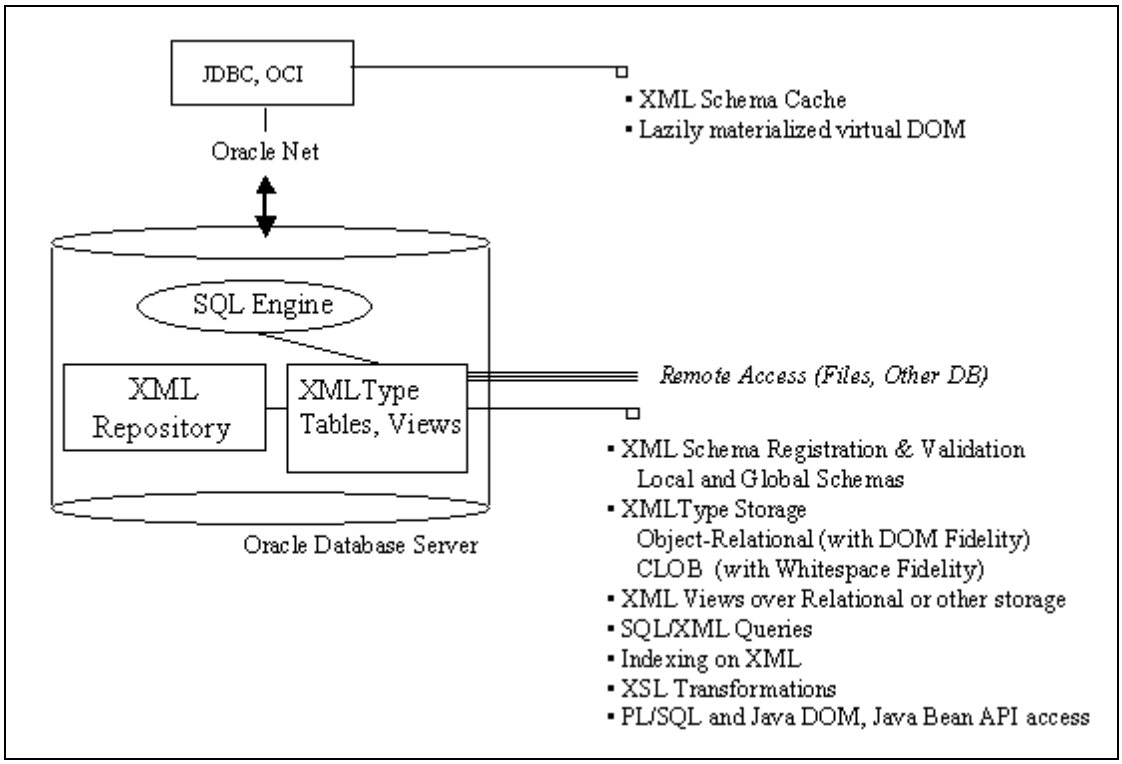


Figura 8.3. Accés orientat a dades estructurades.

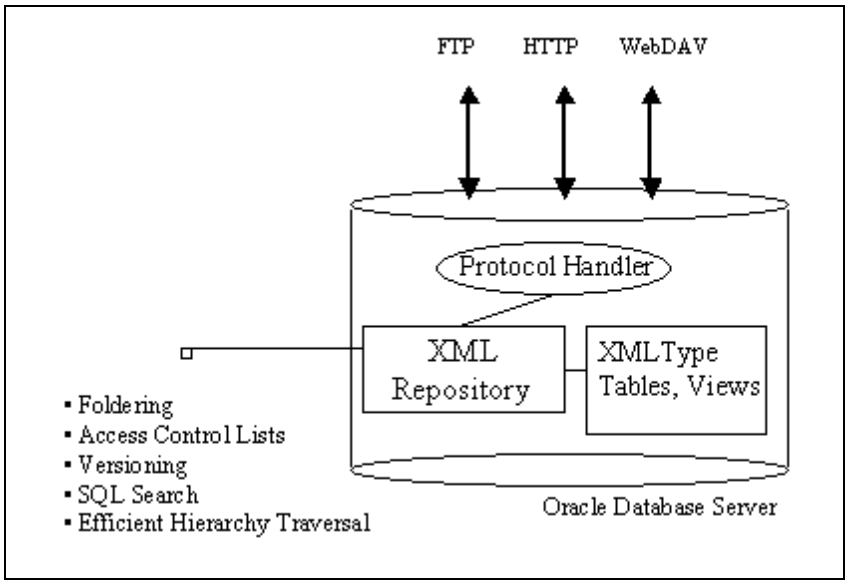


Figura 8.4. Accés orientat a contingut.

Entre els avantatges que suposa l'ús d'XML DB, en destaquem els següents:

- Proporciona una única manera d'emmagatzemar i manipular les dades, tant si es tracta de dades estructurades com no estructurades, seguint l'estàndard definit pel model de dades W3C XML. D'aquesta manera, s'aconsegueix transparència i interrelació entre XML i SQL (es poden realitzar operacions XML sobre taules i operacions SQL sobre documents XML).
- Proporciona un repositori XML dins del propi servidor de base de dades. Aquest repositori XML permet la manipulació del contingut, organitzar-lo, realitzant control d'accés, etc.

- Dóna lloc a una plataforma d'alt rendiment i gran escalabilitat per les operacions XML (que cada cop tendeixen a ser més complexes).
- Ofereix una millor manipulació de les dades no estructurades, permetent realitzar modificacions parcials, indexacions, cerques, múltiples vistes de les dades, etc.
- Permet que dades i documents procedents de sistemes de diferents naturaleses puguin ser accedides i combinades dins del model de dades XML. Aquesta integració redueix la complexitat alhora de dissenyar les aplicacions que han de treballar amb dades de diferents fonts.

Tots aquests avantatges són proporcionats gràcies als dos puntals principals d'Oracle XML DB:

- XMLType: El tipus de dades proporcionat per emmagatzemar el contingut XML.
- XML Repositori: Magatzem per manipular les dades i els documents XML.

Entre les característiques que ofereix el tipus XMLType, en destaquem les següents:

- Suport a l'XML Schema: Permet crear taules i tipus automàticament a partir de l'esquema XML.
- Modificacions parcials dels documents XML: Mitjançant XPath podem especificar elements i atributs concrets d'un document XML per tal de modificar-los sense necessitat d'extreure el document sencer.
- Cerques XPath: Es poden realitzar consultes amb XPath i després aplicant operadors SQL sobre els resultats obtinguts.
- Índexs XML: Amb XPath podem especificar parts del document per crear índexs que seran utilitzats en les cerques XPath.
- Operadors XML: Disponibilitat de nous operadors per realitzar consultes XML i generar XML de forma senzilla (per exemple XMLTABLE, XMLELEMENT, etc.).
- XSLT per a XMLType: Permet aplicar transformacions XSLT sobre documents XML mitjançant operadors SQL.
- Vistes XML: Permeten crear agregacions de diversos documents XML o taules, creant les representacions pròpies XML més eficients.

Les principals característiques del Repositori XML són les següents:

- Control d'accés: Proporciona mecanismes eficients que gestionen llistes de control d'accés sobre qualsevol objecte i que permeten definir els propis privilegis de seguretat.
- Classificació en carpetes: Gestió de carpetes per guardar els documents XML dins les estructures de la base de dades.
- Accés via WebDAV i FTP: És permet l'accés als documents a través dels protocols WebDAV i FTP.
- Cerques SQL al repositori: És permet realitzar consultes SQL sobre el repositori per cercar carpetes, fitxers en funció del seu propietari, de la seva data de creació, etc.
- Servlet d'accés: És proporciona un servlet per poder processar i manipular els documents XML via Java.

8.3 Tamino

Tamino és el servidor XML natiu desenvolupat per l'empresa Software AG, membre del consorci del W3C. Software AG va desenvolupar Tamino amb l'objectiu de dissenyar un servidor òptim per al tractament específic d'XML, sobretot en l'àmbit dels negocis electrònics. Tamino és molt més que un nou sistema de base de dades i un mitjà d'emmagatzematge per a XML ja que també ofereix accés a dades i a aplicacions existents, de manera que aquestes estiguin totalment integrades en les noves aplicacions per als negocis electrònics. Tamino comprèn un arquitectura completa i els seus principis bàsics són estàndards oberts basats en XML, TCP/IP i HTTP.

El servidor Tamino XML és el primer sistema que permet emmagatzemar la informació en format natiu XML. D'aquesta manera no és requereix cap capa addicional que s'ocupi de la conversió de les dades (com en el cas del sistemes gestors de base de dades relacionals) sinó que l'estructura dels documents és manté intacta.

Tamino és particularment adequat per a ser utilitzat conjuntament amb aplicacions de negocis electrònics. Diferents tipus d'informació com text, vídeo, so, etc. poden combinar-se, emmagatzemar-se i ser recuperats com a objectes XML. Tamino ofereix millor rendiment per a les estructures complexes que és troben a la xarxa.

Com més gran és la quantitat d'informació rebuda i emmagatzemada en format XML, més gran és l'avantatge que suposa adoptar la tecnologia XML proporcionada per Tamino. El fet d'emmagatzemar els documents XML en el seu format original i treballar amb un servidor XML natiu, que opera en els mateixos termes definits pels estàndards XML W3C, garanteix l'escalabilitat davant d'increments en el nombre de transaccions o de dades de negoci, menys cost operacional i menys cost en l'administració de les dades.

Les principals característiques que proporciona el servidor XML Tamino són:

- Emmagatzematge natiu de XML: els documents són emmagatzemats intactes, sense necessitat de fer cap transformació, proporcionant persistència en les dades.
- Emmagatzematge de qualsevol tipus de dades: El servidor Tamino accepta documents XML ben formats però també qualsevol tipus de dades encara que no estigui en format XML (documents de so, vídeo, PDF, etc.). No hi ha necessitat de crear estructures complicades per poder emmagatzemar aquests tipus de dades.
- Extensible: Afegir nous elements als documents XML ja emmagatzemats o afegir un nou element a l'esquema XML resulta molt simple.
- Integració de dades procedents de diverses fonts: Proporciona la visió d'un únic servidor en el cas de que les dades de negoci provinguin de fonts de dades distribuïdes.
- Eficient motor de cerca pel contingut en format XML: Tamino realitza les cerques seguint l'estàndard XPath i fent ús d'índexs que permeten alts rendiments.
- Capacitat per generar la sortida en diferents formats: proporciona independència de la plataforma de sortida, permetent extreure la informació emmagatzemada en diferents formats (XML, HTML, WML, PDF, etc.).
- Molta rapidesa: al ser un servidor natiu de XML, Tamino ofereix un rendiment molt més alt que l'ofert pels sistemes gestors de bases de dades relacionals.
- Extremadament robust: Tamino va ser dissenyat per executar aplicacions de missió crítica.
- Altament escalable: Tamino pot evolucionar i adaptar-se d'acord amb el nombre d'usuaris i el volum de dades.
- Segur: Tamino suporta conceptes flexibles de seguretat a diferents nivells.

L'arquitectura del servidor Tamino inclou els següents elements:

- X-Machine: és el nucli robust d'alt rendiment per a l'emmagatzematge d'objectes XML. Administra totes les estructures de dades en els diversos formats (XML, SQL, imatges, fitxers de so, vídeo, etc.).
- X-Node: permet l'accés a base de dades existents amb estructures de dades tradicionals. X-Node projecta aquestes dades sobre estructures XML, permetent a Tamino actuar com un servidor per a les bases de dades ja existents i basades en el model relacional.
- SQL Engine, permet l'emmagatzematge de dades SQL en Tamino. Aquestes dades poden ser part de document XML o utilitzats per aplicacions que només treballen amb dades SQL però que s'executen en el mateix servidor que Tamino.
- X-Manager, permet a l'administrador de Tamino projectar tots els objectes XML a estructures internes per tal d'administrar tot el sistema.
- Data Map, conté informació meta-data com els esquemes Tamino derivats dels DTDs o dels esquemes XML, fulles d'estil, esquemes, etc., que defineix les regles de com els objectes XML s'han de guardar i composar.

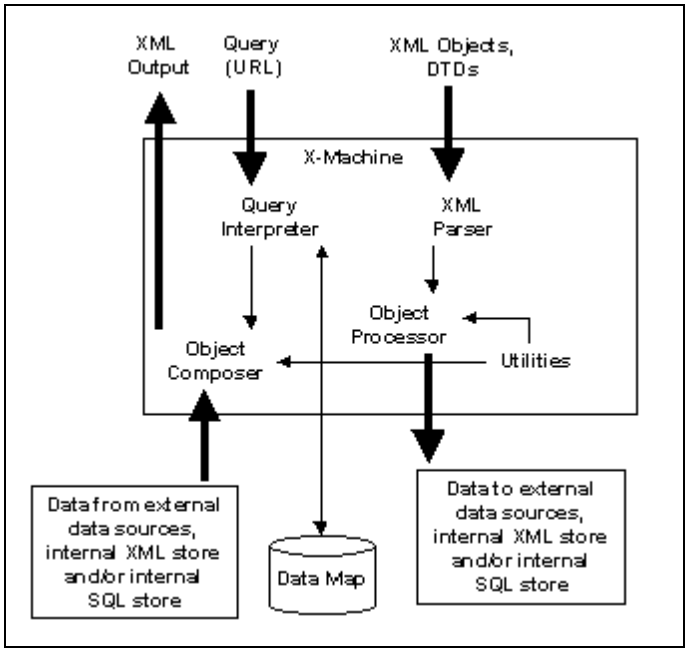


Figura 8.5. Arquitectura de Tamino.

9 Conclusions

L'estudi realitzat sobre l'XML i la família d'estàndards relacionats, per tal d'elaborar la present memòria, ens ha fet adonar de la forta evolució que ha experimentat en els darrers anys tant l'XML com totes les tecnologies que hi estan relacionades. Aquesta evolució no ha estat fruit de la casualitat sinó de l'esforç constant que estan realitzant els diversos grups de treball del W3C que, veient els avantatges que comporta l'ús de l'XML, treballen per tal de garantir l'estandardització i difusió de l'XML, així com de la resta d'estàndards que donen lloc a la potència, robustesa i versatilitat de l'XML.

Cal remarcar que davant l'abundant i complexa informació existent, en la present memòria es fa una primera introducció a XML i es presenten les principals característiques dels diversos estàndards i tecnologies XML. Ens estem referint als estàndards XSL, XSLT, XSL-FO, XLink, XPointer, XPath, etc., i hem de tenir present que, per tal de conèixer la potència real d'aquests estàndards, cal aprofundir molt més en el seu estudi. A més, en molts casos encara no hi ha una aplicació pràctica del seu ús (o bé, actualment no està molt generalitzada), ja que es tracta de tecnologies molt noves i poc experimentades que, sens dubte, demostraran el seu potencial en els propers anys.

Entre les característiques i els diversos avantatges que comporta l'ús de l'XML, en destaquem el fet de ser un llenguatge totalment extensible, la seva capacitat innata d'independitzar les dades de la seva forma de representació i la seva naturalesa intrínseca que el converteix en un llenguatge òptim per a l'estructuració de informació. Justament, aquesta capacitat per estructurar les dades, permet per una banda que la informació pugui ser estructurada jeràrquicament i sigui fàcilment accessible gràcies a la seva representació en forma d'arbre i per l'altra, relaciona l'XML amb el camp de les bases de dades. En aquest sentit, hem vist com el llenguatge de consulta XQuery, juga un paper molt important alhora de consultar, recuperar i interpretar les dades estructurades mitjançant els documents XML.

Ja centrant-nos més directament en el camp de les bases de dades, hem vist com tracten l'XML els sistemes gestors de bases de dades relacionals, en concret Oracle, i com ho fan els sistemes de bases de dades basats totalment en tecnologia XML, en concret Tamino.

Els sistemes de base de dades relacionals, com Oracle, estan evolucionant per tal d'adaptar-se a les noves necessitats i fer-se un lloc en el mercat, per tal de convertir-se en una bona opció que faciliti l'emmagatzematge i l'intercanvi de dades en format XML. Aquests sistemes, han desenvolupat interfícies, eines i utilitats que permeten emmagatzemar XML dins les seves estructures internes basades en taules relacionals. Tot i això, el model basat en les taules relacionals no satisfà la naturalesa jeràrquica i interconnectada dels objectes XML i això, provoca que els sistemes de bases de dades relacionals, necessitin desglossar els documents XML per tal de poder emmagatzemar-los en diverses taules relacionades entre sí. A més, el sistemes de base de dades més avançats i orientats a objectes, no poden manipular dades d'acord amb l'estructura dinàmica que permet la naturalesa extensible de l'XML. Per tal de solucionar tots aquests problemes, Oracle ha evolucionat encara més i ha presentat l'opció Oracle XML DB, que permet la integració total a nivell intern del model XML amb el model relacional, per tal d'oferir als usuaris els avantatges de les dues tecnologies.

El sistema natiu XML Tamino, a diferència dels sistemes relacionals, pot emmagatzemar la informació directament en format natiu XML, fent que no sigui necessari cap procés intermedi de transformació d'un model a un altre. A més, la naturalesa de Tamino, el converteix en un sistema robust, altament escalable i amb un rendiment més òptim que els sistemes de bases de dades relacionals. També, cal destacar la seva facilitat d'integració amb sistemes ja existents.

Degut a l'abast del present treball (XML, els estàndards i tecnologies XML i XML en el camp de les bases de dades), només hem presentat breument les principals característiques de les solucions proposades per Oracle i Tamino, i no hem realitzat, tot i que hagués resultat molt interessant, un cas pràctic comparatiu de les dues tecnologies. Així doncs, una possible línia a seguir en treballs futurs, seria aprofundir en l'estudi de les dues solucions, per tal de veure en més detall quins són els seus avantatges i inconvenients i també, realitzar un cas pràctic que permetés fer un estudi comparatiu i un anàlisis de rendiment de les dues tecnologies.

GLOSSARI:

API: Interfície de programació per al desenvolupament d'aplicacions.

atribut: Component de l'element que actua com a modificador d'aquest, afegint informació addicional a l'element al que està associat.

base de dades XML nativa: Base de dades en la que les seves estructures internes de dades es projecten directament al format jeràrquic de XML.

data model XQuery: Model de dades formal sobre el que es basa l'XQuery, consistent en representar el document XML com un arbre de nodes.

document XML: Document format únicament per text pla i que conté informació estructurada i delimitada mitjançant marques.

document XML ben format: Document XML que segueix les regles de marcatge correctament.

document invàlid: Document XML que inclou o indica un DTD i no el satisfà.

document vàlid: Document XML que inclou o indica un DTD i el satisfà.

DOM: *Document Object Model*, interfície de programació orientada a objectes que representa els documents XML mitjançant un conjunt d'objectes en forma d'arbre.

DTD: *Document Type Definitions*, document que defineix la gramàtica dels documents XML.

element: Unitat bàsica amb capacitat per representar la lògica i la semàntica d'un document XML.

element arrel: Element del document XML que penja directament del node arrel.

element document: element arrel.

entitat: Unitat física d'emmagatzematge del document XML.

entitat analitzable: Entitat que conté informació que el processador XML entén i és capaç de processar.

entitat externa: Entitat que té total o parcialment definit el seu contingut fóra del document XML.

entitat interna: Entitat que té definit el seu contingut totalment dins del document XML.

entitat no analitzable: Entitat que conté informació que no pot ser tractada pel processador XML.

entitat predefinida: Entitat que escapa un caràcter especial.

esquema XML: *XML Schema*, document en sintaxi XML que defineix l'estructura, la gramàtica, el contingut i la semàntica dels document XML de forma més estricta i robusta que els DTD.

expressió FLWOR: Expressions de gran potència utilitzades per XQuery per tal de fer iteracions i extreure informació dels documents XML.

HTML: *Hypertext Markup Language*, llenguatge de marques per a la creació de pàgines web.

interfície: Plataforma de programació que proporciona un conjunt de classes i mètodes per facilitar les tasques de desenvolupament.

node arrel: Node identificat per '/' i que inicia l'arbre que forma el document.

node atribut: Qualsevol atribut d'un document XML es converteix en un node atribut de l'arbre.

node element: Qualsevol element d'un document XML es converteix en un node element de l'arbre.

Oracle XDK: *XML Developers Kit*, conjunt d'eines proporcionades per Oracle8i i Oracle9i per al processament d'XML tant dins com fóra de la base de dades.

Oracle XML DB: Tecnologia d'alt rendiment per l'emmagatzematge d'XML disponible amb el servidor Oracle9i Rel 2, que es basa en el model de dades XML definit per W3C.

RDBMS: Sistema manipulador de bases de dades relacionals.

SAX: *Simple API for XML*, interfície de programació simple basada en events i fàcil d'utilitzar per analitzar els documents XML.

SGBD: Sistema gestor de bases de dades.

SGML: *Standard Generalized Markup Language*, llenguatge de marques que constitueixen un estàndard extensible més potent que XML però més difícil d'implementar.

SQL: Llenguatge de consulta de les base de dades relacionals.

Tamino: Servidor XML natiu desenvolupat per l'empresa Software AG que permet emmagatzemar la informació seguint el propi format XML.

W3C: *World Wide Web Consortium*, organisme creat amb l'objectiu de treure el màxim rendiment de la web mitjançant el desenvolupament de protocols (especificacions, guies d'estil, software, etc.).

XLink: *XML Linking Language*, llenguatge que permet introduir enllaços en els documents XML per tal de relacionar documents entre ells.

XML: *Extensible Markup Language*, llenguatge extensible de marques que defineix el format estàndard per a l'estructuració de dades i informació.

XPath: Llenguatge que permet adreçar parts d'un document XML, seguint l'estructura lògica (arbre de nodes) que descriu el contingut del document.

XPointer: Llenguatge que permet identificar i fer referències a parts que es troben dins d'un document XML, mitjançant apuntadors o especificacions de camins.

XSL: *Extensible Stylesheet Language*, llenguatge que aplica format als documents XML indicant com s'han de visualitzar i donant certa capacitat de transformació del contingut dels documents.

XSLT: *Extensible Stylesheet Language Tranformation*, llenguatge que permet transformar un document XML en un altre document XML.

XSLT-FO: *XML Stylesheet Language Formatting Objects*, llenguatge de formateig que descriu de forma precisa com s'ha de visualitzar el contingut dels documents XML

XQuery: Llenguatge de consulta que basant-se en l'estructura dels documents XML, proporciona els mecanismes necessaris per poder interrogar, recuperar i interpretar la informació.

BIBLIOGRAFIA

Elliotte Rusty Harold & W. Scott Means, (2001) "*XML IN A NUTSHELL: A Desktop Quick Reference*". O'REILLY.

Abraham Gutiérrez, Raúl Martínez (2001) "*XML a través de ejemplos*". Editorial Ra-Ma.

Michael Champion , Senior R&D Advisor, Software AG "*XML Nativo vs. Habilitado para XML: La diferencia Hace la diferencia*"
http://www.xml.com.ve/articulos_3.htm

XML & Co. "*Uniendo la Sociedad de la Información*"
http://www.xml.com.ve/articulos_2.htm

XML Path Language (XPath) Version 1.0 , W3C Recommendation 16 November 1999
http://www.w3.org/TR/xpath

XML Linking Language (XLink) Version 1.0, W3C Recommendation 27 June 2001
http://www.w3.org/TR/2001/REC-xlink-20010627/

XSL Transformations (XSLT) Version 1.0 W3C Recommendation 16 November 1999
http://www.w3.org/TR/1999/REC-xslt-19991116

Extensible Stylesheet Language (XSL) Version 1.0, W3C Recommendation 15 October 2001
http://www.w3.org/TR/2001/REC-xsl-20011015/

XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001
http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

XML Schema Part 1: Structures , W3C Recommendation 2 May 2001
http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

XML Schema Part 2: Datatypes , W3C Recommendation 02 May 2001
http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

XQuery 1.0 and XPath 2.0 Data Model, W3C Working Draft 12 November 2003
http://www.w3.org/TR/2003/WD-xpath-datamodel-20031112/

XQuery 1.0 and XPath 2.0 Formal Semantics, W3C Working Draft 12 November 2003
http://www.w3.org/TR/2003/WD-xquery-semantics-20031112/

XQuery 1.0: An XML Query Language, W3C Working Draft 12 November 2003
http://www.w3.org/TR/2003/WD-xquery-20031112/

XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Working Draft 12 November 2003
http://www.w3.org/TR/2003/WD-xpath-functions-20031112/

Document Object Model (DOM) Level 2 Core Specification Version 1.0, W3C Recommendation 13 November, 2000
http://www.w3.org/TR/DOM-Level-2-Core/

Oracle XML: http://otn.oracle.com/tech/xml/xmldb/index.html

Tamino XML SERVER: http://www.softwareag.com/tamino/

SAX: http://www.saxproject.org/