

# Àmbits d'aplicació de l'XML

Xavier Sala

Llenguatges de marques i sistemes de gestió  
d'informació



# Índex

<b>Introducció</b>	<b>5</b>
<b>Resultats d'aprenentatge</b>	<b>7</b>
<b>1 Sindicació de continguts</b>	<b>9</b>
1.1 "Feeds" o canals . . . . .	10
1.2 RSS . . . . .	12
1.2.1 RSS 3.0 . . . . .	12
1.2.2 Llenguatge RSS 2.0 . . . . .	13
1.3 Atom . . . . .	20
1.3.1 Llenguatge Atom . . . . .	20
1.4 Validació . . . . .	24
1.5 Agregadors / lectors . . . . .	27
1.5.1 Lectors via web . . . . .	28
1.5.2 Programari d'escriptori . . . . .	29
1.6 Problemes . . . . .	29
<b>2 Conversió i adaptació de documents XML</b>	<b>31</b>
2.1 Ús de CSS . . . . .	31
2.2 Transformació de documents . . . . .	33
2.2.1 XSL-FO . . . . .	33
2.3 Processadors XPath o XSLT . . . . .	34
2.3.1 Biblioteques . . . . .	34
2.4 XPath . . . . .	37
2.4.1 Vista d'arbre . . . . .	38
2.4.2 Programari per avaluar XPath . . . . .	39
2.4.3 Navegació . . . . .	40
2.4.4 Seqüències . . . . .	48
2.4.5 Funcions . . . . .	50
2.5 XSLT . . . . .	51
2.5.1 Programes . . . . .	53
2.5.2 El procés de transformació . . . . .	53
2.5.3 Exemple . . . . .	62
<b>3 Bases de dades natives XML. Llenguatge de consultes (XQuery)</b>	<b>67</b>
3.1 Bases de dades relacionals . . . . .	68
3.1.1 Convertir les dades XML en relacionals . . . . .	68
3.1.2 Sistemes relacionals amb extensions XML . . . . .	70
3.2 XQuery . . . . .	73
3.2.1 Processadors XQuery . . . . .	74
3.2.2 Consultes XQuery . . . . .	77
3.2.3 Comentaris . . . . .	78
3.2.4 Càrrega de documents . . . . .	78
3.2.5 Variables . . . . .	79

3.2.6	Expressions avaluables . . . . .	80
3.2.7	Expressions FLWOR . . . . .	81
3.2.8	Alternatives . . . . .	87
3.2.9	El pròleg XQuery . . . . .	87
3.2.10	Actualitzacions de dades . . . . .	89
3.3	Bases de dades natives XML . . . . .	91
3.3.1	Característiques més importants . . . . .	92
3.3.2	Exemples de bases de dades natives XML . . . . .	96
3.3.3	eXist-db . . . . .	96

## Introducció

En aquesta unitat veureu diferents aspectes relacionats amb XML: les tecnologies de sindicació de continguts, la transformació de documents i l'emmagatzematge de dades XML.

La sindicació de continguts ha estat un dels darrers èxits a Internet, ja que permet alliberar de feina els usuaris. En l'apartat “Sindicació de continguts” s'exploren les característiques de dos dels llenguatges de sindicació més usats actualment: RSS i Atom.

Atesa la llibertat que dóna XML, sovint l'intercanvi de dades entre programes no serà immediat, sinó que requerirà que abans s'hi faci algun tipus de transformació. En l'apartat “Conversió i adaptació de documents XML” veureu el llenguatge XSLT per transformar documents XML.

En l'apartat “Bases de dades natives XML. Llenguatge de consultes (XQuery)” s'exploraran les diferents possibilitats que hi ha per emmagatzemar documents XML de manera que pugui ser recuperats ràpidament.



## Resultats d'aprenentatge

En acabar la unitat, l'alumne:

**1. Genera canals de continguts analitzant i utilitzant tecnologies de sindicació.**

- Identifica els avantatges que aporta la sindicació de continguts en la gestió i transmissió de la informació.
- Defineix els àmbits d'aplicació de la sindicació de continguts.
- Analitza les tecnologies en què es basa la sindicació de continguts.
- Identifica l'estructura i la sintaxi d'un canal de continguts.
- Crea i valida canals de continguts.
- Comprova la funcionalitat i l'accés als canals de continguts.
- Utilitza eines específiques com agregadors i directoris de canals.

**2. Fa conversions sobre documents XML utilitzant tècniques i eines de processament.**

- Identifica la necessitat de la conversió de documents XML.
- Estableix àmbits d'aplicació de la conversió de documents XML.
- Analitza les tecnologies implicades i la manera de funcionament.
- Descriu la sintaxi específica utilitzada en la conversió i adaptació de documents XML.
- Crea especificacions de conversió.
- Identifica i caracteritza eines específiques relacionades amb la conversió de documents XML.
- Fa conversions amb diferents formats de sortida.
- Documenta i depura les especificacions de conversió

**3. Gestiona informació en format XML analitzant i utilitzant tecnologies d'emmagatzematge i llenguatges de consulta.**

- Identifica els principals mètodes d'emmagatzematge de la informació utilitzada en documents XML.
- Identifica els inconvenients d'emmagatzemar informació en format XML.
- Estableix tecnologies eficients d'emmagatzematge d'informació en funció de les seves característiques.
- Utilitza sistemes gestors de bases de dades relacionals en l'emmagatzematge d'informació en format XML.

- Utilitza tècniques específiques per crear documents XML a partir d'informació emmagatzemada en bases de dades relacionals.
- Identifica les característiques dels sistemes gestors de bases de dades natives XML.
- Instal·la i analitza sistemes gestors de bases de dades natives XML.
- Utilitza tècniques per gestionar la informació emmagatzemada en bases de dades natives XML.
- Identifica llenguatges i eines per al tractament i emmagatzematge d'informació i la inclusió en documents XML.

## 1. Sindicació de continguts

Quan es parla de sindicació de continguts s'està fent referència a l'intercanvi actualitzat d'informació entre pàgines web.

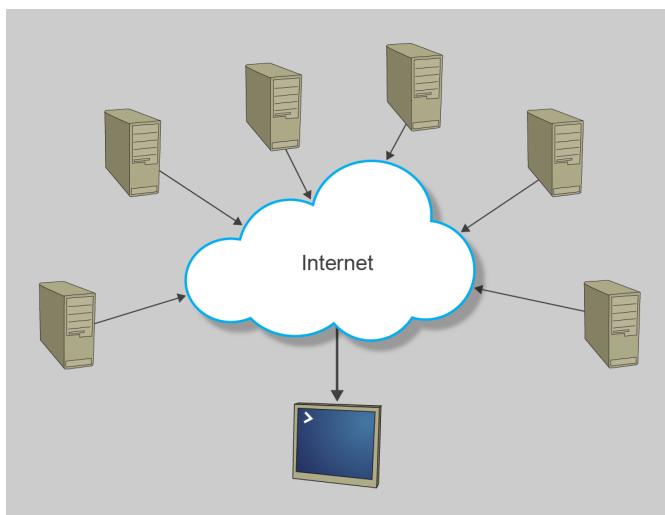
La primera manera de sindicació que hi va haver va ser la integració de notícies d'una pàgina dins d'una altra per mitjà d'algún tipus de programa que obtenia la informació cercant per dins del contingut HTML. Però aquests programes tenien el problema que sovint quedaven obsolets, ja que qualsevol canvi en la pàgina original podia fer que deixessin de funcionar correctament.

A pesar del que pugui semblar per l'abundància de “decoració” que hi ha en les pàgines web, el més important és el contingut. La informació continguda en els articles, els fitxers, etc., és el que fa que els visitants hi tornin o no.

Amb el sistema tradicional de navegació per Internet, per a un usuari era molt important obtenir els enllaços dels llocs web que li interessaven i emmagatzemar-los d'alguna manera per poder-hi tornar ràpidament. Si es volien seguir els canvis en les pàgines web l'única manera que hi havia era anar visitant de tant en tant la pàgina per comprovar si hi havia novetats.

L'aparició del que es va conèixer com a **Web 2.0** va complicar les coses. El Web es va emplenar d'una gran quantitat de blogs i pàgines que publicaven informació, i visitar-les totes per veure si hi havia canvis va passar a requerir molt de temps, tot plegat per visitar pàgines que potser no havien canviat. S'havia d'optimitzar d'alguna manera aquesta tasca.

**FIGURA 1.1.** Amb RSS és la informació la que ve a nosaltres



L'aparició dels sistemes estàndard de sindicació va fer possible obtenir la informació de les actualitzacions d'un lloc web d'una manera estable per mitjà d'una

adreça. La sindicació de continguts va canviar la manera com es recupera la informació. Ja no calia anar a buscar la informació: era la informació la que acudia a l'usuari (figura 1.1).

Fent servir la sindicació **ja no cal que l'usuari visiti les pàgines** que li interessen per a veure si hi ha canvis perquè si n'hi ha ja els rebrà. Això comporta un estalvi de temps, ja que no caldrà visitar pàgines per descobrir que no hi ha canvis.

Amb la sindicació el disseny de la pàgina original no afecta els programes que hi busquen informació, ja que la sindicació de continguts està basada en XML i no prioritza la informació d'estil sinó el contingut.

Aquesta està pensada perquè hi pugui interactuar tant els humans com els programes, i això fa que es pugui dissenyar fàcilment aplicacions que obtinguin la informació de manera automàtica sense que calgui cap tipus d'intervenció humana. La participació de l'humà es limitarà a dir al programa quins llocs ha de vigilar.

A més, **un cop l'usuari rep la informació en pot fer el que vulgui**: filtrar-ne els continguts, classificar-la per temes... Per tant, tindrà el control de quina és la informació que vol veure i quina no.

Un altre dels avantatges que aporta la sindicació és inherent a XML. A diferència del que passa amb HTML, és fàcil interpretar el contingut de la informació que es rep i, per tant, també **serà fàcil poder reutilitzar-ne el contingut** per fer-hi altres tasques.

A pesar que la sindicació es veu sovint com un sistema enfocat a detectar novetats en el Web, també s'està fent servir per a **mantenir actualitzacions** en altres camps. Per exemple, alguns programes d'ordinador fan servir RSS per saber si n'han sortit actualitzacions i d'aquesta manera mantenen els programes actualitzats.

## 1.1 "Feeds" o canals

Un canal és un arxiu que conté una versió específica de la informació que s'ha publicat en un lloc web.

En aquest arxiu es troba tota la informació sobre el lloc web i enllaços als seus continguts. El gran avantatge és que en estar basat en XML es pot aconseguir transmetre la informació de manera automatitzada i els receptors la podran interpretar fàcilment.

Per poder obtenir la informació del canal normalment caldrà localitzar el fitxer. Generalment aquests canals estan associats a una pàgina web i s'hi pot accedir per mitjà d'un enllaç.

Els enllaços soLEN estar clarament especificats amb el text *RSS*, *XML* o bé fENT servir un GRUP d'ICONES (figura 1.2).

**FIGURA 1.2.** ICONES TÍPIQUES PER MARCAR CONTINGUT RSS



És bastant habitual que les pàGINES que tenEN contingut de sindicació tinguIN en algUN lLOC del seu web UN enllaç o LA icona d'RSS (figura 1.3).

**FIGURA 1.3.** ENLLAÇ AL CANAL DE CONTINGUT DE L'XTEC



Els fitxERS dels canals normalMENT es passARAN a proGRAMES que serAN els que s'encarregARAN de recollir periòDICAMENT les actualitzACIONS de la informació del canal. En la terminologIa de sindicació això se sol anomenar **subscripció**.

L'ús de canals aportarÀ diferENTS avantatges als usuariS:

- Com que funciona per subscripció, aquests només rebran les noticies d'interès seu.
- El programa tendirÀ a donar informació més acurada a les seves preferencies i gustos del que ho fan els cercadors generalistes com Google, ja que els canals contenen el resultat d'una cerca.
- La informació es pot classificar i ordenar segons els gustos de l'usuari i consumir-se segons els criteris que es vulgui.
- En qualsevol moment es pot deixar de seguir un canal sense haver de demanar cap tipus de permís.

Al llarg dels anys s'han desenvolupat diverses tecnologies per crear canals com CDF (*channel definition format*, desenvolupat per Microsoft), PointCast o Apple MCF (*meta content framework*), però els llenguatges de creació de canals que s'han fet més populars i que s'han convertit en la manera estàndard de sindicació han estat sobretot **RSS** i en menys mesura **Atom**.

## 1.2 RSS

RSS són les sigles que es fan servir per anomenar diferents estàndards molt populars per a la sindicació de continguts que s'han convertit en una manera estàndard d'intercanviar informació al Web.

RSS no ha estat desenvolupat per cap organisme d'estàndards ni és una recomanació del W3C, però és un format obert i lliure sota llicència Creative Commons:

A pesar que generalment es parla d'RSS de manera genèrica, al parlar d'RSS sempre s'hauria d'especificar quina és la versió d'RSS de la qual es parla, ja que les versions existents són bastant diferents.

Els vocabularis RSS són probablement uns dels vocabularis XML que han tingut més èxit de tots els que es fan servir a Internet. Han aconseguit aquesta popularitat entre d'altres coses perquè són senzills i oberts.

Des de l'aparició d'RSS 0.9 el 1999 han anat apareixent diferents versions d'RSS que es poden resumir en dos grans grups:

- RSS 0.92/ RSS 2.0
- RSS 1.0

Aquesta divisió va ser deguda a problemes d'enteniment sobre quin havia de ser l'objectiu d'RSS i va fer que es creessin dues versions. A pesar de la semblança del nom de totes dues, RSS 1.0 és un llenguatge basat en RDF (*resource description framework*), cosa que el fa bastant diferent d'RSS 0.9.

La versió que es fa servir més és la 2.0, que s'assembla més a la versió 0.92 però incorpora alguns aspectes de la versió 1.0. Probablement la versió 2.0 és una de les més senzilles d'utilitzar; a pesar d'això, no obstant això, és una de les més malinterpretades. Algunes de les definicions dels elements no sempre es fan servir de la mateixa manera per a tots els canals.

### 1.2.1 RSS 3.0

RSS està aturat en les versions 2.0, en concret en la versió 2.0.1, perquè es considera que ja té tot el suficient per funcionar, i perquè ha costat que fos molt adoptat.

Hi ha una proposta per fer un “RSS 3.0” que, per no trencar amb la tradició, també seria incompatible amb les altres versions:

- Abandonant XML per fer servir un llenguatge de marques més lleuger, ja

que es considera que XML és massa complex per fer tasques senzilles.

- Abandonar l'ús d'espais de noms, ja que compliquen les coses.
- No permetre fer servir HTML dins d'RSS.

En aquesta proposta es defineix que els elements es posin en línies diferents i que si el seu contingut no hi cap es comenci a la línia següent deixant un espai en blanc. A més, hi ha d'haver una línia en blanc per separar les notícies.

```

1 title: Notícies
2 description: Canal de notícies
3 link: http://noticiesA.cat/rss30
4 creator: esport@noticiesA.cat
5 language: ca-ES
6
7 title: Primera notícia
8 created: 2011-11-02
9 guid: 00795648-C1E0-11D6-9AA6-003065F376B6
10 description: Notícia número 1
11
12 title: Segona notícia
13 created: 2011-06-13
14 guid: 0894CB2F-C1E0-11D6-9649-003065F376B6
15 description: Notícia número 2

```

Molta gent no s'ha pres seriosament aquesta prescripció, però qui sap si amb el temps s'aconseguirà que s'appliqui majoritàriament.

---

Podeu trobar més informació sobre aquesta versió a l'enllaç <http://www.aaronsw.com/2002/rss30>

---

## 1.2.2 Llenguatge RSS 2.0

Entre tots els canals disponibles, RSS 2.0 és el més usat amb molta diferència, i per tant, la gran majoria dels programes lectors d'RSS el suporten.

---

Algunes fonts diuen que el 80% dels canals RSS són 2.0, tot i que no hi ha estadístiques oficials

---

Una de les característiques que defineixen RSS és que fa honor al seu nom oficial, *really simple syndication* (sindicació realment senzilla), i **és un sistema senzill**. Es tracta d'un sistema que no té cap estructura complexa, en què les etiquetes descriuen el contingut que hi ha en l'element, en què pràcticament no es fan servir els atributs per a res i els espais de noms només es fan servir en les extensions, si n'hi ha.

### L'arrel

RSS és un llenguatge XML, de manera que n'ha de complir les normes i, per tant, només té un sol element arrel, <rss>. La funció d'aquest element és simplement informar a qui llegeixi el document que el que està llegint és un canal RSS.

L'element arrel té un dels pocs atributs obligatoris de l'especificació, *version*, que és necessari per indicar la versió que s'està fent servir. Aquest atribut serveix perquè els programes sàpiguen quina versió d'RSS es fa servir en el document.

```

1 <rss version="2.0">
2 ...
3 </rss>
```

A pesar que s'hi pot especificar l'espai de noms, RSS no el té en compte, de manera que mai no caldrà posar cap àlies davant de les seves etiquetes.

Però si el canal fa servir alguna extensió sí que caldrà que es defineixi l'espai de noms de les extensions que es fan servir.

```

1 <rss version="2.0"
2   xmlns:dc="http://purl.org/dc/elements/1.1/"
3   xmlns:content="http://purl.org/rss/1.0/modules/content/"
4   ...
5 </rss>
```

### L'element <channel>

L'arrel només serveix per indicar que el document és de tipus RSS, i el contingut del canal estarà dins de l'únic fill de <rss>, que s'anomena <channel>. L'element <channel> serà el que contindrà totes les etiquetes que aporten informació sobre el canal, i sobretot les que tindran les novetats del lloc.

Només hi pot haver una sola etiqueta <channel> en tot el document RSS.

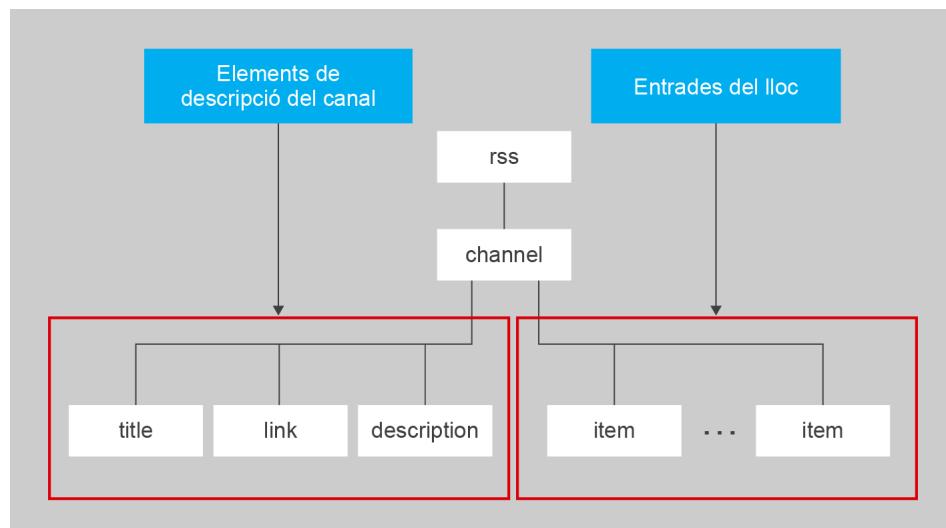
```

1 <rss version="2.0">
2   <channel>
3     ...
4   </channel>
5 </rss>
```

Es pot dividir el contingut d'un canal RSS en dos grans grups (figura 1.4):

1. Un grup d'etiquetes destinades a **descriure el canal**.
2. Els **elements** <item>, que són els que contindran el contingut del canal.

**FIGURA 1.4.** Divisió del contingut d'un document RSS



Per fer-ho tot més senzill algunes de les etiquetes es repeteixen en els dos grups. Per exemple les etiquetes més importants de <channel>, que són <title>, <link> i <description>, són també les més importants dels elements <item>.

## Etiquetes per a descriure el canal

Els primers elements que es troben dins de l'element <channel> estan destinats a donar informació sobre el canal RSS. Aquestes etiquetes no són de contingut ni serà habitual que es produueixin canvis en els seus valors.

Els elements més importants d'aquesta part són els elements <title>, <link> i <description> (taula 1.1) que són obligatoris en tots els canals RSS.

**TAULA 1.1.** Elements obligatoris de l'element 'channel'

Element	Ús
title	El nom del canal. És la manera com es coneix el servei.
link	Normalment s'inclou un enllaç a l'adreça on hi ha el contingut HTML que es correspon amb el canal.
description	Conté una descripció curta del contingut del canal.

L'especificació a l'hora de definir els valors que s'han de posar en cada una de les etiquetes és poc concreta i alguns dels seus elements han estat subjectes a diferents interpretacions per part dels creadors de contingut.

RSS 1.0, que està basat en l'estàndard del W3C RDF, va néixer amb l'objectiu d'intentar que els elements d'RSS deixessin de ser tan malinterpretats. Però RSS 2.0 no parteix d'RSS 1.0 sinó de les versions 0.9.

Aquests són els únics elements obligatoris i, per tant, el document següent seria un document RSS vàlid:

```

1  <rss version="2.0">
2    <channel>
3      <title>Canal de llenguatges de marques</title>
4      <link>http://ioc.xtec.cat/rss/Marques.html</link>
5      <description>Canal per entrar les notes del mòdul 4 d'ASIX</description>
6    </channel>
7  </rss>
```

A part dels elements obligatoris en RSS també n'hi ha uns quants que són voluntaris i que serveixen per donar informació extra sobre el canal. Es poden veure aquests elements a la taula 1.2.

**TAULA 1.2.** Altres elements possibles de l'element 'channel'

Element	Ús
language	Especifica l'idioma en que està escrit el canal.
copyright	Informació sobre el copyright del contingut.
managingEditor	Correu electrònic del responsable del contingut.
webMaster	Correu electrònic del responsable tècnic.
pubDate	Darrera data de publicació en el canal.
category	Categoría del contingut.
lastBuildDate	Darrera data de modificació del canal.
generator	Programa fet servir per generar el contingut.
docs	Descriu el format específic.
ttl	Temps que els clients han d'esperar per tornar a demanar.

**TAULA 1.2** (continuació)

Element	Ús
image	Icona que representa el canal.
rating	Fa servir una classificació americana sobre el contingut.
cloud	Grup de gent a la qual s'informa dels canvis.
textInput	És una etiqueta antiga que ja no es fa servir.
skipHours	En quines hores no es poden demanar actualitzacions.
skipDays	Quins dies no es poden demanar actualitzacions.

## Contingut del canal

El contingut visible d'un canal RSS anirà en els elements <item>, dels quals n'hi pot haver la quantitat que es vulgui (fins i tot cap).

Normalment cada vegada que es produeix una novetat en el lloc associat al contingut d'un canal es crea un nou element item que s'afegeix al document.

A pesar que no sembla que tingui sentit crear ítems sense contingut, això és estrictament possible perquè item no té cap element obligatori però sempre hi ha d'haver un <description> o un <title>.

Per tant, el següent seria un document RSS correcte. Tenim dos ítems, un amb títol sense contingut i un amb contingut sense títol.

```

1  <rss version="2.0">
2    <channel>
3      <title>RSS</title>
4      <link>http://ioc.xtec.cat/rss/RSS</link>
5      <description>Provant</description>
6      <item>
7        <title>Títol sense contingut</title>
8      </item>
9      <item>
10        <description>Contingut sense títol</description>
11      </item>
12    </channel>
13 </rss>
```

El més normal serà que hi hagi els elements <title>, <link>, i <description> però a més també hi ha altres atributs que es poden veure a la taula 1.3 com <pubDate> o <guid>.

**TAULA 1.3.** Subelements d'ítem

Element	Ús
title	Títol del nou contingut.
link	Enllaç al contingut en el lloc de referència.
description	Contingut.
guid	Una cadena que identifica totalment l'ítem per evitar entrades duplicades en els programes. S'hi sol posar l'URL del lloc.

**TAULA 1.3** (continuació)

Element	Ús
pubDate	Data i hora en què s'ha publicat.
author	Correu electrònic de l'autor.
category	Nom de la categoria que descriu el contingut.
comments	Adreça de la pàgina on es poden entrar comentaris.
enclosure	Identifica un fitxer extern associat. Normalment música o vídeo.
source	Una referència al canal. És rar que hi sigui.

## Contingut HTML

Al principi tothom feia servir text pla en els continguts del canal però aviat es va començar a pensar a incloure HTML, sobretot en aquelles etiquetes que els usuaris poden veure.

Estrictament parlant no es pot fer servir HTML dins d'RSS, perquè com que està basat en XML ha d'estar ben format, mentre que l'HTML no té l'obligació d'estar ben format i, per tant, en afegir HTML a un RSS hi ha la possibilitat que el resultat sigui mal format.

En l'exemple següent, dins de l'element `description` es volia afegir una imatge i s'ha fet servir l'etiqueta `<img>` d'HTML per fer-ho. El problema és que aquesta etiqueta converteix el document en incorrecte, ja que aquest deixa d'estar ben format (invalida la regla que diu que **s'han de tancar totes les etiquetes que s'obren**).

```

1 <description>
2 Quina imatge més bonica! 
3 </description>

```

Per tant, per afegir HTML a un document RSS s'ha de fer com en XML. Si es vol afegir contingut que no ha de ser processat s'ha de fer per mitjà d'una secció `<! [CDATA[ . . . ]]>`. Per tant, l'exemple anterior es podria representar de la manera següent:

```

1 <description>
2 <! [CDATA[
3   <div>
4     Quina imatge més bonica! 
5   </div>
6   ]]>
7 </description>

```

Per evitar problemes sovint tots els canals fan servir aquest sistema per a incloure HTML dins dels elements RSS, tant si el contingut està ben format com si no.

## Representació de dates

RSS fa servir el sistema de representació de dates que hi ha en un document relativament obsolet, l'especificació RFC 822 (“ARPA Internet text messages”)

El format de les dates ha de seguir la forma següent:

<sup>1</sup> Dia de la setmana, Dia Mes Any Hora:Minut:Segon Zona\_Horaria

On:

- El **dia de la setmana** és opcional i només pot ser només una abreviació dels noms en anglès: “Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”, “Sun”.
- El **mes** també es defineix en abreviacions (“Jan”, “Feb”, “Mar”, “Apr”, “May”, “Jun”, “Jul”, “Aug”, “Sep”, “Oct”, “Nov”, “Dec”).
- L'**any** sempre ha de tenir quatre dígits.
- La zona horària es pot expressar en diferències numèriques o per mitjà de constants de temps americanes. El més corrent sol ser que sigui expressada en GMT (*universal time*).

Per tant, seran dates correctes:

- *Sat, 13 Aug 2011 10:43:37 GMT*
- *13 Aug 2011 10:43:37 +0100*

## Extensions

La popularitat d’RSS ha fet que moltes grans companyies l’hagin fet servir per a les seves tasques. Però a vegades no solament l’han fet servir sinó que hi han afegit funcions segons les seves necessitats.

### Amazon

La llibreria en línia Amazon permet accedir al seu catàleg de productes per mitjà d’RSS, de manera que qualsevol pot estar al dia de què és el que s'està venent més, quines són les novetats, etc., simplement subscrivint-se a un dels RSS.

Per exemple, aquest és el canal de llibres més venuts de programació web:

[http://www.amazon.com/gp/rss/bestsellers/books/377888011/ref=zg\\_bs\\_377888011\\_rsslink](http://www.amazon.com/gp/rss/bestsellers/books/377888011/ref=zg_bs_377888011_rsslink)

The screenshot shows the Amazon.com Help section for RSS Feeds. It includes a search bar, a sidebar with links to topics like 'RSS Feeds' and 'RSS Options', and a main content area with sections for 'RSS Feeds for Best Sellers, Hot New Releases, Most Gifted, Most Wished For, and Movers & Shakers', 'Advanced RSS Options', and 'Self-Service' features.

Com que RSS està basat en XML es pot barrejar amb altres llenguatges XML fent servir els espais de noms. Això ha permès que qualsevol pugui incrementar la funcionalitat d'RSS sense canviar-ne el nucli de funcionament.

Per tant, hi ha tota una sèrie d'extensions que es poden usar dins d'RSS per obtenir funcions no previstes en l'especificació. Entre les més populars destaquen:

- **Dublin core metadata initiative:** es tracta d'una extensió que permet introduir els elements definits en el *Dublin core* en RSS. Aquests elements serveixen com a metadades per a descriure els recursos d'una xarxa

```

1  <rss version="2.0"
2   xmlns:dc="http://dublincore.org/documents/dcmi-namespace/">
```

- **Content:** l'objectiu de content és incloure el contingut d'un web dins d'un canal RSS.

```

1  <rss version="2.0"
2   xmlns:content="http://purl.org/rss/1.0/modules/content/">
```

- **Yahoo Media RSS:** afegeix suport multimèdia. Incorpora tota una sèrie d'elements i atributs pensats per donar millor suport multimèdia a RSS.

```

1  <rss version="2.0"
2   xmlns:media="http://search.yahoo.com/mrss/">
```

- **BlogChannel RSS:** afegeix un grup de funcionalitats pensades per treballar amb blogs.

```

1  <rss version="2.0"
2   xmlns:C="http://backend.userland.com/blogChannelModule/">
```

Però com que estem parlant de vocabularis XML, això implica que qualsevol pot crear la seva extensió pròpia d'RSS.

S'ha de tenir en compte que **no sempre tots els lectors d'RSS comprenen les extensions**. Per tant, la recomanació general és només fer servir les extensions que siguin absolutament necessàries per al contingut que es vol transmetre.

S'ha d'anar amb compte en fer servir extensions perquè no tots els lectors d'RSS les entenen.

## 1.3 Atom

Atom va ser dissenyat pensant a **superar els problemes d'interpretació que tenia RSS 2.0** i evitar la complexitat afegida d'RSS 1.0. La seva idea era aprofitar les millors coses dels RSS i arreglar les parts que en causaven confusió.

Un segon objectiu que el diferencia clarament d'RSS és que també es volia que no solament servís per recuperar els canvis en la informació del canal sinó **que també es pogués fer servir de manera estandarditzada per afegir-hi informació**. La sindicació ha estat molt lligada als blogs i fins al moment cada programa per fer blogs feia servir el seu protocol propi (Blogger API, MetaWebLog API, ...), que estaven pensats només per un blog en concret.

Per tant, podem dividir Atom en dues parts:

- **Atom syndication format**: un llenguatge XML per sindicar continguts.
- **Atom publishing protocol**: un protocol basat en HTTP pensat per actualitzar i crear recursos en el Web.

Va ser desenvolupat per un comitè de l'IETF (Internet Engineering Task Force) i va ser publicat en dos RFC (RFC 4287 i RFC 5023).

A part de les diferències en les etiquetes, les grans diferències amb RSS són que:

- Permet definir quin és el contingut de les etiquetes (text, HTML, etc.), però també permet referències a arxius externs. En RSS no es defineix quin contingut hi ha.
- Es pot fer servir dins d'altres documents XML, ja que té la seva definició i fa servir els espais de noms. No es pot posar RSS dins d'altres documents XML perquè no té en compte l'espai de noms.
- RSS no ofereix un protocol de publicació com Atom.

Però Atom, malgrat el suport obtingut (va ser adoptat per Google), no ha pogut superar RSS 2.0.

### 1.3.1 Llenguatge Atom

Com a bon llenguatge XML, Atom té només una arrel, que és `<feed>`. Aquesta etiqueta la poden fer servir els programes per detectar que el document que estan llegint és de tipus Atom.

L'arrel `<feed>` sempre ha de tenir definit l'espai de noms dels documents Atom, que és <http://www.w3.org/2005/Atom>. Si no s'especifica l'espai de noms el document no validarà.

```

1 <feed xmlns="http://www.w3.org/2005/Atom">
2   ...
3 </feed>
```

El fet de tenir un espai de noms i de fer-lo servir possibilita que els documents Atom es puguin mesclar amb documents XML d'altres vocabularis sense problemes.

Atom té disponibles els atributs d'XML `xml:lang`, que serveix per identificar l'idioma del document, i `xml:base`, que es fa servir per controlar com es resolen les adreces relatives.

De la mateixa manera que en altres llenguatges de canals, com RSS, es poden agrupar les etiquetes d'Atom en dos grups:

- Etiquetes que proporcionen dades sobre el canal.
- Etiquetes amb el contingut del canal.

### **Etiquetes amb dades del canal**

Els elements obligatoris dins de l'etiqueta `<feed>` sempre han de tenir els elements fills `<title>`, `<id>` i `<updated>` (taula 1.4).

**TAULA 1.4.** Elements obligatoris en Atom

Element	Ús
<code>title</code>	És el nom del canal. Normalment és el de la pàgina web que l'ha creat.
<code>updated</code>	Darrer cop que es va actualitzar el canal.
<code>id</code>	Identificador únic i universal del canal. Si es té una adreça web que no està prevista que canviï en molt de temps es pot fer servir aquesta adreça com a <code>id</code> .

Per tant, aquest seria un document Atom vàlid, ja que aquestes són les úniques etiquetes obligatòries:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <feed xmlns="http://www.w3.org/2005/Atom">
3
4   <title>Atom IOC</title>
5   <updated>2011-08-13T15:20:02Z</updated>
6   <id>http://ioc.xtec.cat/</id>
7
8 </feed>
```

A part dels elements obligatoris, en l'especificació d'Atom també es fa referència a uns elements que anomena “molt recomanables” (taula 1.5). Els elements recomanables són `<link>` i `<author>`.

**TAULA 1.5.** Elements molt recomanables en Atom

Element	Ús
<code>link</code>	Identifica la pàgina web equivalent al canal. S'hi defineix la relació amb l'atribut <code>rel</code> .
<code>author</code>	Es tracta d'un element que permet identificar l'autor del canal. Hi pot haver múltiples autors.

I lògicament també hi ha tota una llista d'elements opcionals que es fan servir per aportar informació extra sobre el canal (taula 1.6).

**TAULA 1.6.** Elements opcionals

Element	Ús
<code>category</code>	A quina categoria pertany el <i>feed</i> .
<code>contributor</code>	Gent que col·labora. N'hi pot haver tants com calguin.
<code>generator</code>	Identifica el programa que s'ha fet servir per crear el canal.
<code>icon</code>	Enllaç a una icona identificativa del canal.
<code>logo</code>	Enllaç al logotip del canal.
<code>rights</code>	Informació sobre els drets d'ús del canal.
<code>subtitle</code>	Títol secundari del canal.

### Etiquetes de contingut del canal

Per definir les diferents entrades dins d'un canal es fa servir com a base l'element `<entry>`. Cada nova aportació crearà un nou element `<entry>`, que com a mínim ha de tenir les etiquetes `<title>`, `<id>`, `<updated>`, i a més un element `<content>` o bé un element `<link>` (taula 1.7).

**TAULA 1.7.** Elements obligatoris de "entry"

Element	Ús
<code>title</code>	Títol de l'entrada que es publica.
<code>id</code>	Identificador únic de l'entrada. Sol ser la URI de la pàgina on és.
<code>updated</code>	Data en què es va crear o actualitzar l'entrada.
<code>content</code>	El contingut de l'entrada. S'hi pot especificar de quin tipus és el contingut.
<code>link</code>	Conté l'URL de l'entrada. N'hi pot haver diverses sempre que es canviï el tipus o s'apunti a un lloc diferent.

Com fa sovint l'especificació, Atom també defineix un segon nivell d'elements, considerats “molt recomanats”, i que, per tant, també haurien de sortir. Entre els recomanats hi haurà `<content>` o `<link>` si no s'han especificat anteriorment (taula 1.8).

L'element `<author>` es converteix en obligatori si no se n'ha especificat cap en les metadades del canal.

**TAULA 1.8.** Elements molt recomanats a 'entry'

Element	Ús
author	Nom de l'autor de l'entrada. N'hi pot haver diversos.
summary	Resum del contingut. Pot tenir l'atribut que indiqui de quin tipus és el contingut que té.

I lògicament també hi ha elements opcionals (taula 1.9).

**TAULA 1.9.** Elements opcionals a 'entry'

Element	Ús
category	Categoría de l'entrada. N'hi pot haver diverses.
published	Data en què es va crear l'entrada.
rights	Informació del copyright associada a l'entrada.
source	Informació sobre d'on prové l'entrada. Es fa servir en cas que l'entrada vingui d'un altre lloc i s'estigui reaprofitant.
contributor	Gent que ha col·laborat en l'entrada. Hi pot haver diversos noms.

## Descripció de persones

Les etiquetes <author> i <contributor> serveixen per a descriure una persona. Les persones en Atom es defineixen fent servir una estructura XML que pot tenir els tres elements que es poden veure a la taula 1.10.

**TAULA 1.10.** Descripció de persones

Etiqueta	Ús
name	Nom de la persona.
email	Correu electrònic.
uri	Adreça de la seva pàgina web.

De tots tres elements només <name> és obligatori; els altres són opcionals. Per tant, podem definir el contingut de l'element <author> d'aquesta manera:

```

1 <author>
2   <name>Pere Garcia</name>
3 </author>
```

Però mai deixant el nom. I fer-ho d'aquesta manera seria incorrecte perquè no hi ha l'element <name>.

```

1 <author>
2   <email>pere@ioc.cat</email>
3   <uri>http://ioc.xtec.cat/pere</uri>
```

## Format de les dates

Atom fa servir l'RFC 3339 (ISO 8601) per definir el format de les dates. Les dates en Atom han de tenir aquesta forma:

<sup>1</sup> Any–Mes–DiaTHora:Minuts:Segons–zonahoraria

De manera que:

- Tots els valors són numèrics excepte la zona horària, que en alguns casos pot ser el caràcter “Z” per indicar l'hora universal.
- Davant de la zona horària s'especifiquen les hores de retard o d'avancament amb els símbols de suma o resta.
- Es fa servir la lletra “T” per separar els dies de les hores.

Per tant aquests valors serien correctes:

- 2011-08-13T19:16:20-00:00
- 2011-08-13T19:16:20Z

## El contingut

Si no s'especifica cap atribut en l'element `<content>` o en `<summary>`, aquest serà tractat com si fos text pla. Si es vol deixar clar que el contingut és en algun altre format s'ha d'especificar amb l'atribut `type`, que normalment tindrà els valors “text”, “html” o “xhtml”.

<sup>1</sup> `<content type="text">Contingut</content>`

També es pot enllaçar a contingut extern via una adreça en l'atribut `src` i podem especificar el tipus d'atribut amb `type`.

<sup>1</sup> `<content src="http://ioc.xtec.cat/Hola.mp3" type="audio/mpeg" />`

## 1.4 Validació

Com que tant RSS com Atom són documents XML, es podrà comprovar que són correctes fent servir les mateixes eines de comprovació que es fan servir en XML.

Malgrat que és possible fer servir els validadors d'XML, el més normal és fer servir programes específics per validar RSS i Atom. I com que aquests dos

vocabularis només tenen sentit en xarxa, els validadors més populars d'Atom i RSS són en línia.

Entre els validadors d'RSS i Atom destaquen:

- Feed Validator (<http://feedvalidator.org>)
- Feed Validation Service de W3C (<http://validator.w3.org/feed/>)
- Validome (<http://www.validome.org/rss-atom/>)

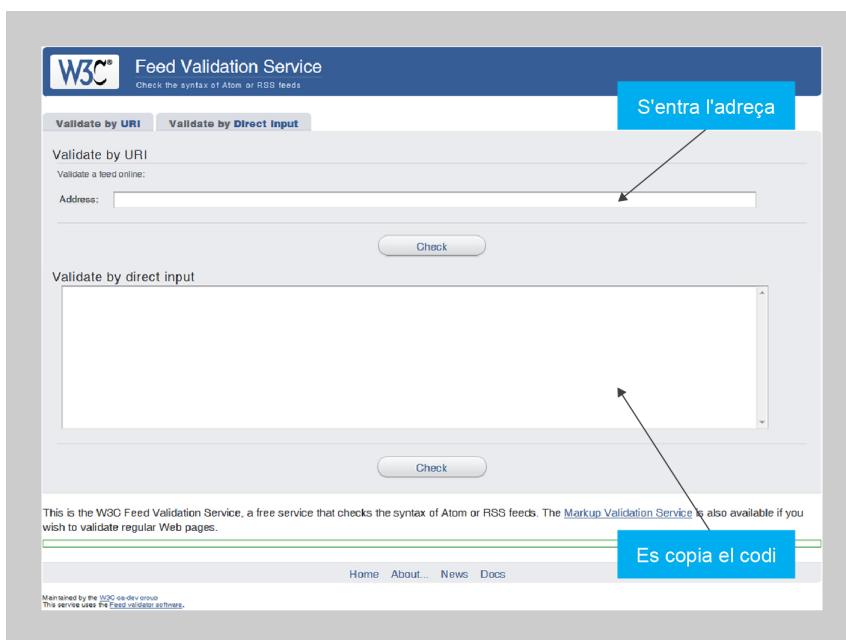
Com a exemple es veurà com es fa per validar un document RSS fent servir un d'aquests validadors (el procediment fent servir Atom és exactament el mateix).

#### Exemple de validació d'un document RSS

Un cop s'ha generat el fitxer del canal, per validar-lo cal accedir a l'adreça del validador amb qualsevol navegador que suporti Javascript. En aquest cas anem a l'adreça del W3C (<http://validator.w3.org/feed/>).

El validador del W3C permet validar els documents RSS tant a partir d'una adreça d'Internet com copiant el fitxer en el web mateix, tal com es pot veure en la figura 1.5.

**FIGURA 1.5.** Possibles maneres de validar RSS en el web del W3C



Independentment de quin sigui el sistema triat, si se li passa aquest codi:

```

1  <rss version="2.0">
2      <channel>
3          <title>Canal de Llenguatges de Marques</title>
4          <link>http://ioc.xtec.cat/rss/Marques.html</link>
5          <description>Canal per entrar les notes del mòdul 4 d'ASIX</
6              description>
7          <item>
8              <link>http://ioc.xtec.cat/marques/RSS.html</link>
9          </item>
10     </channel>
11 </rss>
```

El programa detectarà una sèrie d'errors (figura 1.6). S'ha d'anar amb compte perquè

aquest validador no diferencia gaire l'aspecte dels errors dels recomanacions.

**FIGURA 1.6.** Errors detectats en validar el document

```
Sorry

This feed does not validate.
line 1, column 6: item must contain either title or description [help]
    </item>

In addition, interoperability with the widest range of feed readers could be improved by implementing the following recommendations.
line 11, column 6: item should contain a guid element [help]
    </item>

line 12, column 3: Missing atom:link with rel="self" [help]
    </channel>
```

Es pot obtenir més informació dels errors detectats clicant en l'enllaç "[help]". Aquest enllaç porta a una nova pàgina on es descriu l'error amb detall i en alguns casos explica com solucionar-lo. En l'exemple, l'error és que l'element `<item>` ha de contenir o bé un `<title>` o bé un `<description>`. S'hi afegeixen tots dos i tornem a validar el document.

```
1  <rss version="2.0">
2      <channel>
3          <title>Canal de Llenguatges de Marques</title>
4          <link>http://ioc.xtec.cat/rss/Marques.html</link>
5          <description>Canal per entrar les notes del mòdul 4 d'ASIX</
               description>
6      </channel>
7      <item>
8          <title>Exemple de com validar RSS</title>
9          <link>http://ioc.xtec.cat/marques/RSS.html</link>
10         <description>
11             Això és un exemple per veure com es fa per validar RSS
12         </description>
13     </item>
14 </rss>
```

Ara el document ja és vàlid, com es pot veure en la figura 1.7.

**FIGURA 1.7.** Document vàlid

```
Congratulations!

 This is a valid RSS feed.

Recommendations

This feed is valid, but interoperability with the widest range of feed readers could be improved by implementing the following recommendations.
line 13, column 4: item should contain a guid element [help]
    </item>

line 14, column 4: Missing atom:link with rel="self" [help]
    </channel>
```

A pesar d'això encara hi surten les recomanacions per a no tenir problemes de compatibilitat. En aquest els problemes són:

- Que s'afegeixi l'espai de noms d'Atom.
- Que s'hi defineixi un guid.

Definir el `guid` és relativament senzill. Es defineix dins de l'element `<entry>` un identificador únic que pot ser una URL.

```
1  <guid>http://ioc.xtec.cat/rss/Marques.html</guid>
```

I l'altra recomanació és que s'afegeixi l'espai de noms d'Atom i un enllaç al document Atom. Per tant, s'afegeix l'espai de noms a `<rss>`.

```
1 <rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
```

I dins de <entry> s'afegeix l'enllaç Atom.

```
1 <atom:link href="http://ioc.xtec.cat/rss/Marques.xml" rel="self"
2 type="application/rss+xml" />
```

El resultat final serà aquest:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
3   <channel>
4     <title>Canal de Llenguatges de Marques</title>
5     <link>http://ioc.xtec.cat/rss/Marques.xml</link>
6     <description>Canal per entrar les notes del mòdul 4 d'ASIX</
      description>
7     <atom:link href="http://ioc.xtec.cat/rss/Marques.xml" rel="
      self"
      type="application/rss+xml" />
8   <item>
9     <title>Exemple de com validar RSS</title>
10    <link>http://ioc.xtec.cat/marques/RSS.html</link>
11    <description>
12      Això és un exemple per veure com es fa per validar RSS.
13    </description>
14    <guid>http://ioc.xtec.cat/rss/Marques.html</guid>
15  </item>
16 </channel>
17 </rss>
```

El document ara validarà correctament.

## 1.5 Agregadors / lectors

Els agregadors i lectors de *feeds* són programes que permeten a l'usuari mantenir en un sol lloc tota la informació dels canals que li interessen.

Entre altres coses:

- S'encarreguen d'actualitzar els canvis que s'hi van produint sense que l'usuari hagi de visitar la pàgina.
- Porten el control dels continguts llegits i no llegits dels canals.
- Permeten veure un resum de les notícies d'un lloc web.
- S'hi poden organitzar les notícies en grups personalitzats.
- Permeten fer cerques d'informació entre la informació del canal.

Tant RSS com Atom són estàndards oberts, i això ha permès que s'hagin creat una gran quantitat de lectors que els suporten i que ofereixen funcions extra per als usuaris per intentar millorar-ne l'experiència.

A pesar que es poden llegir els canals des de diferents programes de correu o els navegadors, normalment s'aconsegueix treballar de manera més còmoda i personalitzable fent servir programes especialitzats.

En general podem dividir els programes lectors de canals en dos grans grups:

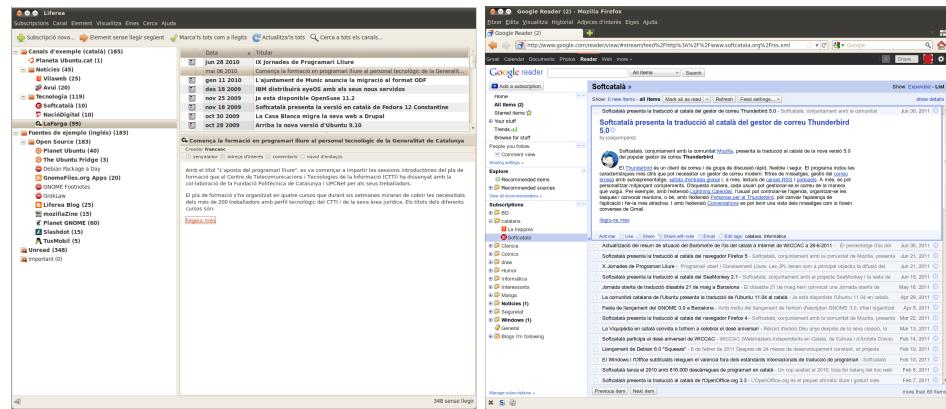
- Lectors web
- Lectors d'escriptori

A pesar de la divisió normalment l'aspecte visual d'aquests programes és relativament semblant. Tots solen tenir la pantalla dividida en blocs:

- En un dels blocs hi sol haver la llista de subscripcions en la qual es poden agrupar aquestes subscripcions per temes.
- Un bloc per mostrar el contingut de cada una de les subscripcions.

En la imatge següent podem veure la semblança entre l'aspecte d'un programa web com el Google Reader i un programa d'escriptori com Liferea (figura 1.8).

**FIGURA 1.8.** Comparació entre el Google Reader i el Liferea



La gran diferència sol estar en els filtres o les característiques extra que ofereixen cada un dels programes per fer l'ús més interessant per als usuaris: visualitzacions originals, estadístiques, etc.

### 1.5.1 Lectors via web

Alguns dels agregadors més populars són els que s'executen directament des d'un lloc web. Aquestes aplicacions se n'encarreguen d'ofrir les funcions d'un agregador sense que l'usuari hagi d'instal·lar res.

Si s'hi afegeix el fet de s'hi pot accedir per mitjà de diferents dispositius (ordinadors, telèfons mòbils, etc.) tot plegat ha propiciat que aquests siguin els lectors més populars.

Hi ha una gran quantitat d'aplicacions web disponibles a Internet i sovint n'apareixen de noves. Alguns exemples són:

- Google Reader
- Netvibes
- BlogLines
- FeedLooks

### 1.5.2 Programari d'escriptori

Són programes tradicionals que requereixen ser instal·lats en el sistema de l'usuari. Solen tenir interfícies gràfiques semblants a les dels programes de correu electrònic.

N'hi ha una gran quantitat i amb les eines de programació actuals no és gaire difícil programar-ne un. Alguns exemples són:

- Liferea
- SharpReader
- FeedDemon
- FeedReader

## 1.6 Problemes

No tot són avantatges amb la sindicació de continguts, ja que alguns usuaris s'han queixat que el fet que sigui més fàcil i ràpid obtenir la informació **no sempre fa que es perdi menys temps per revisar la informació**, ja que aquesta facilitat fa que els usuaris acabin seguint més llocs i, per tant, acabin perdent el mateix temps que abans, fins i tot més.

D'altra banda, malgrat que la sindicació de continguts s'ha fet molt popular, encara hi ha **molts usuaris que no entenen per a què serveix** i no la fan servir.

També alguns usuaris creadors dels continguts han expressat les seves reserves a la sindicació, ja que sovint els resta ingressos per publicitat: el fet que els usuaris no hagin de visitar els seus llocs web els resta visites i, per tant, acaben tenint menys ingressos.

De totes maneres, des d'un punt de vista tècnic i d'utilitat és una tecnologia que cal tenir en compte, atès que els avantatges són molt superiors als possibles inconvenients.



## 2. Conversió i adaptació de documents XML

A pesar que l'XML és un format relativament lleigible en ser visualitzat, aquest no és un dels seus objectius principals, i menys si es té en compte que als humans els agrada llegir les dades col·locades en determinats formats que els facin la lectura més agradable –mentre que l'especificació XML exposa que en cap cas no s'ha d'incloure informació sobre la visualització en els documents.

En determinats casos pot caldre transformar els documents XML perquè siguin més fàcils de visualitzar, o també adaptar-los perquè puguin ser llegits per programes específics.

XML està pensat sobretot per a **emmagatzemar i intercanviar informació**, de manera que si cal representar les dades d'una manera diferent per optimitzar un procés o per millorar-ne la visualització hi haurà diverses possibilitats:

1. **Desenvolupar un programa:** com que és relativament senzill treballar amb XML, es podria desenvolupar un programa que agafi les dades XML i generi la sortida tal com la volem. Això té l'inconvenient que caldrà tenir coneixements de programació i que pot representar molta feina encara que el que calgui fer sigui trivial.
2. **Fer servir CSS:** en molts casos una solució senzilla seria fer servir CSS per representar la informació de manera més amigable fent servir un navegador. Però només serveix per canviar la visualització, no per canviar el document.
3. **Transformar el document:** una solució alternativa consisteix a transformar el document en un altre que estigui pensat per ser visualitzat. Hi ha molt formats que estan pensats sobretot per ser visualitzats: PDF, HTML, XHTML, etc.

### 2.1 Ús de CSS

En molts casos si l'objectiu és que el document sigui visualitzat d'una manera més agradable la solució més senzilla pot ser visualitzar-lo mitjançant CSS.

CSS és interessant sobretot quan es vulguin fer adaptacions senzilles. Si es té un document XML com aquest:

```

1 <?xml version="1.0" ?>
2 <professor>
3   <nom>Marcel</nom>
4   <cognom>Garcia</cognom>
5   <departament>Departament d'Informàtica</departament>
6   <carrecs>
7     <carrec>Cap de Departament</carrec>
```

```

8      <carrec>Tutor</carrec>
9  </carreccs>
10 </professor>

```

...i es vol representar com una targeta de vista, es pot fer servir un document CSS que contingui aquestes línies:

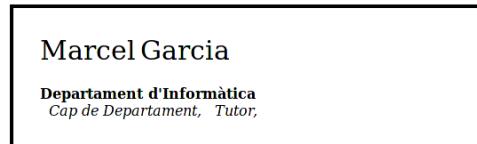
```

1 professor {
2     padding: 30px;
3     margin: 30px;
4     border: 4px black solid;
5     width: 40%;
6 }
7
8 nom,cognom {
9     font-size:30px;
10 }
11
12 departament {
13     padding-top:20px;
14     display:block;
15     font-weight:bold;
16 }
17
18 carrec{
19     font-style: italic;
20     padding-left:10px;
21 }
22
23 carrec:after { content:","; }

```

Així, es representarà el document XML com en la figura 2.1.

**FIGURA 2.1.** Resultat de la transformació



Malgrat això, CSS té moltes limitacions a l'hora de presentar la informació:

1. La informació no pot ser reordenada com vulguem. L'única manera de simular el canvi d'ordre és fer servir posicionaments absoluts.
2. Els atributs es poden mostrar però hi ha moltes limitacions per fer-ho.
3. No es poden afegir estructures noves producte de càlculs o de procés de les dades del document.
4. No té maneres senzilles de formatar les dades en pàgines de text per ser impreses.

Si l'objectiu final no és simplement decorar el fitxer sinó transformar-lo en un altre document totalment diferent, CSS no serveix. CSS no transforma el document sinó que simplement canvia la manera com es visualitza.

## 2.2 Transformació de documents

Per intentar aconseguir fer tot allò que CSS no podia fer es va crear un nou llenguatge de plantilles: XSL (*extensible stylesheet language*).

Inicialment es van concentrar a poder representar la informació de manera que pogués ser mostrada en documents impresos i en pantalla però després es va acabar definint un sistema per fer transformacions genèriques de documents XML en altres coses: documents de text, documents XML, pàgines web, etc.

Actualment XSL és una família de llenguatges que serveixen per definir transformacions i presentacions de documents XML.

La família XSL està formada per tres llenguatges:

- **XSL-FO** (*XSL formatting objects*): un llenguatge per definir el format que s'ha d'aplicar a un document.
- **XSLT** (*XSL transformations*): un llenguatge per transformar documents XML.
- **XPath**: un llenguatge per accedir a parts dels documents XML.

### 2.2.1 XSL-FO

XSL-FO és un llenguatge basat en XML que està pensat per donar format als documents XML, però a diferència d'altres llenguatges amb objectius similars, com CSS o XHTML, està pensat per generar sortides tant per a formats de pantalla com per a formats paginats.

XSL-FO és un llenguatge que:

- Permet especificar amb molta precisió el contingut d'un document (paginació, organització, estil, etc.).
- Permet crear documents d'alta qualitat.
- És ideal per generar documents amb dades que canvien sovint.

XSL-FO sobretot es fa servir per generar documents en formats pensats per ser impresos com PDF o Postscript.

És corrent que la generació del document es faci en dues fases (figura 2.2):

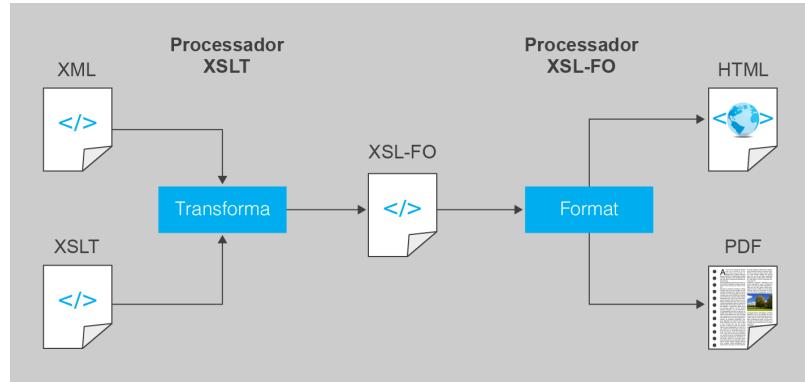
1. Transformació del document XML en un document XSL-FO amb el llenguatge de transformacions XSLT.

#### Generació de PDF

Un dels processadors d'XSL-FO més populars és l'Apache FOP, que permet crear documents en PDF a partir de documents XSL-FO.

2. Transformació del document XSL-FO en el format que volem amb un processador XSL-FO.

**FIGURA 2.2.** Transformació d'un document XML en PDF i HTML



Per tant, els processadors XSL-FO s'encarreguen de generar el format que volem a partir de les dades especificades en els documents XSL-FO.

## 2.3 Processadors XPath o XSLT

En general els processadors XPath o XSLT es proporcionaran per mitjà de biblioteques que podran ser cridades des dels programes.

Això permet simplificar tot el procés de creació de programes que facin servir XPath o XSLT, ja que quan una aplicació es vulgui aprofitar de les característiques d'alguns d'aquests llenguatges no ho haurà d'implementar tot de nou sinó simplement fer servir les biblioteques.

### 2.3.1 Biblioteques

En alguns casos les biblioteques tindran suport tant per a XSLT com per a XPath, i en d'altres casos s'haurà de recórrer a dues biblioteques diferents. Per exemple, en el cas del **suport XML del projecte Gnome**, per poder tenir suport XPath i XSLT hem de recórrer a dues biblioteques diferents:

- **libxml2**: ofereix suport per processar XML i XPath.
- **libxslt**: agafa de base libxml2 per permetre fer transformacions XSLT.

Mentre que amb **Saxon** el suport XPath i XSLT està en la mateixa biblioteca.

Les biblioteques més usades són:

- libxml2 i libxslt

- Apache Xalan
- Saxon
- MSXML

A l'hora de triar una biblioteca o una altra cal tenir clar si necessitem el suport per a les versions 2.0 d'aquests llenguatges, ja que no hi sol haver gaire problemes per fer transformacions amb les versions 1.0, i tots els processadors les solen suportar; no sempre passa el mateix amb les versions 2.0.

### **libxml2 i libxslt**

Del projecte GNOME han sortit les biblioteques *libxml2* i *libxslt*, que són biblioteques que permeten tenir suport per a XML, XPath i transformacions XSLT.

La utilitat *xmllint* de *libxml2* permet executar expressions XPath de dues maneres. De manera interactiva amb el paràmetre **-shell**. Amb aquest paràmetre entrem en un entorn d'ordres que ens permet executar ordres contra el fitxer.

Una de les ordres possibles dins d'aquest entorn és **xpath**, que ens donarà informació sobre com seran els resultats.

```

1 $ xmllint --shell personas.xml
2 / > xpath //nom
3 Object is a Node Set :
4 Set contains 4 nodes:
5 1 ELEMENT nom
6 2 ELEMENT nom
7 3 ELEMENT nom
8 4 ELEMENT nom

```

Per veure els resultats d'una manera més amigable és més útil fer servir **cat**.

```

1 $ xmllint --shell personas.xml
2 / > cat //nom
3
4 <nom>Marcel Puig</nom>
5
6 <nom>Frederic Pi</nom>
7
8 <nom>Filomeno Garcia</nom>
9
10 <nom>Manel Puigdevall</nom>

```

L'altra possibilitat consisteix a especificar l'expressió XPath en la línia d'ordres fent servir el paràmetre **-xpath**:

```

1 $ xmllint --xpath //nom personas.xml
2 <nom>Marcel Puig</nom>
3 <nom>Frederic Pi</nom>
4 <nom>Filomeno Garcia</nom>
5 <nom>Manel Puigdevall</nom>

```

La biblioteca *libxslt* porta una utilitat des de l'entorn d'ordres que permet fer transformacions XSLT, que s'anomena **xsltproc**.

Amb **xsltproc** es pot transformar un fitxer en un altre passant-li com a paràmetres el fitxer per transformar i la plantilla.

```
1 $ xsltproc fitxer.xml fitxer.xsl -o sortida
```

Es poden trobar versions d'**xsltproc** compilades per a Windows i que funcionaran de la mateixa manera.

### Apache Xalan

Apache Xalan és un intent de fer unes biblioteques de C++ i de Java de qualitat comercial, lliures i multiplataforma.

En els exemples que vénen amb la biblioteca es pot trobar un programa d'exemple amb el qual es poden fer transformacions XPath anomenat **XPathResolver**.

En el mateix paquet es pot trobar una utilitat d'entorn d'ordres que permet fer transformació de documents.

```
1 $ xalan -in fitxer.xml -xsl fitxer.xsl -out sortida
```

### Saxon

Saxon és el processador més complet de tots, ja que està desenvolupat per un dels especificadors d'XSLT, i per tant segueix fidelment l'estàndard.

Hi ha dues versions de Saxon, una per a Java i una per a .NET. Té una versió de les biblioteques de codi obert, Saxon-HE (Home Edition), i dues de comercials, Saxon-PE (Professional Edition) i Saxon-EE (Enterprise Edition).

Amb la biblioteca es poden trobar exemples d'ús que es poden fer servir per fer proves i transformacions.

En sistemes basats en Debian, com l'Ubuntu, després d'instal·lar el paquet *libsaxonb-java* es pot executar una consulta XPath passant l'expressió amb echo a **saxonb-xquery**.

```
1 $ echo //nom | saxonb-xquery -s personnes.xml -
```

Es poden fer transformacions XSLT amb l'ordre **saxonb-xsbt** passant-hi el fitxer XML i el fitxer de plantilles XSLT

```
1 $ saxonb-xsbt -o sortida personnes.xml personnes.xsl
```

### Microsoft XML Parser

Es tracta del més usat en el món de la programació en .NET. Es pot descarregar des del web de Microsoft.

L'execució des de consola no difereix gaire de les dels altres processadors. Per exemple, per fer transformacions podem fer:

```
1 c:\> msxml fitxer.xml fitxer.xsl -o sortida
```

## 2.4 XPath

XPath (*XML path language*) és una manera d'especificar parts d'un document XML que té eines per manipular el contingut de les dades de text, numèriques, etc.

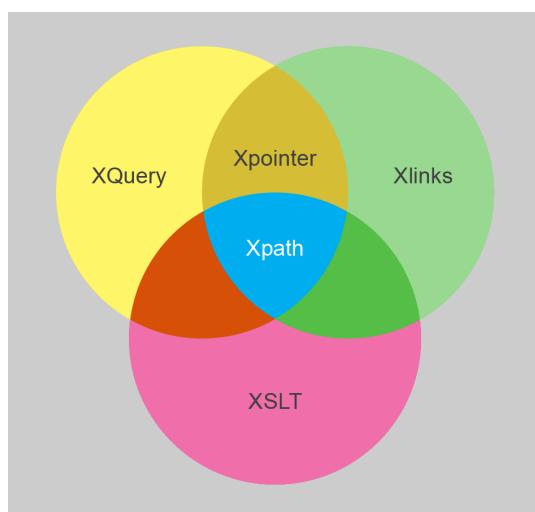
XPath és una recomanació del W3C (<http://www.w3.org/TR/xpath>), que tot i que serveix per treballar amb XML no és un llenguatge XML. La idea és que en no estar basat en XML es podrà incloure en altres llenguatges XML sense haver de preocupar-se de si el resultat està ben format o no.

La base del funcionament d'XPath és l'avaluació d'expressions. Una expressió que s'avaluarà contra un document XML i ens donarà un resultat que pot ser de diferents tipus:

1. Un boleà: cert o fals
2. Un nombre
3. Una cadena de caràcters
4. Un grup d'elements

XPath està desenvolupat pels comitès de creació d'XSL i XQuery i s'ha convertit en un component essencial per a diferents llenguatges XML com XLinks, XSLT i XQuery, com es pot veure en la figura 2.3.

**FIGURA 2.3.** Relació d'XPath amb diferents llenguatges XML



La versió 2.0 d'XPath està tan integrada dins d'XQuery que qualsevol expressió XPath és també automàticament una expressió XQuery correcta.

### 2.4.1 Vista d'arbre

XPath tracta tots els documents XML des del punt de vista d'un arbre de nodes en què hi ha **una arrel** que no es correspon amb l'arrel del document, sinó que **és el document** i es representa amb el símbol “/”.

A part de l'arrel també hi ha nodes per representar els **elements**, els **atributs**, els nodes de **dades**, els **comentaris**, les **instruccions de procés** i els **espais de noms**.

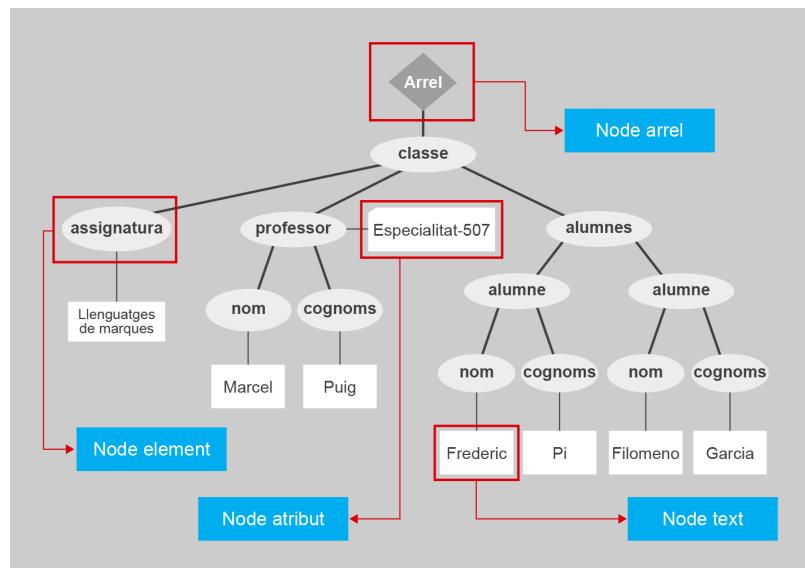
L'exemple següent XML:

```

1  <?xml version=""=1.0 ?>
2  <classe>
3      <assignatura>Llenguatges de marques</assignatura>
4      <professor Especialitat="507">
5          <nom>Marcel</nom>
6          <cognoms>Puig</cognoms>
7      </professor>
8      <alumnes>
9          <alumne>
10         <nom>Frederic</nom>
11         <cognoms>Pi</cognoms>
12     </alumne>
13     <alumne>
14         <nom>Filomeno</nom>
15         <cognoms>Garcia</cognoms>
16     </alumne>
17 </alumnes>
18 </classe>
```

Es representarà en XPath amb un arbre com el de la figura 2.4.

**FIGURA 2.4.** Arbre XPath



En un arbre XPath els atributs no són considerats nodes fills sinó que són **“proprietats”** del node que els conté i els nodes de dades són nodes **sense nom** que només contenen les dades.

## 2.4.2 Programari per avaluar XPath

Molts programes permeten executar una consulta XPath contra un document XML. Hi ha programes específics, editors XML, components dels navegadors web, pàgines web online, etc.

És impossible mostrar tots els programes que permeten treballar amb XPath i, per tant, només se'n mostraran alguns exemples.

### "xpath"

La instal·lació per defecte d'Ubuntu instal·la automàticament una ordre basada en Perl anomenada `xpath` que permet comprovar les ordres XPath des de la consola. L'expressió es posa rere el paràmetre `-e`.

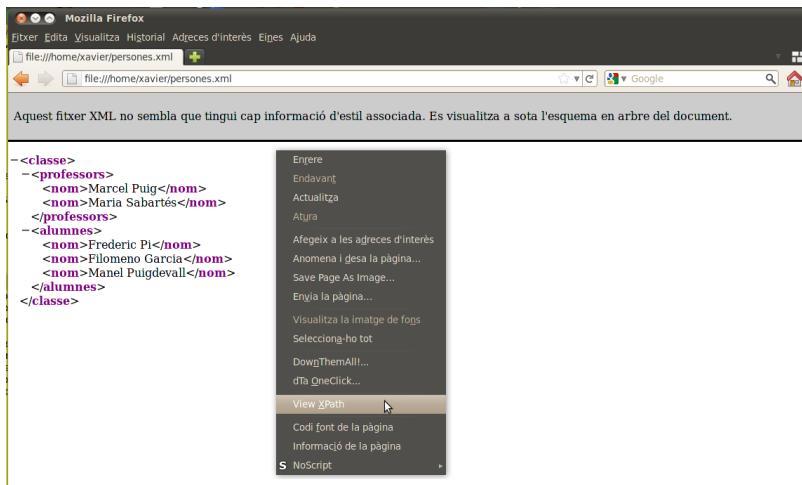
```
1 $ xpath -q -e //nom personas.xml
2 <nom>Marcel Puig</nom>
3 <nom>Frederic Pi</nom>
4 <nom>Filomeno Garcia</nom>
5 <nom>Manel Puigdevall</nom>
```

### XPath Checker

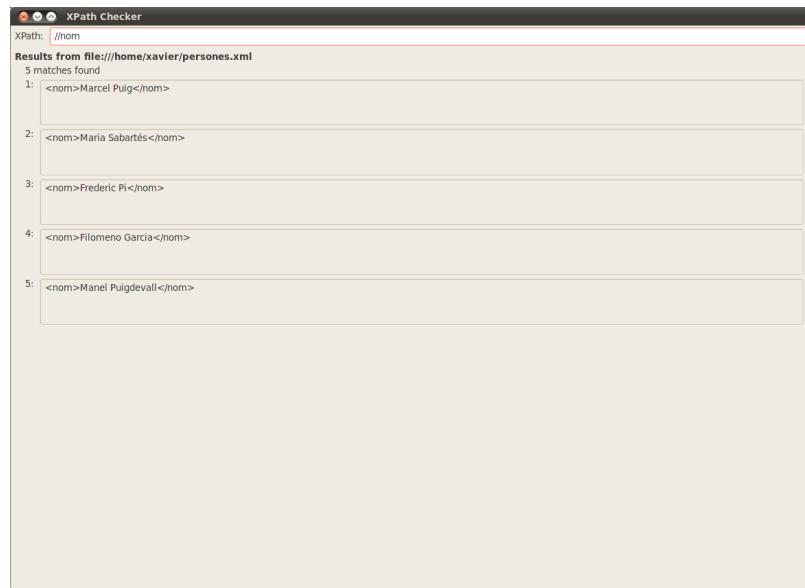
Una utilitat molt interessant és un component del Firefox anomenat **XPath Checker**, que permet fer consultes XPath interactivament, tant en pàgines HTML com en documents XML que s'hagin visualitzat en el navegador.

Un cop instal·lat el component i carregat el document per avaluar amb el botó dret del ratolí, es tria l'opció “View XPath” (figura 2.5).

**FIGURA 2.5.** Engregar XPath Checker

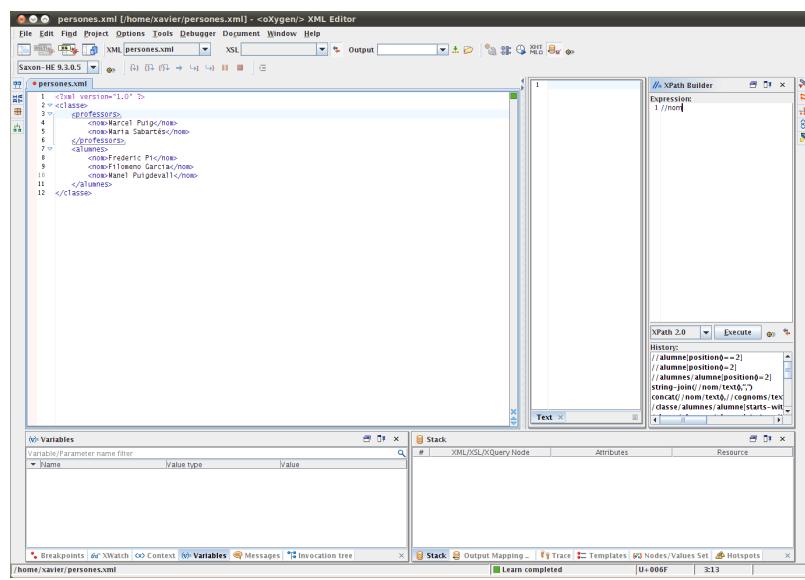


S'obrirà una finestra nova on es poden anar executant expressions XPath i també es mostra el resultat de la consulta en la part inferior de manera interactiva (figura 2.6).

**FIGURA 2.6.** Els resultats van canviant en funció de l'expressió que s'especifiqui

## Editors XML

L'opció més professional és fer servir algun dels editors especialitzats en XML. Aquests editors permeten avaluar expressions XPath des d'un entorn gràfic amb tot tipus d'assistències en la creació de les expressions, possibilitats de depuració, etc. En la imatge següent es pot veure un exemple de l'assistent de l'oXygen XML Editor (figura 2.7).

**FIGURA 2.7.** Assistent de creació d'expressions XPath

### 2.4.3 Navegació

Com que la representació interna del document XML per XPath serà un arbre, s'hi pot navegar especificant-hi camins d'una manera semblant a com es fa en els

directoris dels sistemes operatius.

El més important per tenir en compte a l'hora de crear una expressió XPath és saber el node en el qual està situat el procés (**node de context**), ja que és des d'aquest que s'avaluarà l'expressió. El node de context al principi és en l'arrel però es va movent a mesura que es van avaluant les expressions, i per tant podem expressar els camins XPath de dues maneres:

- **Camins absoluts**
- **Camins relatius**

Expressar camins en XPath s'assembla tant a com ho fan els sistemes operatius que fins i tot es fan servir símbols semblants.

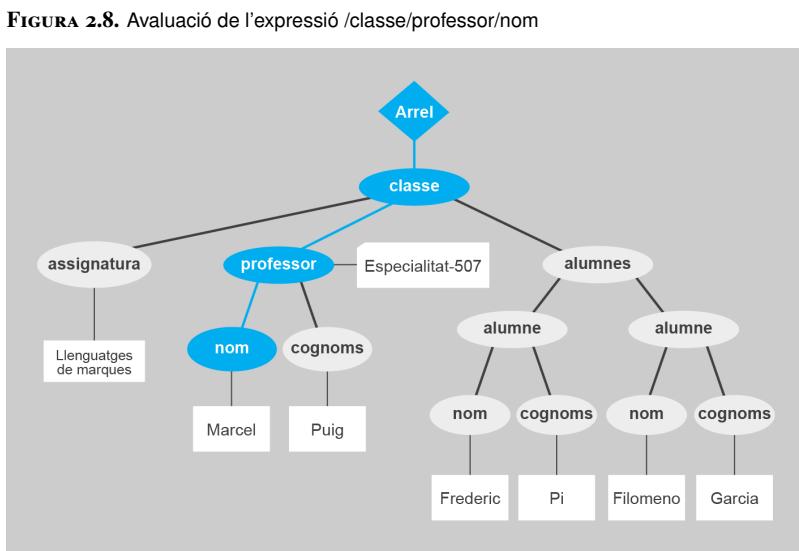
Els **camins absoluts** són camins que sempre comencen en l'arrel de l'arbre. Es poden identificar perquè el primer caràcter de l'expressió sempre serà l'arrel "/". No importa quin sigui el node de context si es fan servir camins absoluts, perquè el resultat sempre serà el mateix.

En canvi, els **camins relatius** parteixen des del node en el qual estem situats.

Per exemple, es pot obtenir el node <nom> del professor de l'exemple que hem especificat anteriorment fent servir l'expressió XPath següent:

<sup>1</sup> /classe/professor/nom

Podem veure com s'avalua l'expressió en l'arbre XPath en la figura 2.8.



Cal tenir en compte que el resultat d'aquesta expressió no és només el contingut de l'element sinó tot l'element <nom>.

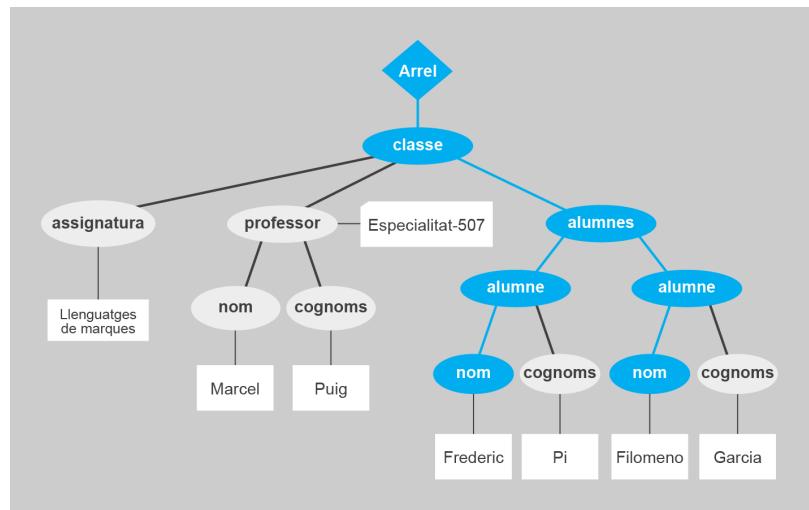
<sup>1</sup> <nom>Marcel</nom>

Mai no s'ha d'oblidar que l'expressió sempre intenta aconseguir el nombre màxim de camins correctes i, per tant, no necessàriament ha de retornar només un sol valor. Per exemple, si l'expressió per avaluar fos la següent:

1 /classe/alumnes/alumne/nom

XPath l'avaluaria intentant aconseguir tots els camins que quadren amb l'expressió, tal com podeu veure visualment en la figura 2.9.

**FIGURA 2.9.** Avaluació de l'expressió /classe/alumnes/alumne/nom



El resultat seran els dos resultats possibles:

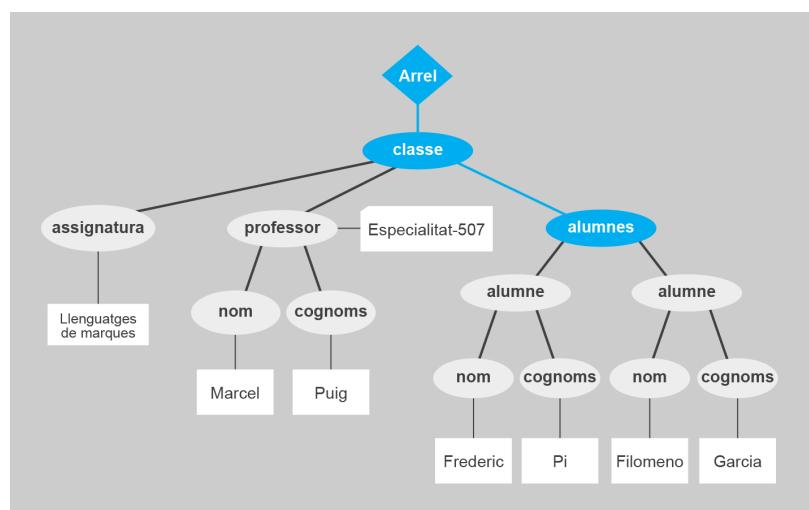
1 <nom>Frederic</nom>  
2 <nom>Filomeno</nom>

A pesar que en els exemples anteriors sempre s'han retornat nodes finals això no necessàriament ha de ser així, ja que XPath pot retornar qualsevol tipus d'element com a resultat.

1 /classe/alumnes

La navegació per l'arbre no difereix gaire dels altres casos (figura 2.10):

**FIGURA 2.10.** Avaluació de l'expressió /classe/alumnes



En aquest cas el resultat no és un element simple sinó que és tot un subarbre d'elements.

```

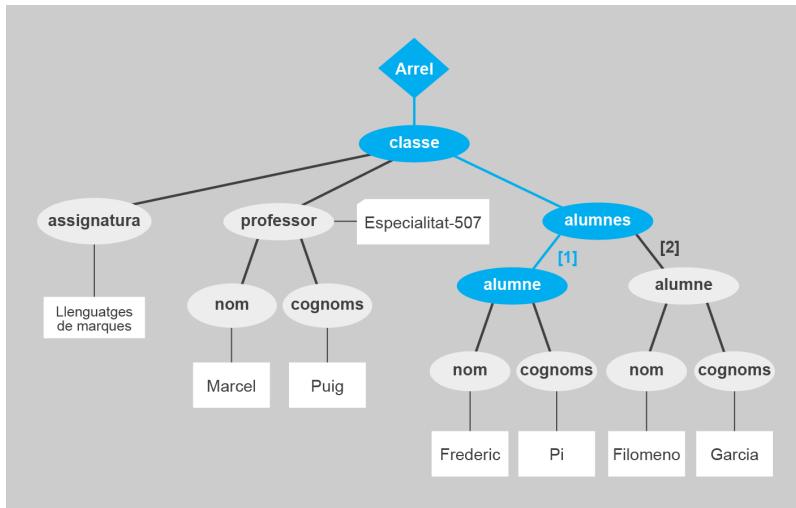
1 <alumnes>
2   <alumne>
3     <nom>Frederic</nom>
4     <cognom>Pi</cognom>
5   </alumne>
6   <alumne>
7     <nom>Filomeno</nom>
8     <cognom>Garcia</cognom>
9   </alumne>
10 </alumnes>
```

Si se sap que una expressió retornarà diversos resultats però només se'n vol un d'específic es pot fer servir un nombre envoltat per claudàtors quadrats "[]" per indicar quin és el que es vol aconseguir. Per retornar només el primer alumne podeu fer el següent:

```
1 /classe/alumnes/alumne[1]
```

Dels dos nodes disponibles com a fills de `<alumnes>`, només se seleccionarà el primer (figura 2.11).

**FIGURA 2.11.** Avaluació de l'expressió /classe/alumnes/alumne/nom



O sigui:

```

1 <alumne>
2   <nom>Frederic</nom>
3   <cognom>Pi</cognom>
4 </alumne>
```

Es poden fer servir els claudàtors en qualsevol lloc de l'expressió per fer determinar quina de les branques es triarà. Per exemple, es pot obtenir només el nom del segon alumne amb una expressió com aquesta:

```
1 /classe/alumnes/alumne[2]/nom
```

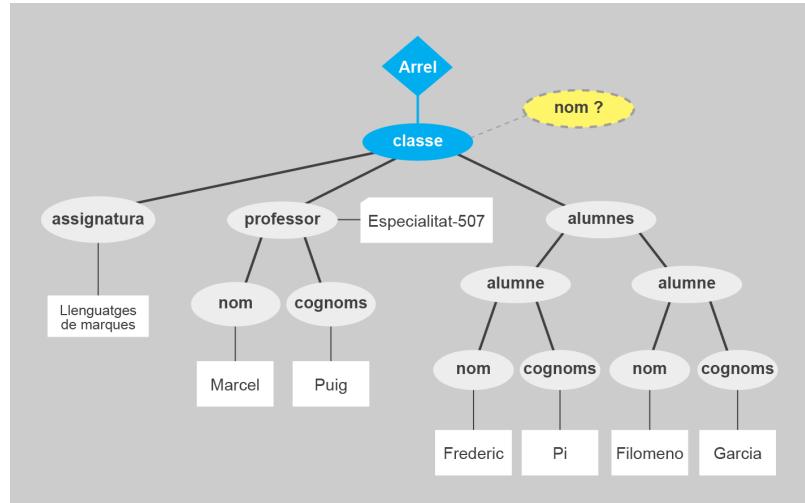
Sempre s'ha d'anar en compte en escriure les expressions XPath, ja que si el camí

especificat no es correspon amb un camí **real** dins de l'arbre no es retornarà cap resultat.

<sup>1</sup> /classe/nom

Com que en arribar al node `classe` no n'hi trobarà cap d'anomenat `<nom>`, no retornarà cap resultat, com podeu veure si seguiu l'arbre (figura 2.12).

**FIGURA 2.12.** Avaluació de l'expressió /classe/alumnes

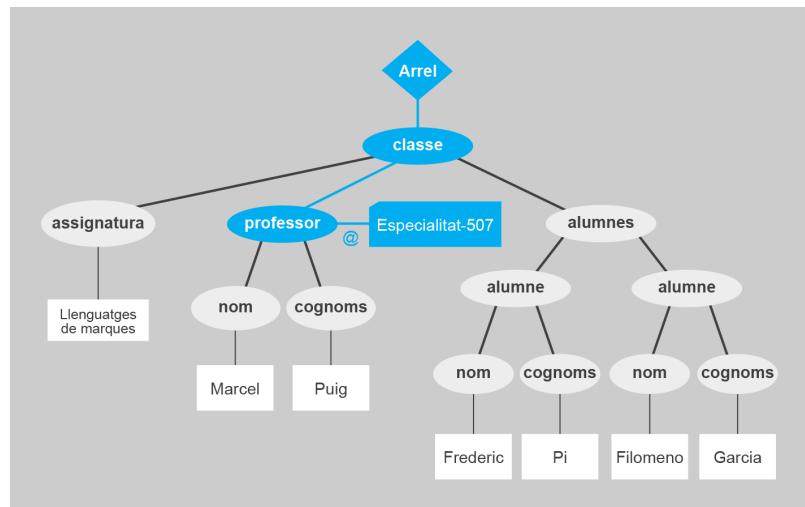


### Obtenir els atributs d'un element

Els valors dels atributs es poden aconseguir especificant el símbol @ davant del nom un cop s'hagi arribat a l'element que el conté (figura 2.13).

<sup>1</sup> /classe/professor/@especialitat

**FIGURA 2.13.** Avaluació de l'expressió /classe/professor/@especialitat



S'ha de tenir en compte que a diferència del que passa amb els elements, en obtenir un atribut no tindrem un element sinó només el seu valor:

<sup>1</sup> 507

## Obtenir el contingut d'un element

Per a aquells casos en què només vulguem el contingut de l'element, s'ha definit la funció **text()** per obtenir aquest contingut. Això s'ha fet així perquè d'altra manera, com que els nodes de text no tenen nom, no s'hi podria accedir.

De manera que si a un element que tingui contingut de dades se li afegeix *text()*:

```
1 /classe/professor/nom/text()
```

...retornarà el contingut del node sense les etiquetes:

```
1 Marcel
```

## Comodins

De la mateixa manera que en els sistemes operatius, es poden fer servir comodins diversos en les expressions XPath. Es poden veure els comodins en la taula 2.1.

**TAULA 2.1.** Comodins en XPath

Comodi	Significat
*	L'asterisc es fa servir per indicar tots els elements d'un determinat nivell.
.	Com en els directoris dels sistemes operatius el punt serveix per indicar el node actual.
..	Es fa servir per indicar el pare del node en el qual estem.
//	Les dobles barres indiquen que quadrarà amb qualsevol cosa des del node en el qual estem. Pot ser un sol element o un arbre de nodes.

Amb l'asterisc es poden obtenir tots els elements d'un determinat nivell. Amb aquesta expressió es poden obtenir tots els elements de dins del node **professor**.

```
1 /classe/professor/*
```

Aquesta expressió retornarà per separat els dos nodes fills de **<professor>** (**<nom>** i **<cognom>**).

```
1 <nom>Marcel</nom>
2 <cognoms>Puig</cognoms>
```

O bé fer servir les dobles barres (//) per obtenir tots els elements **<nom>** del fitxer independentment del lloc on siguin dins del document XML.

```
1 //nom
```

El resultat serà:

```
1 <nom>Marcel</nom>
2 <nom>Frederic</nom>
3 <nom>Filomeno</nom>
```

Es poden posar les dobles barres en qualsevol lloc dins de l'expressió per indicar que hi pot haver qualsevol cosa enmig a partir del lloc on apareguin.

1 /classe/alumnes//nom

Donarà els dos noms dels alumnes:

1 <nom>Frederic</nom>  
2 <nom>Filomeno</nom>

Tot i que facilita la creació d'expressions no és gaire recomanable abusar del comodí // per motius d'eficiència. Les expressions amb aquest comodí requeriran molts més càlculs per ser evaluades, i per tant les expressions trigaran més a donar resultats.

## Eixos XPath

Per avaluar les expressions XPath s'explora un arbre, de manera que també es proporcionen una sèrie d'elements per fer referència a parts de l'arbre. Aquests elements s'anomenen **eixos XPath** (taula 2.2).

**TAULA 2.2.** Eixos XPath

Eix	Significat
self::	El node en el qual està el procés (fa el mateix que el punt)
child::	Fill de l'element actual
parent::	El pare de l'element actual (idèntic a fer servir ..)
attribute::	Es fa servir per obtenir un atribut de l'element actual (@)

Alguns d'aquests eixos pràcticament no es fan servir perquè generalment és més còmode i curt definir les expressions a partir del símbol. Tothom prefereix fer servir una expressió com aquesta:

1 /classe/professor/nom

Que no pas la seva versió equivalent fent servir els eixos:

1 /child)::classe/child)::professor/child)::nom

A part dels vistos en la taula 2.2 n'hi ha d'altres, que en aquest cas no tenen cap símbol que els simplifiqui (taula 2.3).

**TAULA 2.3.** Eixos XPath

Eix	Significat
descendant::	Tots els descendents del node actual
desendant-or-self::	El node actual i els seus descendents
ancestor::	Els ascendents del node

**TAULA 2.3** (continuació)

Eix	Significat
ancestor-or-self::	El node actual i els seus ascendents
precedint::	Tots els elements precedents al node actual
preceding-sibling::	Tots els germans precedents
following::	Elements que segueixen el node actual
following-sibling::	Germans posteriors al node actual
namespace::	Conté l'espai de noms del node actual

## Condicions

Un apartat interessant de les expressions XPath és poder afegir condicions per a la selecció de nodes. A qualsevol expressió XPath se li poden afegir condicions per obtenir només els nodes que compleixin la condició especificada.

La selecció de nodes es fa especificant un predicat XPath dins de claudàtors.

Per exemple, aquesta expressió selecciona només els professors que tinguin un element `<nom>` com a fill de `<professor>`:

```
1 /classe/professor[nom]
```

Si s'aplica l'expressió a l'[exemple](#) que hem fet servir per fer la vista d'arbre, el resultat serà el node `<professor>` que té dins seu `<nom>`.

```
1 <professor especialitat="507">
2   <nom>Marcel</nom>
3   <cognoms>Puig</cognoms>
4 </professor>
```

En el valor de l'expressió s'hi especifiquen camins relatius des del node que tingui la condició. Fent servir condicions es pot fer una expressió que només retorna el professor si té alumnes.

```
1 /classe/professor[../alumnes/alumne]
```

Normalment la complexitat de les condicions va més enllà de comprovar si el node existeix, i es fan servir per comprovar si un node té un valor determinat. Per exemple, per obtenir els professors que es diguin “Marcel”:

```
1 /classe/professor[nom="Marcel"]
```

Les condicions es poden posar en qualsevol lloc del camí i n'hi pot haver tantes com calgui. Per obtenir el cognom del professor que es diu “Marcel” es pot fer servir una expressió com aquesta.

```
1 /classe/professor[nom="Marcel"]/cognoms
```

Que donarà de resultat:

```
1 <cognoms>Puig</cognoms>
```

De la mateixa manera que per obtenir-ne els valors, es poden fer comparacions amb els valors dels atributs especificant el seu nom rere el símbol @. Per saber si un element té l'atribut 'especialitat':

```
1 /classe/professor[@especialitat]
```

Retornarà:

```
1 <professor especialitat="507">
2   <nom>Marcel</nom>
3   <cognom>Puig</cognom>
4 </professor>
```

De la mateixa manera que amb els elements, es poden posar condicions als atributs per saber si el seu valor té un determinat valor, etc.

```
1 /classe/professor[@especialitat="507"]
```

```
1 /classe/professor[@especialitat]>=507
```

Es poden mesclar les expressions amb condicions sobre atributs i sobre elements per aconseguir expressions més complexes especificant-les una al costat de l'altra. Per exemple, podem obtenir el professor que té l'atribut especialitat a "507" i que es diu "Marcel" amb l'expressió:

```
1 /classe/professor[@especialitat="507"] [nom="Marcel"]
```

La funció **not()** es fa servir per negar les condicions:

```
1 /classe/professor[not(@especialitat)]
```

L'expressió pot ser tan complexa com calgui. Per exemple es pot obtenir la llista dels cognoms dels alumnes del professor de tipus "507" que es diu "Marcel":

```
1 /classe/professor[@Especialitat="507"] [nom="Marcel"]/..../alumnes/alumne/cognoms
```

Que retornarà els dos cognoms:

```
1 <cognoms>Pi</cognoms>
2 <cognoms>Garcia</cognoms>
```

## 2.4.4 Seqüències

Una seqüència és una expressió XPath que retorna més d'un element. S'assemblen bastant a les llistes d'altres llenguatges:

- Tenen ordre
- Permeten duplicats
- Poden contenir valors de tipus diferent en cada terme

És fàcil crear seqüències, ja que només cal tancar-les entre parèntesis i separar cada un dels termes amb comes. L'expressió següent aplicada a qualsevol document:

```
1 (1,2,3,4)
```

Retorna la seqüència de nombres d'un en un:

```
1
2
3
4
```

També es poden crear seqüències a partir d'expressions XPath. En aquest cas s'avaluarà primer la primera expressió, després la segona, etc.

```
1 (///nom/text(), //cognoms/text())
```

Aplicat al nostre exemple retornarà primer tots els noms i després tots els cognoms:

```
1 Marcel
2 Frederic
3 Filomeno
4 Puig
5 Pi
6 Garcia
```

## Unió, intersecció i disjunció

També es pot operar amb les seqüències d'elements. Una manera seria fer servir els operadors d'unió (**union**), intersecció (**intersec**) o disjunció (**except**).

Per exemple, l'expressió següent ens retornaria els cognoms dels alumnes que coincideixin amb els d'un professor:

```
1 (///alumne/nom) intersect (///professor/nom)
```

Amb la unió es poden unir les llistes de manera que en quedi una de sola sense duplicats:

```
1 (///alumne/nom) union (///professor/nom)
```

I amb la disjunció obtenim els noms de la primera seqüència que no surten en la segona:

```
1 (///alumne/nom) except (///professor/nom)
```

---

La creació dinàmica de seqüències funciona a partir d'XPath 2.0.

## 2.4.5 Funcions

XPath ofereix una gran quantitat de funcions destinades a manipular els resultats obtinguts.

Les funcions que ofereix XPath es classifiquen en diferents grups:

- **Per manipular conjunts de nodes:** es pot obtenir el nom dels nodes, treballar amb les posicions, comptar-los, etc.
- **Per treballar amb cadenes de caràcters:** permeten extreure caràcters, concatenar, comparar... les cadenes de caràcters.
- **Per fer operacions numèriques:** es poden convertir els resultats a valors numèrics, comptar els nodes, fer-hi operacions, etc.
- **Funcions booleans:** permeten fer operacions booleans amb els nodes.
- **Funcions per dates:** permeten fer operacions diverses amb dates i hores.

És impossible especificar-les totes aquí, de manera que el millor és consultar l'[especificació XPath](http://www.w3.org/TR/xpath-functions) (<http://www.w3.org/TR/xpath-functions>) per veure quines funcions es poden fer servir.

Entre totes les funcions podem destacar les que surten en la taula 2.4.

**TAULA 2.4.** Funcions XPath destacades

Funció	Ús
name()	Retorna el nom del node
sum()	Retorna la suma d'una seqüència de nodes o de valors
count()	Retorna el nombre de nodes que hi ha en una seqüència
avg()	Fa la mitjana dels valors de la seqüència
max()	Retorna el valor màxim de la seqüència
min()	Dóna el valor mínim de la seqüència
position()	Diu en quina posició es troba el node actual
last()	Retorna si el node actual és l'últim
distinct-values()	Retorna els elements de la seqüència sense duplicats
concat()	Uneix dues cadenes de caràcters
starts-with()	Retorna si la cadena comença amb els caràcters marcats
contains()	Ens diu si el resultat conté el valor
string-length()	Retorna la llargada de la cadena
substring()	Permet extreure una subcadena del resultat

**TAULA 2.4** (continuació)

Funció	Ús
string-join()	Uneix la seqüència amb el separador especificat
current-date()	Ens retornarà l'hora actual
not()	Inverteix els valors booleans

Amb les funcions es podran fer peticions que retornin valors numèrics. Per exemple, “quants alumnes tenim?”:

1 `count(/classe/alumnes/alumne)`

Que tornarà el nombre d'alumnes del fitxer:

1 2

O bé retornar cadenes de caràcters. Per exemple, unir tots els noms separant-los per una coma:

1 `string-join(//nom, ",")`

Que retornarà en un únic resultat els noms separats per una coma:

1 `Marcel, Frederic, Filomeno`

També es poden posar les funcions en els predicats. Per exemple, aquesta expressió ens retornarà els alumnes amb cognom que comenci per *p*.

1 `/classe/alumnes/alumne[starts-with(cognoms, "P")]`

En aquesta expressió volem obtenir el segon alumne de la llista:

1 `/classe/alumnes/alumne[position()=2]`

## 2.5 XSLT

XSLT (*extensible stylesheet language for transformations*) és un llenguatge de plantilles basat en XML que permet convertir l'estructura dels elements XML en altres documents.

Es tracta d'una recomanació del W3C (World Wide Web Consortium), com ho és CSS, però XSLT va molt més enllà, ja que supera moltes de les limitacions de CSS.

Amb XSLT es poden fer transformacions que canviïn totalment l'estructura del document, pot reordenar la informació del document XML, afegir-hi informació

nova on sigui, prendre decisions en funció de la informació que s'hi trobi, fer càlculs, etc.

XSLT es recolza en altres tecnologies XML per funcionar:

- Fa servir XPath per determinar les plantilles per aplicar en cada moment (i per tant s'integra en XQuery).
- Suporta XML Schemas per definir els tipus de dades.

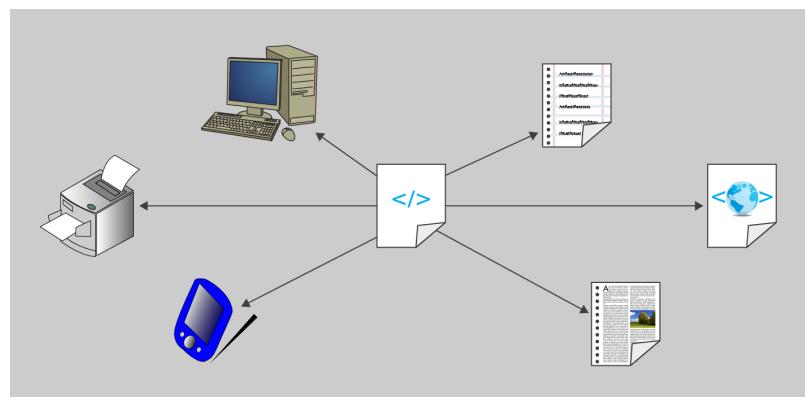
Tot i que hi ha la versió 2.0 d'XSLT, la que es fa servir més és la 1.0.

La versió 2.0 de XSLT afegeix tota una sèrie de característiques que encara fan més potent XSLT:

- Suporta els tipus de dades d'XML Schemas
- Inclou elements nous que permeten agrupar resultats, etc.
- Pot generar múltiples sortides en una sola transformació
- Afegeix un grup de funcions noves a més de les d'XPath 2.0
- Pot processar fitxers que no siguin XML

Cada vegada es fan servir més les plantilles XSLT per generar vistes personalitzades d'un document XML per ser visualitzades en diferents destinacions (impressora, pantalla d'ordinador, telèfon mòbil...). D'aquesta manera n'hi ha prou només tenint el document XML original, i els altres es crearan sota demanda (figura 2.14).

**FIGURA 2.14.** L'ús més corrent és transformar documents en funció de la destinació



Una de les limitacions d'XSLT és que el document d'entrada ha de ser sempre XML, a pesar que la sortida no té aquesta limitació i pot acabar essent en qualsevol format: text, HTML, XML, etc.

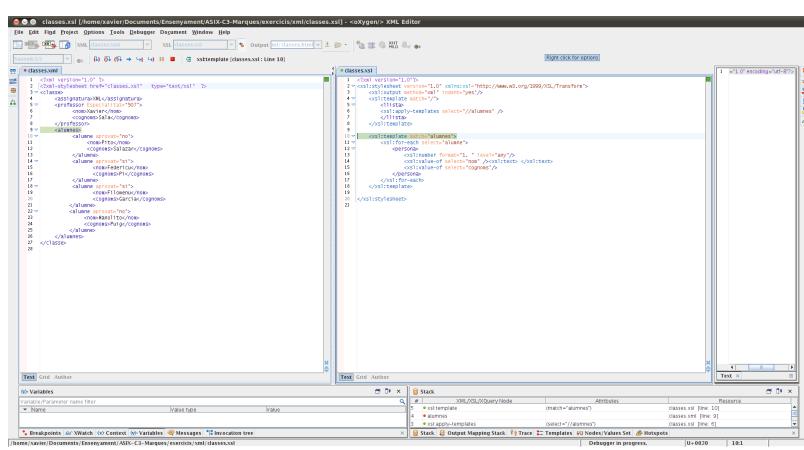
## 2.5.1 Programes

A pesar que les biblioteques XSLT tenen utilitats que es poden fer servir en temps de creació per provar que la plantilla fa el que ha de fer, normalment les transformacions es faran internament des de programes. És per aquest motiu que la majoria de les implementacions XSLT estan en forma de biblioteques.

Uns dels programes d'ús corrent que tenen incorporades biblioteques XSLT són els navegadors web (Firefox, Internet Explorer, Google Chrome, i d'altres). Si es carrega un document XML que té associada una transformació XSLT en un navegador web es transformarà automàticament i s'hi mostrerà el resultat format.

Les plantilles XSLT són documents XML que es poden crear fent servir editors de text o bé editors especialitzats en XML. En la creació de plantilles XSLT, els avantatges que ofereixen els editors XML són tan importants (depuradors d'expressions, ajudes...) que els fan una eina molt valiosa per treballar-hi professionalment (figura 2.15).

**FIGURA 2.15.** El depurador XSLT incorporat a l'oXygen XML Editor



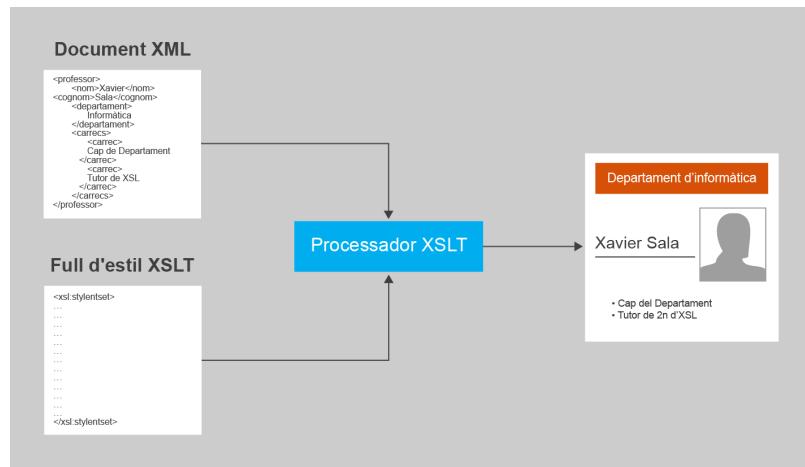
## 2.5.2 El procés de transformació

Una transformació XSLT consisteix a passar un document XML i una plantilla per un processador XSLT, el qual aplicarà les regles que trobi en la plantilla al document per generar un document nou (figura 2.16). El fitxer de resultat de la transformació pot ser tant un document XML com en qualsevol altre format.

Per associar un document XML a una plantilla XSLT específica es fa servir l'etiqueta <?xml-stylesheet ?>.

Per exemple si volem associar la plantilla XSLT *alumnes.xsl* a un document cal editar-lo i afegir-hi l'etiqueta <?xml-stylesheet ?> rere la declaració xml:

```
<?xml-stylesheet href="alumnes.xsl" type="text/xsl" ?>
```

**FIGURA 2.16.** Processar XSLT

### L'arrel del document XSLT

Les plantilles XSLT són documents XML, i per tant, han de complir les regles dels documents XML. Això vol dir que han de tenir un element arrel:

L'arrel de les plantilles XSLT és l'element `<stylesheet>`.

L'element arrel `<stylesheet>` ha de tenir obligatòriament dos atributs:

- L'atribut **version**, que especificarà quina és la versió d'XSLT que s'ha fet servir per crear la plantilla i que es farà servir per fer la transformació.
- L'espai de noms de XSLT, que normalment s'associa amb l'àlies 'xsl'.

Per exemple, per a la versió 2.0 d'XSLT la definició de l'arrel seria una cosa com aquesta:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3      xmlns:xs="http://www.w3.org/2001/XMLSchema"
4      version="2.0">
5
6  </xsl:stylesheet>

```

Dins de l'arrel es definiran les plantilles que determinaran quina és la transformació que es vol fer en el document original.

### Crear una plantilla

El procés de transformació es farà intentant aplicar alguna plantilla als nodes del document original.

Qualsevol element que no es pugui processar amb una plantilla es transformarà amb el funcionament per defecte:

- Si el node té contingut es retornarà el contingut del node
- Si el node no té contingut es retornarà sense escriure res

Per tant l'element bàsic per fer les transformacions són les plantilles. Les plantilles es defineixen amb l'element `<xsl:template>`.

L'element de plantilla pot tenir diversos atributs però el més corrent és `match`. L'atribut `match` serveix per determinar a quins nodes s'ha d'aplicar aquesta plantilla per mitjà d'una expressió XPath.

En el contingut de l'element de plantilla s'especificarà quina és la transformació que cal aplicar als elements. La transformació més senzilla consisteix a escriure un text literal. Per exemple, en la plantilla següent s'està definint que en arribar algun element `<nom>` es desi en el fitxer de destinació la lletra A:

```
1 <xsl:template match="nom">
2   A
3 </xsl:template>
```

Per tant si s'aplica la plantilla anterior a aquest document:

```
1 <persona>
2   <nom>Pere Garcia</nom>
3 </persona>
```

El processador anirà analitzant els nodes del document original i primer intentarà buscar una plantilla per a l'element `<persona>`, i com que no en trobarà cap recorrerà al comportament per defecte i escriurà una línia en blanc.

Després intentarà trobar una plantilla per a l'element `<nom>` que, segons la plantilla, s'ha de transformar en una lletra A.

Com que no hi ha més nodes el resultat serà aquest:

```
1 -
2   A
```

Cal tenir en compte que un cop un element ha estat processat per una plantilla se'n processa tot el seu contingut amb ell (i, per tant, també els elements que pugui contenir). Si canviem la plantilla anterior per la següent:

```
1 <xsl:template match="persona">
2   Persona
3 </xsl:template>
4
5 <xsl:template match="nom">
6   A
7 </xsl:template>
```

En aplicar-la al mateix XML primer processarà l'element `<persona>` i el transformarà en "Persona" en el document de sortida, però com que `<persona>` conté

tots els altres elements, la transformació s'acabarà.

```
1  Person
```

### Afegir elements

El fet de poder fer transformacions fent servir text literal fa que també es pugui fer servir el mateix sistema per crear etiquetes noves. Si es modifica la plantilla anterior per una com la següent:

```
1  <xsl:template match="nom">
2      <senyor>Manel</senyor>
3  </xsl:template>
```

El resultat serà que el transformarà en aquest:

```
1  <senyor>Manel</senyor>
```

S'ha d'anar amb compte quan s'especifiquen etiquetes literals, ja que no es poden crear etiquetes **que deixin la plantilla mal formada**. La plantilla següent que només obre l'etiqueta `<senyor>` no es pot fer servir, perquè deixa la plantilla mal formada:

```
1  <xsl:template match="nom">
2      <senyor>Manel
3  </xsl:template>
```

Una manera alternativa de crear elements en una transformació és fer servir `<xs:element>`. Entre els atributs que pot tenir, l'únic obligatori és **name**, que defineix el nom que tindrà l'etiqueta.

Per exemple, amb la plantilla següent creem un element `<senyor>` que sempre tindrà de contingut “Pere” per a cada element `<nom>` que es trobi en el document.

```
1  <xsl:template match="nom">
2      <xs:element name="senyor">Pere</xs:element>
3  </xsl:template>
```

O sigui, que la sortida de cada element `<nom>` serà:

```
1  <senyor>Pere</senyor>
```

### Afegir atributs

Els atributs es poden definir dins d'un nou element amb `<xsl:attribute>` i de la mateixa manera que amb els nous elements, l'únic atribut obligatori és **name**.

Una transformació que tingués això:

```
1  <xs:element name="persona">
2      <xsl:attribute name="home">Si</xsl:attribute>
3      Marcel
4  </xsl:element>
```

Generaria el següent:

```
1 <persona home="Si">Marcel</persona>
```

## Control de les sortides de text

En comptes de fer servir literals també es poden fer transformacions en text fent servir l'element `<xsl:text>`. Aquest element és important quan es volen controlar millor els espais i els salts de línia que hi haurà en les sortides.

Podem fer que surtin 5 espais darrere de la lletra *A* definint la plantilla de la manera següent:

```
1 <xsl:template match="nom">
2   <xsl:text>A      </xsl:text>
3 </xsl:template>
```

Si no ho féssim amb `<xsl:text>` els espais de darrere la *A* es perdrien.

## Obtenir valors

Una tasca habitual a l'hora de fer una transformació sol ser obtenir els valors dels elements del document d'origen per posar-los en l'element de destinació. Es poden obtenir el valor dels elements d'origen fent servir `<xsl:value-of>`.

Aquest element evalua l'expressió XPath de l'atribut **select** i en genera una sortida amb el seu contingut.

Per exemple, a partir de l'XML següent:

```
1 <xsl:template match="persona">
2   <xsl:value-of select="nom">
3 </xsl:template>
```

Afegirà el valor de l'element `<nom>` en el fitxer de sortida:

```
1 Pere Garcia
```

Però s'ha de tenir en compte que només evalua el primer element que compleix la condició, o sigui, que si el fitxer de mostra fos el següent:

```
1 <?xml version="1.0" ?>
2 <persona>
3   <nom tipus="Professor">Pere Garcia</nom>
4   <nom>Frederic Pi</nom>
5   <nom>Manel Puig</nom>
6 </persona>
```

...la sortida d'aplicar la mateixa plantilla només seria el primer dels noms i no tots dos, perquè agafaria només el primer `<nom>` de `<persona>`:

```
1 Pere Garcia
```

Per tant és molt important triar bé quines són les expressions XPath de les plantilles per evitar aquest comportament. En aquest cas la solució podria ser escollir <nom> en comptes de <persona>:

```

1  <xsl:template match="nom">
2      <xsl:value-of select=". "/>
3  </xsl:template>
```

Com a resultat de l'element <xsl:value-of> s'hi pot posar qualsevol expressió XPath. Per tant, també es poden obtenir els valors dels atributs afegint "@" davant del nom.

```

1  <xsl:template match="nom">
2      <xsl:value-of select=". "/> ( <xsl:value-of select="@tipus" /> )
3  </xsl:template>
```

Que generarà:

```

1  Pere Garcia ( Professor )
2  Frederic Pi ( )
3  Manel Puig ( )
```

I també es poden fer servir les funcions XPath. Per tant, es pot generar una sortida amb el total de les persones del fitxer:

```

1  <xsl:template match="/">
2      Total: <xsl:value-of select="count(//nom)"/>
3  </xsl:template>
```

Que donarà de resultat:

```
1  Total: 3
```

## Reenviar nodes a una altra plantilla

Un altre ús habitual de les plantilles és el de fer-les servir per cridar-ne d'altres quan es volen processar alguns dels nodes obtinguts per la plantilla quan aquesta n'obté diversos.

Les crides a plantilles es poden fer amb l'element <xsl:apply-templates>. Amb aquest element es pot fer que el diferents resultats d'una expressió XPath es vagin processant un a un, buscant una plantilla on aplicar-se.

Si partim d'aquest codi XML:

```

1  <classe>
2      <nom>Frederic Pi</nom>
3      <nom>Filomeno Garcia</nom>
4      <nom>Manel Puigdevall</nom>
5  </classe>
```

I el full de plantilles següent:

```

1 <xsl:template match="/">
2   <xsl:apply-templates select="classe/nom"/>
3 </xsl:template>
4
5 <xsl:template match="nom">
6   <xsl:value-of select=". "/>
7 </xsl:template>

```

Al executar-se l'*apply-templates* s'avalua l'expressió XPath que conté, *classe/nom*, que retorna tres resultats:

1. <nom>Frederic Pi</nom>
2. <nom>Filomeno Garcia</nom>
3. <nom>Manel Puigdevall</nom>

Cada un d'aquests resultats individualment intentarà trobar una plantilla on aplicar-se. De manera que es processarà tres vegades el segon *<xsl:template>*, ja que és qui capture *nom*, i donarà com a resultat:

```

1 Frederic Pi
2 Filomeno Garcia
3 Manel Puigdevall

```

Una de les característiques interessants és que *<apply-templates>* també es pot fer servir per reordenar el contingut. Per exemple, amb l'XML següent:

```

1 <classe>
2   <professors>
3     <nom>Marcel Puig</nom>
4     <nom>Maria Sabartés</nom>
5   </professors>
6   <alumnes>
7     <nom>Frederic Pi</nom>
8     <nom>Filomeno Garcia</nom>
9     <nom>Manel Puigdevall</nom>
10    </alumnes>
11 </classe>

```

Es poden obtenir tots els nodes amb una plantilla que capturi l'arrel i que primer mostri els noms dels alumnes i després els dels professors, de la següent manera:

```

1 <xsl:template match="classe">
2   Alumnes
3   _____
4   <xsl:apply-templates select="alumnes/nom"/>
5   Professors
6   _____
7   <xsl:apply-templates select="professors/nom"/>
8 </xsl:template>
9
10 <xsl:template match="nom">
11   <xsl:value-of select=". "/>
12 </xsl:template>

```

Que generarà:

```

1 Alumnes
2 _____
3 Frederic Pi
4 Filomeno Garcia
5 Manel Puigdevall
6
7 Professors
8 _____
9 Marcel Puig
10 Maria Sabartés

```

Es pot veure que un dels usos d'aquest element pot ser reordenar els resultats. En l'exemple, els alumnes en el document original estaven darrere dels professors, i en canvi en el que hem obtingut estan davant.

### Processar els nodes per fer tasques diferents

A vegades pot interessar processar els mateixos nodes diverses vegades per obtenir diferents resultats. Per poder fer això es poden definir les plantilles que tinguin l'atribut **mode**.

Si es defineix una plantilla amb l'atribut **mode** per activar la plantilla no n'hi haurà prou que quadri amb l'atribut **match**, sinó que també caldrà que s'hi passi l'atribut **mode**.

```

1 <template match="nom" mode="resultat">
2 ...
3 </template>

```

Les plantilles amb l'atribut **mode** han de ser cridades específicament. Per exemple, amb `<apply-templates>`:

```

1 <xsl:apply-templates select="nom" mode="resultat"/>

```

Això permet fer dues coses diferents amb l'element `<alumnes>` de l'exercici anterior: comptar els alumnes i fer-ne una llista.

```

1 <xsl:template match="classe">
2   <xsl:apply-templates select="//alumnes" mode="resultat"/>
3   <xsl:apply-templates select="//alumnes"/>
4 </xsl:template>
5
6 <xsl:template match="alumnes">
7   <xsl:apply-templates select="nom" />
8 </xsl:template>
9
10 <xsl:template match="alumnes" mode="resultat">
11   Total: <xsl:value-of select="count(nom)"/>
12 </xsl:template>
13
14 <xsl:template match="nom">
15   <xsl:value-of select=". "/>
16 </xsl:template>

```

El resultat serà:

```

1 Total: 3
2 Frederic Pi

```

3 Filomeno Garcia  
4 Manel Puigdevall

## Forçar el tipus de sortida

XSLT està pensat per generar sortides en text, XML, HTML i XHTML. Per defecte considera que la sortida serà un altre document XML i, per tant, si es vol generar un document XML no cal especificar res.

Podem forçar que la sortida sigui una altra amb `<xsl:output>`. Per exemple podem fer que la sortida sigui interpretada com a text pla amb:

1 `<xsl:output type="text">`

Molts processadors interpreten que la sortida serà HTML si la primera etiqueta que troben és `<html>`.

Si l'atribut `type` no és “xml” no sortirà la capçalera XML en el resultat final.

Però a part de `type` hi ha altres atributs que permeten controlar de quina manera es generaran els resultats. Per exemple, `indent` serveix per formatar les sortides, `encoding` per definir la codificació que cal fer servir, etc.

## Instruccions de control

XSLT proporciona una manera de processar els resultats molt semblant a com ho fan els llenguatges de programació. Com molts llenguatges de programació, incorpora instruccions per processar els resultats.

### Expressions condicionals

`<xsl:if>` permet afegir els resultats al fitxer només si es compleix una determinada condició.

Per exemple, davant d'un document com el següent:

1 `<alumnes>`  
2   `<alumne nota="5">Pere Garcia</nota>`  
3   `<alumne nota="3">Manel Puigdevall</nota>`  
4 `</alumnes>`

Amb `<xsl:if>` es pot aconseguir posar algun text només en els alumnes que tinguin una nota superior a 5.

1 `<xsl:if test="@nota>=5">`  
2   `(aprovat)`  
3 `</xsl>`

Hi ha altres funcions per expressar alternatives, com `<xsl:choose>`, que permet explicitar múltiples alternatives.

1 `<xsl:choose>`  
2   `<xsl:when select="@nota=10"> (·excellent) </xsl:when>`  
3   `<xsl:when select="@nota>=5"> (aprovat)     </xsl:when>`  
4   `<xsl:otherwise> (suspès) </xsl:`

```

5   otherwis
6   e>
7   </xsl:choose>
```

### Iteracions

**<xsl:for-each>** serveix per processar una seqüència de nodes un per un per fer alguna acció en cada un:

```

1 <xsl:for-each select="/classe/alumnes/nom">
2   <xsl:value-of select=". "/>
3 </xsl:for-each>
```

### Altres

Hi ha etiquetes que permeten fer tasques de transformació elaborades per poder ajustar les transformacions als diferents requisits que es puguin donar:

- Ordenar: **<xsl:sort>**.
- Numerar una llista de resultats: **<xsl:number>**.
- Crear variables: **<xsl:variable>**.
- Passar paràmetres a les plantilles: **<xsl:param>**, **<xsl:with-param>**, **<xsl:call-template>**.
- Copiar dades directament: **<xsl:copy>**, **<xsl:copy-of>**.
- Carregar XSL des d'altres arxius: **<xsl:import>** / **<xsl:include>**.
- Definir les nostres funcions: **<xsl:function>**.

En cas que calgui alguna tasca molt específica, és important tenir a mà l'especificació per comprovar si hi ha algun element que pugui simplificar el procés de creació de la plantilla:

- XSLT 1.0: <http://www.w3.org/TR/xslt>
- XSLT 2.0: <http://www.w3.org/TR/xslt20/>

A més, com que XSLT està basat en XML també s'hi poden afegir extensions per ampliar-ne les possibilitats a l'hora de fer transformacions. En aquest sentit són bastant populars les extensions que afegeix el grup **EXSTL** o les del processador **Saxon**.

### 2.5.3 Exemple

Per poder traspasar les dades d'un programa a un altre hem de convertir l'estructura d'un document XML en una altra de diferent.

En una editorial reben les comandes dels llibres en un format XML. El format en què es reben les comandes és un XML agrupat per autors com aquest:

```

1 <peticio biblioteca="Fantastica SL">
2   <data>15-11-2011</data>
3   <autor>
4     <nom>Frédérik McCloud</nom>
5     <llibres>
6       <llibre>
7         <titol>Les empentes</titol>
8         <quantitat>2</quantitat>
9       </llibre>
10      </llibres>
11    </autor>
12    <autor>
13      <nom>Corsaro Levy</nom>
14      <llibres>
15        <llibre>
16          <titol>Marxant de la font del gat</titol>
17          <quantitat>1</quantitat>
18        </llibre>
19        <llibre>
20          <titol>Bèstia!</titol>
21          <quantitat>1</quantitat>
22        </llibre>
23      </llibres>
24    </autor>
25 </peticio>
```

Però com que l'editorial té els llibres organitzats per títols i no per autors, volen que es converteixi en un format que els faci més còmode treballar-hi.

L'editorial vol que tingui una part en què s'identifiqui la data de la comanda i el nom de la llibreria i després que hi hagi la llista de llibres. L'equivalent de l'arxiu anterior seria aquest:

```

1 <comanda>
2   <llibreria data="15-11-2011">Fantastica SL</llibreria>
3   <llibre quantitat="2">
4     <nom>Les empentes</nom>
5     <autor>Frédérik McCloud</autor>
6   </llibre>
7   <llibre quantitat="1">
8     <nom>Marxant de la font del gat</nom>
9     <autor>Corsaro Levy</autor>
10  </llibre>
11  <llibre quantitat="1">
12    <nom>Bèstia!</nom>
13    <autor>Corsaro Levy</autor>
14  </llibre>
15 </comanda>
```

## Resolució

Es crea un arxiu amb la capçalera XML i s'hi defineix l'arrel dels fitxers XSLT.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3   version="2.0">
4
5 </xsl:stylesheet>
```

Com que es vol tenir totalment controlada tota la sortida el més fàcil és capturar tot el document amb una plantilla que capturi l'arrel.

```

1 <xsl:template match="/">
2
3 </xsl:template>
```

Dins d'aquesta plantilla ja es pot definir la primera part de l'estruatura del document, l'arrel `<comanda>` i l'element `<llibreria>`. En ser definits de nou no cal obtenir les dades del fitxer i es poden especificar com a literals.

```

1 <xsl:template match="/">
2   <comanda>
3     <llibreria>
4
5   </llibreria>
6
7   </comanda>
8 </xsl:template>
```

Dins d'aquesta plantilla caldrà aconseguir la llista de tots els llibres i les dades de la llibreria. Com que s'ha capturat l'arrel, les dades de l'etiqueta `<llibreria>` són fàcils d'aconseguir amb XPath:

- El nom de la llibreria està a `/peticio/@llibreria`.
- La data de la petició a `/peticio/data`.

Com que la data ha d'estar en un atribut, la posem dins d'un element `<xsl:attribute>` en el qual es defineix el nom dia:

```

1 <xsl:template match="/">
2   <comanda>
3     <llibreria>
4       <xsl:attribute name="dia">
5         <xsl:value-of select="/peticio/data"/>
6       </xsl:attribute>
7       <xsl:value-of select="/peticio/@llibreria"/>
8     </llibreria>
9   </comanda>
10 </xsl:template>
```

Si provem el codi que hem fet veurem que ja genera la primera part del que es vol obtenir. Apareix l'element llibreria i té els valors del document.

```

1 <comanda>
2   <llibreria data="15-11-2011">Fantastica SL</llibreria>
3 </comanda>
```

Només ens falta obtenir la llista de llibres. Com que es tracta d'una repetició de dades ho podem fer de dues maneres, amb un `for-each` o cridant una plantilla nova amb `apply-templates` que s'encarregui de fer cada llibre.

El més simple és fer-ho amb una plantilla a la qual anirem passant els nodes `<titol>` del document, i a partir d'aquests nodes obtindrem les dades.

Per tant darrere del tancament de l'element <llibreria> es fa una crida a una plantilla que capturarà tots els elements <titol> del document (en XPath ho podem expressar com ”//titol”):

```
1 <xsl:apply-templates select="//titol"/>
```

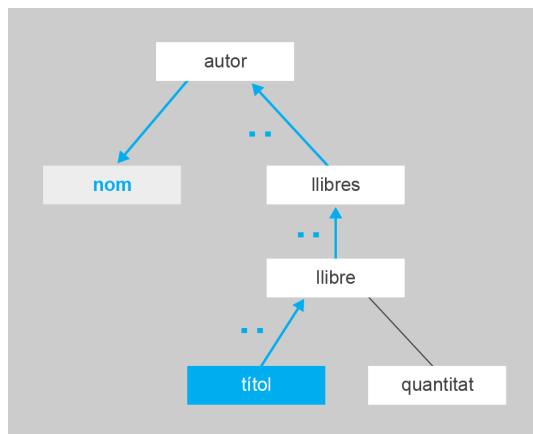
Falta definir la plantilla que rebrà la llista d'elements <titol>. Com que tots els títols estan dins d'una etiqueta <llibre> ja es pot definir:

```
1 <xsl:template match="titol">
2   <llibre>
3     </llibre>
4 </xsl:template>
```

Els valors que ens calen es poden aconseguir fàcilment:

- El títol és el valor del node en el qual estem. O sigui, en XPath seria .
- La quantitat de llibres demanats és el node germà del qual estem. Per tant, reculem fins al pare amb ... i després entrem en el node quantitat. L'expressió XPath és ../quantitat.
- El nom de l'autor es pot aconseguir amb una expressió una mica més llarga. Cal recular tres nivells fins a arribar al node <autor>, com es pot veure en la figura (figura 2.17). Per tant, l'expressió XPath serà ../../.../nom.

**FIGURA 2.17.** Per obtenir l'autor reculem tres nivells i entrem a "nom"



Només falta emplenar les dades dins de la nova plantilla: posar la quantitat com a atribut i crear les noves etiquetes per a l'autor i el títol, que podem definir com a literals.

```
1 <xsl:template match="titol">
2   <llibre>
3     <xsl:attribute name="quantitat">
4       <xsl:value-of select="../quantitat"/>
5     </xsl:attribute>
6
7     <nom><xsl:value-of select="."/></nom>
8     <autor><xsl:value-of select="../../.../nom"/></autor>
9   </llibre>
10 </xsl:template>
```

La plantilla final quedarà d'aquesta manera:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3      version="2.0">
4      <xsl:output indent="yes"/>
5      <xsl:template match="/">
6          <comanda>
7              <llibreria>
8                  <xsl:attribute name="data">
9                      <xsl:value-of select="peticio/data"/>
10                 </xsl:attribute>
11                 <xsl:value-of select="peticio@llibreria"/>
12             </llibreria>
13             <xsl:apply-templates select="//titol"/>
14         </comanda>
15     </xsl:template>
16
17     <xsl:template match="titol">
18         <llobre>
19             <xsl:attribute name="quantitat">
20                 <xsl:value-of select="..//quantitat"/>
21             </xsl:attribute>
22             <nom><xsl:value-of select=".//nom"/></nom>
23             <autor><xsl:value-of select="../../../../nom"/></autor>
24         </llobre>
25     </xsl:template>
26 </xsl:stylesheet>
```

### 3. Bases de dades natives XML. Llenguatge de consultes (XQuery)

Els fitxers en XML estan en un format de text estandarditzat que serveix per representar, emmagatzemar i transportar informació estructurada. Per tant l'**XML es pot fer servir com a mètode d'emmagatzematge de dades neutre**, ja que la informació emmagatzemada en fitxers XML es pot moure d'una màquina a una altra sense que hi hagi problemes. Això el fa un mètode ideal per emmagatzemar dades històriques que ens assegura que es podran tornar a llegir en el futur sense problemes.

Això ha fet que les organitzacions cada vegada tinguin una quantitat més gran de documents XML, i que en conseqüència localitzar els documents en el moment adequat sigui cada cop més complicat. Per tant **cal alguna manera d'organitzar les dades XML** que permeti localitzar-les ràpidament.

Però no solament n'hi ha prou de localitzar els documents, ja que sovint el que es busca estarà repartit en diferents documents. També cal alguna manera de poder **consultar i manipular les dades** que contenen els fitxers XML. Aquest sistema de consulta ha de ser capaç de fer cerques en els documents i modificar-ne dades si cal, però sempre mantenint uns **mínims d'eficiència**.

Davant d'aquestes necessitats la solució se sol donar de dues maneres:

1. Com que majoritàriament les organitzacions ja tenen sistemes d'emmagatzematge organitzats de dades basats en **dades relacionals** una possible solució podria ser **convertir els fitxers XML a dades relacionals**:

- L'emmagatzematge de dades serà totalment organitzat i centralitzat en un punt on ja hi ha les altres dades de l'organització.
- El seu llenguatge de consulta, SQL, és molt conegut i es fa servir molt, de manera que no caldrà que els usuaris aprenguin un nou llenguatge.

2. Una altra solució consisteix a **crear un sistema d'emmagatzematge pensat només per als fitxers XML**. Són els sistemes coneguts com a bases de dades natives XML. Aquests:

- Proporcionen un lloc per emmagatzemar ordenadament fitxers XML.
- Tenen el seu llenguatge de consulta propi: XQuery.

Tot i que en la teoria és fàcil fer aquesta distinció, en la pràctica molt sovint les organitzacions fan servir sistemes d'emmagatzematge mixtos, en els quals desen en dades relacionals aquelles dades que han de ser emmagatzemades o consultades, i en XML aquelles dades que han de ser representades en entorns web o que s'han d'intercanviar o transportar a altres sistemes.

L'objectiu d'aquest mòdul no és ser expert en bases de dades, per tant només

veurem alguns exemples de coses que es poden fer en base de dades per treballar amb XML

### 3.1 Bases de dades relacionals

Gairebé totes les organitzacions tenen les seves dades organitzades amb algun sistema relacional, ja que els sistemes gestors de bases de dades s'han convertit en un element indispensable en el funcionament de les empreses. Es fan servir per controlar moltes de les dades de funcionament de les empreses: facturació, comptabilitat, estocks...

Per tant, davant de l'aparició d'un nou tipus de dades, un dels raonaments fàcils seria: "si ja tenim un sistema d'organització de dades que funciona bé, per què no podem posar aquestes dades en el mateix sistema?". D'aquesta manera, les dades que s'obtinguessin dels fitxers XML aconseguirien un sistema molt eficient d'emmagatzematge i un mètode de manipulació d'informació que ha estat molt provat durant molts anys i que coneix molta gent.

La inclusió dels fitxers XML en els sistemes gestors de bases de dades relacionals es pot fer de dues maneres:

1. Convertir les dades dels fitxers XML en dades relacionals:

- Aquest sistema té l'avantatge que les dades, un cop dins del sistema relacional, seran idèntiques a les ja existents.
- L'inconvenient més important és que si torna a fer falta el document XML original pot ser molt difícil regenerar-lo, ja que sovint hi haurà informació sobre l'estructura XML que no s'emmagatzemarà.

2. Emmagatzemar els documents XML sencers en les bases de dades:

- Es posaran els fitxers XML sencers en un camp d'una taula de la base de dades, de manera que serà com les altres dades.
- Per poder-hi treballar bé caldrà que la base de dades ofereixi alguna manera mitjanament eficient de poder fer cerques en el contingut dels documents XML.

#### 3.1.1 Convertir les dades XML en relacionals

Una solució que pot semblar senzilla però que en realitat no n'és tant consisteix a agafar les dades dels fitxers XML i transformar-les en dades relacionals. Un cop s'hagin incorporat les dades a la base de dades es podran eliminar els fitxers XML.

Com que en el document XML ja hi ha definida l'estructura de les dades, i sovint els documents XML estaran associats a un vocabulari, tindrem que:

- Generar l'estructura de taules relacionals es pot fer analitzant l'estructura del document XML.
- Es poden obtenir els camps de la definició del vocabulari o simplement observant el contingut del document XML.

I a més hi ha sistemes que permeten transformar dades XML en altres tipus de fitxers (per exemple, **XSLT**).

Vegeu XSLT en l'apartat "Conversió i adaptació de documents XML" d'aquesta unitat.

#### Convertir un document XML en dades relacionals

Si es parteix del document XML següent:

```

1  <?xml version="1.0"?>
2  <alumnes>
3      <alumne>
4          <nom>Frederic</nom>
5          <cognom>Pi</cognom>
6      </alumne>
7      <alumne>
8          <nom>Filomeno</nom>
9          <cognom>Garcia</cognom>
10     </alumne>
11 </alumnes>
```

És relativament senzill veure que l'estructura del document consisteix en una llista d'elements `<alumne>`. I que els `<alumne>` tindran dos camps de dades que són `<nom>` i `<cognom>`.

Per tant, l'estructura relacional d'aquest fitxer serà simplement crear una taula 'alumne' que tingui com a dades un nom i un cognom. Això és trivial amb l'ordre CREATE TABLE d'SQL:

```
1  CREATE TABLE alumne (nom VARCHAR(30), cognom VARCHAR(30))
```

Obtenir les dades per emplenar la taula tampoc no ha de ser un problema gaire gran. Simplement cal fer INSERT de cada camp de dades:

```

1  INSERT INTO alumne VALUES("Frederic", "Pi");
2  INSERT INTO alumne VALUES("Filomeno", "Garcia");
```

Per no haver-ho de fer manualment el més fàcil seria crear una plantilla XSLT que faci la transformació del fitxer XML en les regles SQL.

A pesar de l'aparent senzillesa d'aquest sistema, no sempre és senzill fer aquesta conversió, ja que els sistemes relacionals i XML parteixen de conceptes bastant diferents:

- El sistema relacional està basat en dades bidimensionals sense jerarquia ni ordre, mentre que el sistema XML està basat en arbres jeràrquics en què l'ordre és significant.
- En un document XML hi pot haver dades repetides, mentre que els sistemes relacionals fugen de les repeticions.
- Les relacions i les estructures dins dels documents XML no sempre són òbviament evidents.
- Què passa si necessitem tenir el document XML de nou? Fer el procés invers no sempre és trivial. Un dels conceptes difícils és determinar quines dades eren atributs i quines elements.

Per tant, normalment aquest no és el sistema més aconsellable però és un sistema vàlid que pot ser útil en casos concrets.

### 3.1.2 Sistemes relacionals amb extensions XML

Fins a l'aparició d'XML pràcticament tothom emmagatzemava les dades que s'havien de consultar o modificar ràpidament per mitjà d'un sistema relacional. Els sistemes gestors de bases de dades han estat durant anys els reis de l'administració de dades en les organitzacions.

Però l'increment d'informació en XML que s'havia de poder consultar o que es requeria per fer tasques que abans estaven reservades a les bases de dades relacionals ha provocat que s'hagi hagut d'incloure algun tipus de suport XML en els sistemes gestors de bases de dades.

Tots els grans sistemes de bases de dades importants com Oracle, IBM DB2 o Microsoft SQL Server tenen algun tipus de suport per a XML, que normalment es concreta en el següent:

- Permetre exportar les dades relacionals en algun format XML per transportar les dades.
- Tenir alguna manera de poder emmagatzemar documents XML com a camps en taules relacionals.
- Permetre fer cerques i canvis en els documents XML emmagatzemats.
- Generar XML a partir de les dades relacionals de la base de dades.

A pesar que SQL/XML forma part de l'estàndard SQL, no tots els gestors de bases de dades el suporten.

Com que SQL (el llenguatge de consulta de dades relacional per excel·lència) no tenia suport per a XML el 2003, es va modificar l'estàndard del llenguatge SQL per afegir l'**extensió SQL/XML**.

#### SQL/XML

SQL/XML és una extensió de l'estàndard SQL que permet treballar amb el llenguatge XML per mitjà d'instruccions SQL.

A pesar de participar en l'elaboració de l'estàndard, Microsoft va anunciar que no tenia intenció d'incorporar SQL/XML en el Microsoft SQL Server. En comptes d'això, el Microsoft SQL Server fa servir un sistema propi anomenat Microsoft SQLXML o OPENXML per treballar amb SQL i XML.

Aquesta extensió va ser desenvolupada per un grup en el qual hi havia les grans empreses de bases de dades (Microsoft, Oracle, IBM, SyBase, DataDirect Technologies...) i ja està implementada en algun sistema de bases de dades.

SQL/XML defineix tota una sèrie de funcions per a publicació de fitxers XML a partir de dades relacionals, defineix un tipus de dades XML i una manera de consultar i manipular les dades XML emmagatzemades.

## SQL/XML a Oracle

Oracle és un dels sistemes gestors de bases de dades més importants del mercat i, per tant, no podia no donar suport als documents XML. Per fer-ho defineix el tipus de dades XMLType, que serveix per emmagatzemar XML com si es tractés d'un tipus de dades més en les taules.

Per tant, fent servir XMLType es pot crear una taula amb un camp amb documents XML i treballar-hi com si fossin dades relacionals normals.

```
1 CREATE TABLE alumnes(ID INT NOT NULL, document XMLTYPE);
```

En la figura 3.1 podem veure la descripció d'una taula SQL que conté XML en Oracle.

**FIGURA 3.1.** Descripció de la taula amb un camp XMLType

The screenshot shows the Oracle Database Express Edition interface. In the main window, the SQL command `DESCRIBE ALUMNES;` is entered. Below the command, the results are displayed in a table:

Tipo de Objeto TABLE Objeto ALUMNES									
Table	Column	Tipo De Dato	Longitud	Precisión	Escala	Clave Primaria	Nulo	Valor Por Defecto	Comentario
ALUMNES	ID	Number	-	-	0	-	✓	-	
	DOCUMENT	Xmtype	2000	-	-	-	✓	-	

At the bottom of the results, it says "1 - 2".

XMLType permet emmagatzemar els documents XML amb la definició de l'esquema o sense.

1. Amb l'esquema XSD el sistema pot saber de quin tipus són les dades de cada element XML, i per tant, internament pot organitzar les dades de manera que sigui possible fer-hi cerques més eficients. A més, en conèixer el contingut de les etiquetes es podran fer operacions amb aquestes de la mateixa manera que es fa amb les altres dades
2. Si no es proporciona l'esquema els XML seran tractats com un camp de text i les cerques seran menys eficients.

L'entrada de dades XML a una taula Oracle es pot fer amb la instrucció habitual però especificant el camp XML amb el tipus XMLTYPE.

```
1 INSERT INTO alumnes VALUES (1,
2   XMLTYPE('<persona><nom>Pere</nom></persona>'));
```

També es poden consultar els valors XML de la manera habitual (figura 3.2).

**FIGURA 3.2.** Consulta dels valors de la base de dades

The screenshot shows the Oracle Database Express Edition interface. The title bar says 'Comandos SQL' and 'ORACLE Database Express Edition'. The user is 'XAVIER'. The menu bar includes 'Inicio', 'Desconectar', and 'Ayuda'. The main area shows a SQL command: 'SELECT \* FROM ALUMNES;'. Below it, the results are displayed in a table:

ID	DOCUMENT
1	<alumne><nom>Pere</nom></alumne>
3	<alumne><nom>Manel</nom></alumne>
2	<alumne><nom>Frederic</nom></alumne>

3 filas devueltas en 0,00 segundos [Exportación de CSV](#)

Per fer cerques Oracle proporciona una sèrie de funcions que permeten que a un camp XML se li puguin extreure dades fent servir XPath, comprovar l'existència de determinats nodes, crear-hi nodes nous, etc...

Per exemple, es poden extreure tots els elements <nom> d'un camp XML d'una taula fent servir una ordre com la següent (figura 3.3):

```
1 SELECT EXTRACT(document,'//nom/text()') "noms"
2 FROM alumnes;
```

**FIGURA 3.3.** Consulta SQL amb XPath

The screenshot shows the Oracle Database Express Edition interface. The title bar says 'Comandos SQL' and 'ORACLE Database Express Edition'. The user is 'XAVIER'. The menu bar includes 'Inicio', 'Desconectar', and 'Ayuda'. The main area shows a SQL command using the EXTRACT function with an XPath expression:

```
1 SELECT EXTRACT(document,'//nom/text()') "noms"
2 FROM alumnes;
```

The results are displayed in a table:

Noms
Pere
Manel
Frederic

3 filas devueltas en 0,10 segundos [Exportación de CSV](#)

També pot caldre generar documents XML a partir de les dades que hi ha emmagatzemades en la base de dades. Oracle ho pot fer de diverses maneres:

- Per mitjà d'**SQL/XML**.
- Per mitjà del paquet **DBMS\_XMLGEN**, que permet crear XML a partir de consultes.
- Amb la funció **SYS\_XMLGEN**, que crea un document XML per a cada registre resultat d'una consulta.
- Fent servir la utilitat **XSU**, que és específica per desenvolupar en Java.

Per exemple, fent servir l'estàndard SQL/XML disposem d'una sèrie de funcions per generar contingut XML, especificant-les en la consulta SQL (taula 3.1).

**TAULA 3.1.** Algunes funcions SQL/XML

Funció	Ús
XMLELEMENT	Permet crear un element amb un nom determinat
XMLATTRIBUTES	Es fa servir per crear atributs dins d'un element
XMLCONCAT	Concatena múltiples etiquetes
XMLFOREST	Crea elements XML a partir d'un grup de resultats

Amb aquestes funcions es pot generar XML a partir una taula SQL que contingui els camps *nom* i *cognom* com la (figura 3.4).

**FIGURA 3.4.** Taula relacional 'Professors'

ID	NOM	COGNOM
1	Marcel	Puig
2	Antoni	Boix

Amb la instrucció següent:

```
1 SELECT xmlelement('alumne', xmlforest(a.nom, a.cognom))
2 FROM alumnes;
```

La figura 3.5 mostra quin és el resultat d'executar-la en l'Oracle.

**FIGURA 3.5.** Generar XML a partir de dades relacionals

The screenshot shows the Oracle Database Express Edition interface. The SQL command entered is:

```
SELECT XMLEMENT("professor", XMLFOREST(a.nom, a.cognom))
FROM professors a
```

The results pane displays the generated XML output:

```
XMLEMENT("PROFESSOR", XMLFOREST(A.NOM, A.COGNOM))
<professor><NOM>Marcel</NOM><COGNOM>Puig</COGNOM></professor>
<professor><NOM>Antoni</NOM><COGNOM>Boix</COGNOM></professor>
```

Below the results, it says "2 filas devueltas en 0,05 segundos" and "Exportación de CSV".

## 3.2 XQuery

Una de les necessitats més bàsiques a l'hora de poder fer cerques en les dades és disposar d'un llenguatge per fer consultes que sigui prou potent per poder cobrir les necessitats dels que l'han de fer servir. Per aquest motiu es va desenvolupar el llenguatge XQuery.

XQuery és un llenguatge de consultes pensat per convertir-se en la manera estàndard de recuperar dades de col·leccions de documents XML.

Es tracta d'un llenguatge funcional, de manera que en comptes de dir-li quines són les passes per fer una tasca el que es fa és avaluar les expressions contra el fitxer XML i generar un resultat. A diferència dels llenguatges de programació habituals, en XQuery s'especifica què és el que es vol i no la manera com ho ha de fer per obtenir-ho.

### Xquery 3.0

Ja està en desenvolupament una nova versió d'XQuery que permetrà incrementar la potència de les consultes tot afegint clàusules noves com `group by` o el control d'errors dinàmics. La podeu consultar a <http://www.w3.org/TR/xquery-30/>

Entre les característiques més interessants d'XQuery, aquest permet:

- Seleccionar la informació segons criteris. Ordenar, agrupar, afegir dades.
- Filtrar la informació que es vulgui del flux de dades.
- Cercar informació en un document o en un grup de documents.
- Unir dades de múltiples documents.
- Transformar i reestructurar XML.
- No estar limitat a la cerca, ja que pot fer operacions numèriques i de manipulació de caràcters.
- Pot treballar amb espais de noms i amb documents definits per mitjà de DTD o XSD.

Una part important d'XQuery 1.0 és el llenguatge XPath 2.0, que és la part que li permet fer les seleccions d'informació i la navegació pel document.

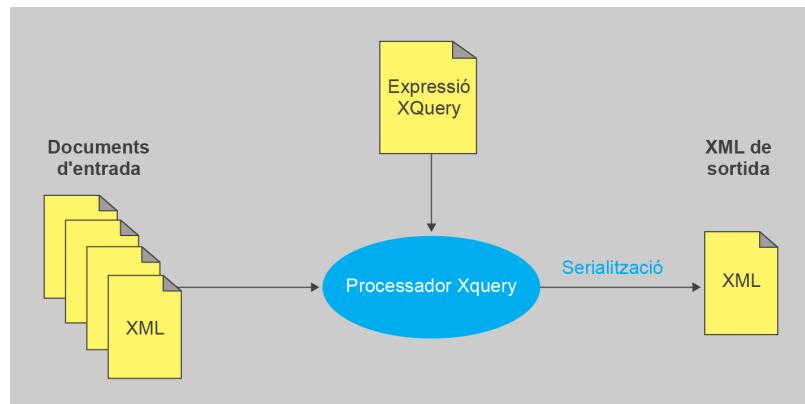
### 3.2.1 Processadors XQuery

L'avaluació d'expressions XQuery requerirà disposar d'algun processador XQuery.

Els processadors XQuery agafen els documents XML d'entrada i els apliquen una transformació per generar un resultat.

En la figura 3.6 es pot veure quin és el funcionament d'un processador XQuery.

**FIGURA 3.6.** Funcionament dels processadors XQuery



Normalment els processadors XQuery venen en forma de biblioteques que es poden incorporar als programes. Molts sistemes de base de dades o programes incorporen aquestes biblioteques per poder donar suport XQuery en els seus productes.

## Saxon

Probablement una de les biblioteques més usades i amb millor suport per a XQuery és **Saxon**.

En la seva versió gratuïta (Saxon Home Edition) a part de les biblioteques s'instal·len dos executables que serveixen per fer transformacions i per avaluar expressions XQuery: Transform i Query.

Podem executar XQuery simplement executant l'executable amb el fitxer amb les expressions:

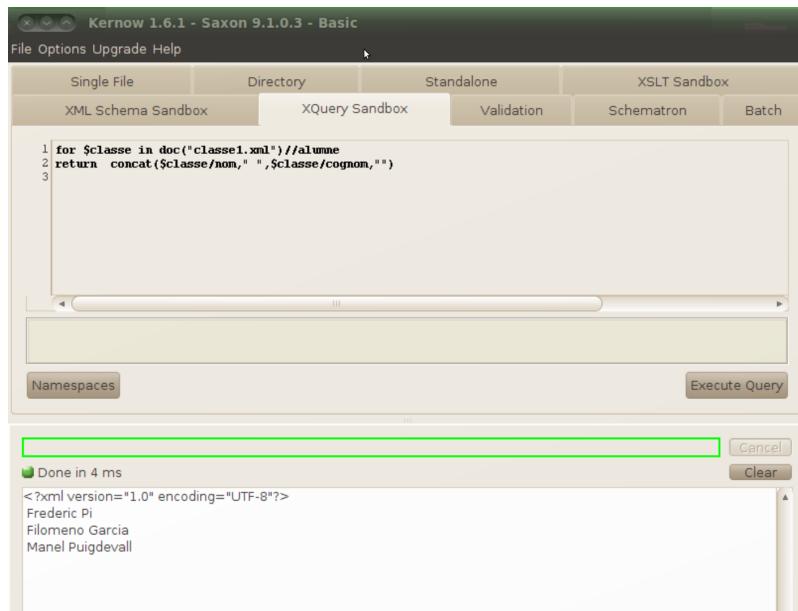
```
1 C:\> Query consulta.xquery
```

Normalment la majoria de distribucions Linux tenen la versió lliure de Saxon en el seu repositori d'aplicacions. Per exemple, per al sistema Ubuntu, es troba en la biblioteca **libsaxonb-java** i la instrucció és **saxonb-xquery**.

```
1 $ saxonb-xquery consulta.xquery
```

## Kernow

**FIGURA 3.7.** Avaluació amb el Kernow



El Kernow és un programa molt popular que fa servir Saxon i que permet fer evaluacions XQuery des d'un entorn gràfic per mitjà del que anomena *XQuery Sandbox* (figura 3.7). Es pot baixar des de <http://kernowforsaxon.sourceforge.net/>.

A part d'avaluar expressions XQuery aquest programa també ofereix altres possibilitats de la biblioteca Saxon, com fer transformacions XSLT, etc.

## XQilla

L'XQilla és un programa que corre sobre la biblioteca Xerces i que permet fer evaluacions Xquery des de l'entorn d'instruccions.

```
1 $ xqilla consulta.xquery
```

## Altova

Altova és l'empresa que desenvolupa un editor XML anomenat Altova XMLSpy, que és bastant popular en entorns Windows. Entre les eines gratuïtes que ofereix hi ha unes utilitats per processar XQuery i XSLT des de consola.

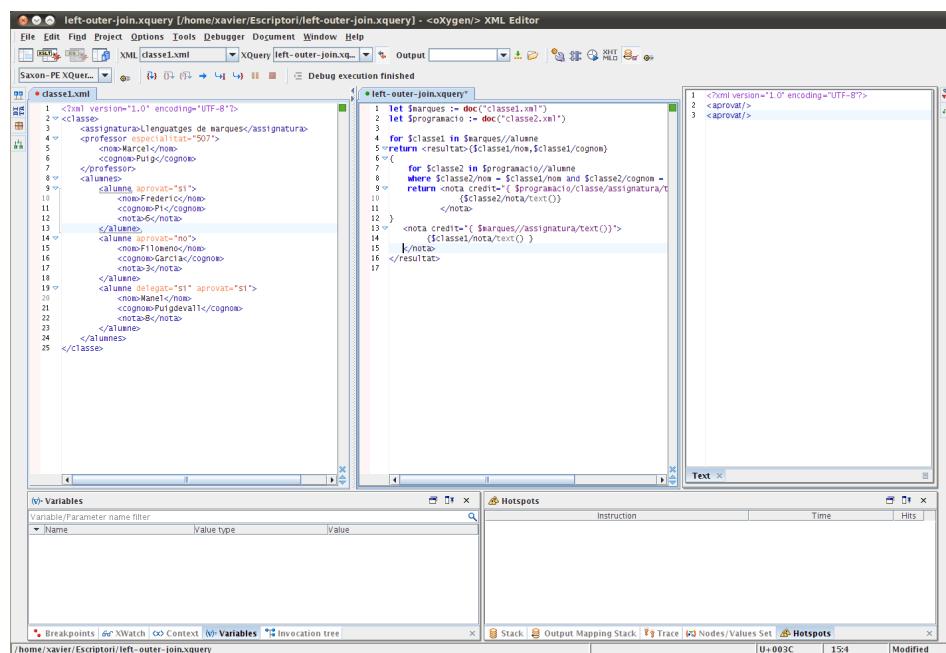
Es pot processar una petició XQuery amb:

```
1 C:\> altovaxml /xquery consulta.xquery
```

## Editors XML

Molts dels editors XML estan enllaçats amb biblioteques per processar XQuery i des de l'entorn mateix de l'editor permeten editar, provar i depurar les peticions XQuery (figura 3.8).

**FIGURA 3.8.** Prova d'edicions XQuery en l'oXxygen XML Editor



A més d'altres avantatges com els assistents d'ajuda, hi ha la possibilitat de fer consultes connectant amb bases de dades XML locals o remotes, etc.

## Bases de dades XML

Lògicament, com que les bases de dades natives fan servir XQuery com a base en les seves consultes, resulten ideals per provar expressions XQuery.

És corrent que aquestes tinguin alguna consola o un entorn en el qual es podran executar les ordres i veure'n el resultat.

### 3.2.2 Consultes XQuery

Generalment una consulta XQuery consistirà a aconseguir les dades amb les quals es vol treballar, filtrar-les i retornar el resultat com a XML. L'expressió més simple que es pot fer en XQuery és escriure un element buit:

```
1 <Hola />
```

En executar aquesta ordre el resultat serà el següent:

```
1 <?xml version="1.0" ?>
2 <Hola/>
```

Aquesta és una de les característiques importants d'XQuery: que escriu literalment el text; mentre que si en el resultat s'hi defineixen etiquetes l'expressió només serà correcta si el resultat final és ben format. Per exemple, executar el següent en XQuery donarà un error, ja que el resultat final no és ben format:

```
1 <a>
```

Però a pesar de poder escriure text, el més corrent és recuperar algun tipus de dades, avaluar-les i retornar el resultat amb un `return`. La instrucció `return` s'encarrega de generar les sortides.

Una altra característica important és que **fa servir variables**. Per exemple la següent és una expressió XQuery correcta que assigna el valor 5 a `$a` i després en retorna el valor:

```
1 let $a:=5
2 return 5
```

El resultat d'executar-ho amb XQuery serà:

```
1 <?xml version="1.0" ?>
2 5
```

A diferència del que passa amb els literals, el `return` s'executa una vegada per cada valor que rebi, de manera que si hi hagués una expressió que es fes més d'un cop:

```
1 for $a in (1,2)
2 return <Hola/>
```

El `return` s'executarà més d'un cop:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Hola/>
3 <Hola/>
```

### 3.2.3 Comentaris

Com en gairebé tots els llenguatges, en XQuery també es permeten els comentaris. Com que XQuery no té sintaxi XML els comentaris no es fan de la mateixa manera que en XML sinó que es defineix el seu sistema propi.

Els comentaris es defineixen posant-los entre el símbol "(:" i el ":)".

```
1 (: Comentari :)
```

També es poden fer comentaris multilínia

```
1 (: 
2 Això també
3 és un comentari
4 :)
```

Tot el que estigui dins d'un comentari serà ignorat pel processador.

### 3.2.4 Càrrega de documents

Una de les coses que caldrà per interrogar un fitxer o fitxers XML és establir específicament a quins documents es vol fer la petició. La càrrega dels fitxers la podem fer amb dues funcions que podem veure a la taula 3.2.

**TAULA 3.2.** Funcions per carregar fitxers XML

Funció	Ús
<code>doc()</code>	Dir qui és el document que volem consultar.
<code>collection()</code>	Especificar un grup de documents alhora. Generalment serà un document amb enllaços a d'altres o una adreça d'una base de dades XML.

Per tant, podem fer servir la funció `doc()` per carregar un document XML. Amb aquesta instrucció XQuery tornarà tot el document *alumnes.xml*.

```
1 return doc("alumnes.xml")
```

XQuery conté completament XPath, de manera que es pot fer servir qualsevol expressió XPath per filtrar resultats quan faci falta, i per tant es pot fer que en

en el mateix procés de càrrega s'hi apliqui un filtre. Per exemple una expressió XQuery que retorna quins són els noms dels alumnes del fitxer *alumnes.xml* pot ser la següent:

```
1 return doc("alumnes.xml")//nom
```

Els documents poden estar en qualsevol lloc al qual es pugui arribar. En aquest cas, per exemple, l'expressió XQuery pot ser una URL:

```
1 return doc("http://ioc.xtec.cat/alumnes.xml")
```

### 3.2.5 Variables

XQuery és un llenguatge pensat per fer cerca en documents XML, però no solament es pot fer servir per fer cerca, ja que té suport per fer operacions aritmètiques i per treballar amb cadenes de caràcters.

En aquest exemple fem servir XQuery per fer una operació matemàtica senzilla:

```
1 let $x := 5
2 let $y := 4
3 return $x + $y
```

El resultat serà:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 9
```

Una característica d'XQuery que el diferencia d'altres llenguatges de cerca és que **disposa de variables**:

- Les variables en XQuery s'identifiquen perquè comencen sempre amb el símbol \$.
- Poden contenir qualsevol valor: literals numèrics o caràcters, seqüències de nodes...
- La instrucció per assignar valors a una variable és `let`.

Un dels usos fonamentals de les variables és emmagatzemar elements per poder-los fer servir posteriorment. En l'exemple següent s'emmagatzemen els elements `<nom>` en la variable `$alumnes` i després en retorna la quantitat d'alumnes:

```
1 let $alumnes := doc("alumnes.xml")//alumnes/nom
2 return count($alumnes)
```

Es poden aplicar filters XPath a les variables:

```
1 let $tot := doc("classe1.xml")
2 let $profe := $tot//professor
3 let $alum := $tot//alumne
```

Els valors de les variables els podrem comparar amb els operadors de comparació habituals que podem veure a la taula 3.3.

**TAULA 3.3.** Operadors de comparació

Operador	Operador2	Ús
=	eq	Dos valors són iguals
!=	ne	Dos valors són diferents
>	gt	És més gran que l'altre
>=	ge	És més gran o igual
<	lt	Més petit
<=	le	Més petit o igual

Els operadors eq, ne, gt, ge, lt i le només es poden fer servir per comparar valors individuals, i per tant només es podran fer servir si hi ha un esquema definit.

### 3.2.6 Expressions evaluables

XQuery està pensat per poder mesclar les consultes amb qualsevol altre tipus de contingut. Si es mescla amb contingut, les expressions que s'hagin d'avaluar s'han de col·locar entre claus "`{...}`".

Per exemple, podem mesclar HTML i XQuery per tal que el processador XQuery generi una pàgina web amb les dades d'un document XML.

```

1 <html>
2   <head><title>Llista de classe</title></head>
3   <body>
4     <h1>
5     {
6       doc("classe1.xml")//assignatura
7     }
8     </h1>
9     </body>
10    </html>

```

En mesclar contingut, el processador XQuery només avaluarà les instruccions que estiguin dins de les claus, i per tant deixarà les etiquetes HTML tal com estan.

### Creació d'elements i atributs

Fent servir les expressions evaluables és fàcil crear nous elements.

```

1 <modul>
2   { doc("classes.xml")//modul/text() }
3 </modul>

```

També es poden crear atributs (s'ha d'anar en compte de no deixar-se les cometes al voltant del valor que obtindran els atributs).

```
1 <modul nom="{doc("classes.xml")//modul/text()}" />
```

Una manera alternativa i més potent de definir elements i atributs és fer servir les paraules clau `element` i `attribute`. Aquestes instruccions permeten crear elements i definir-ne el contingut especificant-lo dins de les claus.

```
1 element modul {  
2 }
```

Crearà un element `<modul>` buit:

```
1 <modul />
```

Si volem crear nous elements o atributs dins d'un element definit d'aquesta manera s'han d'especificar dins de les claus. Per exemple, el codi següent:

```
1 element modul {  
2   attribute nom {"Llenguatges de marques"},  
3   element alumnes { }  
4 }
```

Generarà el següent:

```
1 <element nom="Llenguatges de marques">  
2   <alumnes />  
3 </element>
```

No hi ha cap limitació a l'hora d'imbricar els elements i atributs per generar resultats tan complexos com calgui, i en qualsevol moment s'hi pot posar una expressió XQuery.

```
1 element modul {  
2   attribute nom { doc("classe.xml")//modul/text() }  
3 }
```

### 3.2.7 Expressions FLWOR

Les expressions FLWOR són la part més potent d'XQuery. Estan formades per cinc instruccions opcionals que permeten fer les consultes i alhora processar-les per seleccionar els ítems que interessin d'una seqüència (taula 3.4).

**TAULA 3.4.** Expressions FLWOR

Caràcter	Comanda	Ús
f	for	Permet passar d'un element a un altre de la seqüència
l	let	Serveix per assignar valors a una variable
w	where	Permet filtrar els resultats segons condicions
.	.	.

**TAULA 3.4** (continuació)

Caràcter	Comanda	Ús
o	order by	Usat per ordenar els resultats abans de mostrar-los
r	return	Retorna el resultat de tota l'expressió

**"for ... return"**

La instrucció **for** es fa servir per avaluar individualment cada un dels resultats d'una seqüència de nodes. En un **for** es defineixen dues coses:

- Una variable, que serà la que anirà agafant els resultats de la seqüència un per un.
- La paraula clau **in**, en la qual es defineix quina és la seqüència d'elements per processar.

La manera més senzilla d'expressió FLWOR és combinar el **for** amb el **return** per retornar els valors obtinguts de manera seqüencial:

```
1 for $i in ('Hola', 'Adéu')
2 return $i
```

Generarà:

```
1 Hola Adéu
```

Les seqüències de valors poden ser de qualsevol tipus. Per exemple, poden ser nombres:

```
1 for $i in (1,2,3)
2 return $i*$i
```

Que retorna:

```
1 1 4 9
```

O bé una seqüència de nodes. Per exemple amb l'expressió següent capturem una seqüència de nodes **<alumne>** i els fem servir per obtenir-ne el cognom:

```
1 for $alumne in doc("classe.xml")//alumne
2 return <alumne> { $alumne/cognom } </alumne>
```

El resultat serà quelcom així:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <alumne>
3   <cognom>Pi</cognom>
4 </alumne>
5 <alumne>
6   <cognom>Garcia</cognom>
7 </alumne>
```

S'hi pot veure que els valors del `return` s'han anat processant un per un i per aquest motiu es repeteixen les etiquetes `<alumne>`.

Un aspecte important és que no cal que el `for` sigui la primera expressió de la consulta. També es pot posar com a paràmetre en una funció XPath.

Per exemple, l'expressió següent ens tornarà el nombre d'alumnes:

```

1 count (
2   for $alumne in doc("alumnes.xml")//nom
3     return $alumne
4 )
```

### "for ... at"

En el `for` es pot incloure l'operador `at`, que permet obtenir la posició del node que s'està processant. Amb aquest operador es pot fer que surti el número d'ordre en els resultats.

```

1 for $alumne at $pos in doc("classe.xml")//alumne
2   return <alumne numero="{{$pos}}>
3     { $alumne/cognom/text() }
4   </alumne>
```

Que donarà:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <alumne numero="1">Pi</alumne>
3 <alumne numero="2">Garcia</alumne>
4 <alumne numero="3">Puigdevall</alumne>
```

### "for" amb múltiples variables

Fins ara només s'ha fet servir una variable en el `for` però se n'hi poden posar tantes com faci falta separant-les per comes:

```

1 <resultats>
2 {
3   for $tot in doc("classe.xml")/classe,
4     $nom in $tot/alumnes,
5     $modul in $tot/assignatura/text()
6   return
7     <modul nom="{{$modul}}"
8       alumnes="{{$count($nom/alumne)}}>
9       { $nom/alumne/nom }
10    </modul>
11  }
12 </resultats>
```

Que donarà el resultat següent:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <resultats>
3   <modul alumnes="3" nom="Llenguatges de marques">
4     <nom>Frederic</nom>
5     <nom>Filomeno</nom>
6     <nom>Manel</nom>
7     </modul>
8   </resultats>
```

### "for" niuat

Imbricant les instruccions `for` entre elles es poden aconseguir el que SQL anomena ***outer joins***. O sigui que es retoraran sempre tots els resultats de la primera consulta i, només en el cas que hi hagi alguna relació, s'uniran amb els de la segona consulta.

```

1 for $selec in doc("classe.xml")//alumne
2 return
3 <persona>
4   { $selec/nom }
5   {
6     for $selec2 in $selec
7       return $selec2//cognom
8   }
9 </persona>
```

### let

`let` permet declarar variables i assignar-los un valor. Sobretot es fa servir per emmagatzemar valors que s'han de fer servir més tard.

```

1 let $num := 1
2 let $nom := "manel"
3 let $i := (1 to 3)
```

En les expressions FLWOR hi pot haver tants `let` com calgui.

```

1 for $alumne in doc("notes.xml")//alumne
2 let $nom := $alumne/nom
3 let $nota := $alumne/nota
4 return <nom>concat($nom,":",$nota)
```

S'ha d'anar amb compte amb `let` perquè el seu funcionament és molt diferent del de `for`:

- `for` s'executa per a cada membre d'una seqüència.
- `let` fa referència al seu valor en conjunt. Si és una seqüència el que es processa són tots els valors de cop.

Podem veure la diferència amb un exemple. Aquesta instrucció amb `for`:

```

1 for $selec in doc("classes.xml")//alumne
2 return <persona>{$selec/nom}</persona>
```

Dóna:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <persona><nom>Frederic</nom></persona>
3 <persona><nom>Filomeno</nom></persona>
4 <persona><nom>Manel</nom></persona>
```

I en canvi amb `let`:

```

1 let $selec := doc("classes.xml")//alumne
2 return <persona>{$selec/nom}</persona>
```

Donarà:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <persona>
3   <nom>Frederic</nom>
4   <nom>Filomeno</nom>
5   <nom>Manel</nom>
6 </persona>
```

Els resultats són bastant diferents! Es pot veure clarament que `let` ha tractat tots els valors de cop.

## Condicions

La instrucció `where` és la part que indicarà quin filtre s'ha de posar a les dades rebudes abans d'enviar-les a la sortida del `return`. Normalment en el filtre es fa servir algun tipus de predicat XPath:

```

1 for $alumne in doc("classe.xml")//alumne
2 where $alumne/@aprovat="si"
3 return $alumne/nom
```

Com que XPath forma part d'XQuery moltes vegades el mateix filtre es pot definir de diverses maneres. Per exemple, aquesta expressió és equivalent a l'anterior:

```

1 for $alumne in doc("classe.xml")//alumne[@aprovat="si"]
2 return $alumne/nom
```

En un `where` hi pot haver tantes condicions com calgui simplement encadenant-les amb les operacions lògiques `and`, `or` o `not()`.

Aquesta expressió retorna el cognom dels alumnes que han aprovat i que es diuen *Pere*:

```

1 for $alumne in doc("classe.xml")//alumne
2 where $alumne/@aprovat="si" and nom="Pere"
3 return $alumne/cognom
```

El `where` afegeix més potència del que sembla, ja que ens permet fer els `inner joins` d'SQL en fitxers XML. Per exemple, a partir de dos fitxers amb les dades de dues classes diferents es poden obtenir només els alumnes que es repeteixen entre les dues classes amb:

```

1 for $alum1 in doc("classe1.xml")//alumne
2 let $alum2 in doc("classe2.xml")//alumne
3 where $alum1 = $alum2
4 return $alum1
```

## Quantificadors existencials

A vegades cal comprovar si algun o tots els elements d'una seqüència compleixen una determinada condició, i per això s'han definit les expressions quantificades. Amb les expressions quantificades es poden fer comprovacions sense haver de recórrer a comptadors (taula 3.5).

**TAULA 3.5.**

Quantificador	Serveix per
<code>every ... satisfies</code>	Comprovar que tots els elements compleixin una condició determinada.
<code>some ... satisfies</code>	Comprovar que algun dels elements de la llista compleixin la condició.

A aquestes expressions els passem una variable que capturarà el valor concret respecte al qual es vol fer la comprovació. Per exemple, amb aquesta expressió només donarà el nom de la classe si tots els alumnes de la seqüència tenen de cognom *Pi*:

```

1  for $s in doc("classe.xml")//alumnes
2  where every $e in $s//cognom satisfies ($e="Pi")
3  return $s/modul

```

Canviant `every` per `some` retornarà el nom de la classe si algun dels alumnes es diu de cognom *Pi*:

```

1  for $s in doc("classe.xml")//alumnes
2  where every $e in $s//cognom satisfies ($e="Pi")
3  return $s/modul

```

Aquestes expressions per si soles no serveixen per filtrar contingut, ja que sempre retornen cert o fals. Per aquest motiu sempre se solen fer servir dins d'una clàusula `where`.

## Ordenació

Una de les operacions habituals en les cerques és representar els resultats en un ordre determinat. Aquesta és la funció que fa *order by*.

Es pot ordenar de manera ascendent amb `ascending` (que és el que fa per defecte) o bé descendent amb `descending`.

```

1  for $alumne in doc("classe.xml")//alumne
2  order by $alumne/cognom
3  return $alumne

```

També es poden especificar diferents conceptes en l'ordenació. En aquest exemple els resultats s'ordenaran primer per cognom i després per nom:

```

1  for $alumne in doc("classe.xml")//alumne
2  order by $alumne/cognom, $alumne/nom
3  return $alumne

```

En aquest altre el cognom s'ordena de manera descendent i el nom de manera ascendente.

```

1  for $alumne in doc("classe.xml")//alumne
2  order by $alumne/cognom descending, $alumne/nom ascending
3  return $alumne

```

Per defecte ordena com si el contingut de les etiquetes fos text. Si cal obtenir una ordenació numèrica caldrà convertir els valors a nombres amb la funció d'XPath `number()`.

```

1 for $alumne in doc("classe.xml")//alumne
2 order by number($numero)
3 return $alumne

```

### 3.2.8 Alternatives

La instrucció `if` ens permetrà fer una cosa o una altra segons si es compleix la condició especificada o no. Per exemple, podem fer una llista d'alumnes en què s'especifiqui si han aprovat o no a partir del valor que hi ha en l'element `<nota>`.

```

1 let $doc := doc("classe1.xml")
2 let $aprovat := 5
3 for $b in $doc//alumne
4 return
5 if ($b/nota >= 5) then
6 <data>
7 { $b/nom/text() } està aprovat
8 </data>
9 else
10 <data>
11 { $b/nom/text() } no està aprovat
12 </data>

```

Això generarà el resultat següent:

```

1 <data>Frederic està aprovat</data>
2 <data>Filomeno no està aprovat</data>
3 <data>Manel està aprovat</data>

```

La instrucció `else` és habitual en llenguatges de programació, però a diferència del que passa normalment, en XQuery, encara que no es vulgui fer res, en cas que la condició falli s'hi ha d'especificar l'`else`. Si no es vol que faci res es deixa en blanc.

```

1 for $classe in doc("classe1.xml")//alumne
2 return if ($classe/nota > 5) then
3     <aprovat/>
4     else ()

```

### 3.2.9 El pròleg XQuery

A pesar que en moltes expressions XQuery no s'especifica cap pròleg, les expressions XQuery estan formades per dues parts:

- **Pròleg:** és una part opcional i serveix per donar informació al processador per saber com s'ha d'avaluar l'expressió. Exemples típics són declarar

espais de noms, variables o funcions, importar esquemes, o definir com s'han de tractar els espais en blanc.

- **Cos:** és on hi ha el que es vol fer amb l'expressió.

Un valor que se sol especificar en el pròleg és la versió d'XQuery que s'ha de fer servir en la consulta. És opcional, però si s'hi posa ha de ser la primera línia del fitxer.

```
1 xquery version "1.0";
```

També s'hi poden definir altres coses com informació sobre la consulta o dades que posteriorment es podran fer servir en el cos de l'expressió.

En aquest exemple, en la capçalera es passa informació al processador sobre com ha de tractar els espais, s'hi declara una variable \$alum que contindrà una part del document XML i es defineix un espai de noms `ioc`. Les variables i l'espai de noms es poden fer servir en l'expressió.

```
1 xquery version "1.0";
2 declare boundary-space preserve;
3 declare namespace ioc="http://ioc.xtec.cat/alum";
4 declare variable $alum := doc("alumnes.xml")//alumne;
5
6 <ioc:llista>
7 {
8     for $nom in $alum/nom
9     return <alumne>{$nom}</alumne>
10 }
11 </ioc:llista>
```

En executar la consulta el processador afegirà automàticament l'espai de noms al resultat de l'expressió en la definició de `<llista>`:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ioc:llista xmlns:ioc="http://ioc.xtec.cat/alum">
3     <ioc:alumne>
4         <nom>Frederic</nom>
5     </ioc:alumne>
6     <ioc:alumne>
7         <nom>Filomeno</nom>
8     </ioc:alumne>
9     <ioc:alumne>
10        <nom>Manel</nom>
11    </ioc:alumne>
12 </ioc:llista>
```

Un dels usos interessants del pròleg és que permet passar paràmetres a les expressions. Si es declara una variable com `external`, per poder executar la consulta s'haurà de proporcionar un valor a la variable:

```
1 declare variable $entrada as xs:string external;
2
3 <resultat>
4 { $entrada }
5 </resultat>
```

Per executar la consulta anterior amb la utilitat Query de Saxon s'haurà d'especificar el valor de la variable `entrada` de la manera següent:

```
1 C:\> Query prova.xquery entrada=Hola  
2 <?xml version="1.0" encoding="UTF-8"?><resultat>Hola</resultat>
```

Un altre ús interessant del pròleg és que permet definir funcions:

```
1 declare function quadrat($numero as xs:integer) return xs:integer  
2 {  
3     return ($numero*$numero)  
4 };
```

Aquestes es poden cridar de la mateixa manera que les funcions normals d'XPath:

```
1 quadrat($alumne/nota)
```

### 3.2.10 Actualitzacions de dades

Un llenguatge que pretengui ser un estàndard de consulta ha de poder gestionar completament les dades, i per tant necessita alguna manera de fer consultes però també canviar, afegir o esborrar dades. XQuery 1.0 és bàsicament un llenguatge de consultes, i tot i que la majoria de les operacions que es fan en els sistemes de base de dades són consultes, necessita alguna manera de poder actualitzar les dades.

Per aquest motiu en el W3C s'ha posat en marxa la creació d'un sistema d'actualitzacions anomenat **xupdate (XQuery Update Facility 1.0)**, que ha de permetre a XQuery fer les actualitzacions. Alhora també hi ha altres iniciatives per crear un estàndard per actualitzar valors en sistemes XML com el que promociona el grup XMLDB, que s'anomena **XUpdate (XML update language)**.

La falta d'existència d'un estàndard clar fa que els programes que implementen XQuery, si volen oferir la possibilitat de fer actualitzacions, no sempre tinguin clar quin sistema d'actualitzacions han de fer servir i es decantin pel que els agrada més, per incloure'n diversos o fins i tot per crear el seu propi sistema.

#### Fer servir XQuery per respondre preguntes sobre documents XML

Partint d'aquest XML d'exemple que representa comandes de llibres:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <petició llibreria="Fantastica SL">
3      <data>15-11-2011</data>
4      <autor>
5          <nom>Frédéric McCloud</nom>
6          <llibres>
7              <llibre>
8                  <titol>Les empentes</titol>
9                  <quantitat>2</quantitat>
10             </llibre>
11         </llibres>
12     </autor>
13     <autor>
14         <nom>Corsaro Levy</nom>
15         <llibres>
16             <llibre>
17                 <titol>Marxonat de la font del gat</titol>
18                 <quantitat>2</quantitat>
19             </llibre>
20             <llibre>
21                 <titol>Bèstia!</titol>
22                 <quantitat>2</quantitat>
23             </llibre>
24         </llibres>
25     </autor>
26     <autor>
27         <nom>Marc Blairet</nom>
28         <llibres>
29             <llibre>
30                 <titol>Tres de tres</titol>
31                 <quantitat>3</quantitat>
32             </llibre>
33         </llibres>
34     </autor>
35 </petició>
```

Es vol respondre la pregunta següent: **de quins autors s'han demanat tres o més llibres?**

Per respondre aquesta pregunta el primer que cal és separar el document en blocs d'autor i processar-ne cada un de manera diferent. Per tant, es defineix un bucle amb un `for` per cada autor i s'escriu el nom per pantalla

```

1  for $autor in doc("llibres.xml")//autor
2  return $autor/nom
```

Si es passa l'expressió per un processador XQuery el resultat serà una llista de tots els autors:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <nom>Frédéric McCloud</nom>
3  <nom>Corsaro Levy</nom>
4  <nom>Marc Blairet</nom>
```

Es vol definir una restricció amb la quantitat de llibres que s'han demanat de cada autor; per tant, ho podríem intentar fer amb el `where`

```

1  for $autor in doc("llibres.xml")//autor
2  where $autor/llibres/llibre/quantitat >= 3
3  return $autor/nom
```

Aquesta és la llista dels autors que tenen alguna petició de 3 o més llibres

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <nom>Marc Blairet</nom>
```

Però es pot veure que encara no és el demanat, ja que no surt l'autor Corsaro Levy, del qual

s'han demanat dues unitats llibres diferents. Per tant, agrupem les quantitats per obtenir el resultat correcte

```
1 for $autor in doc("llibres.xml")//autor
2 where sum($autor/llibres/livre/quantitat) >= 3
3 return $autor/nom
```

Ara el resultat sí que és el correcte:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <nom>Corsaro Levy</nom>
3 <nom>Marc Blairet</nom>
```

Podem fer que el resultat sigui més “bonic” mostrant les quantitats venudes al costat de cada autor i ordenant de més a menys els resultats.

```
1 for $autor in doc("llibres.xml")//autor
2 let $suma := sum($autor/llibres/livre/quantitat)
3 where $suma >=3
4 order by $suma descending
5 return concat($autor/nom, " : ", $suma)
```

Es pot veure que per no haver d'escriure diverses vegades l'expressió de suma s'ha definit una variable `$suma` i d'aquesta manera l'expressió ha quedat més fàcil d'escriure

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 Corsaro Levy : 4
3 Marc Blairet : 3
```

La funció `concat()` ha eliminat les etiquetes XML i ha generat un resultat de text. Si es volgués mantenir una estructura XML es pot crear manualment amb l'ajuda de les expressions evaluables:

```
1 for $autor in doc("llibres.xml")//autor
2 let $suma := sum($autor/llibres/livre/quantitat)
3 where $suma >=3
4 order by $suma descending
5 return <autor> $1 concat($autor/nom, " : ", $suma) $1 </autor>
```

Que generarà:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <resultat>Corsaro Levy : 4</resultat>
3 <resultat>Marc Blairet : 3</resultat>
```

### 3.3 Bases de dades natives XML

Si no entrem en detalls tècnics, la definició d'una base de dades és un lloc en el qual es poden emmagatzemar dades. En el cas d'una base de dades en XML, consistiria simplement en emmagatzemar els fitxers XML en un punt concret del sistema operatiu.

L'increment de documents XML a emmagatzemar ha fet que, malgrat que l'emmagatzematge es pugui fer manualment, sigui interessant disposar d'alguna manera d'automatitzar el procés. Per facilitar aquesta automatització han aparegut les bases de dades natives XML (NXD).

Quan es parla de bases de dades natives XML es fa referència a bases de dades dissenyades per contenir i emmagatzemar dades en format XML.

A diferència del que passa amb les bases de dades relacionals, que fa molts anys que funcionen i tenen darrere seu una base teòrica important, les NXD no tenen uns estàndards definits per fer les coses i la teoria que les suporta no està gaire definida. Això fa que sovint cada base de dades faci les coses d'una manera que pot ser totalment diferent de com ho fa una altra.

Les bases de dades natives XML es fan servir sobretot per emmagatzemar dades que contenen:

- Contingut narratiu.
- Que són menys previsibles que les que s'emmagatzemen normalment en bases de dades relacionals.
- Que han de generar sortides per a entorns web.
- Que s'han de transportar d'un sistema a un altre.

### 3.3.1 Característiques més importants

Els sistemes NXD són molt heterogenis i sovint s'estan desenvolupant en direccions que no sempre coincideixen. Tot i la dispersió, sempre podem trobar punts en comú en la majoria de les bases de dades.

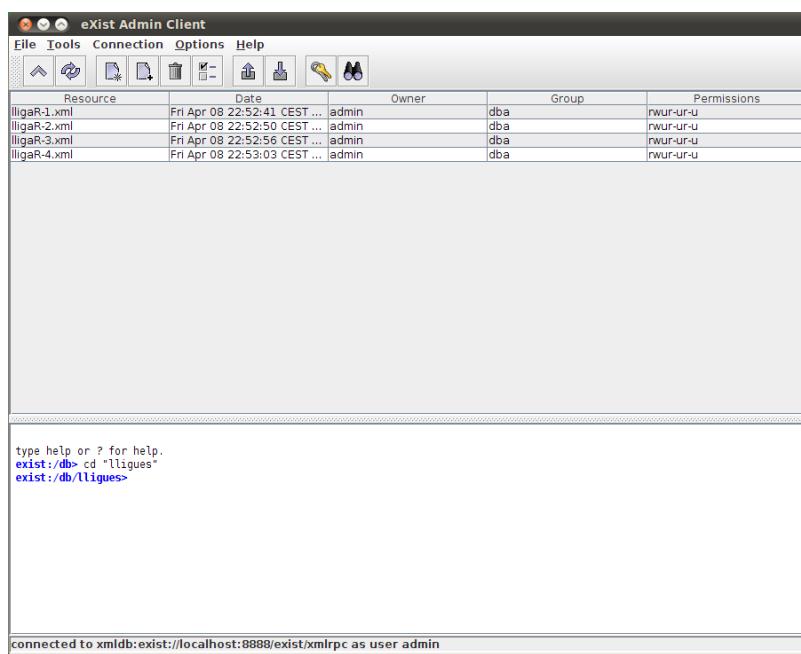
#### Llenguatge de consulta

Tots els sistemes NXD suporten com a mínim XPath com a llenguatge de consultes, tot i que la tendència general és adoptar XQuery; és lògic si es té en compte que XPath no va ser pensat per ser un llenguatge de consulta de base de dades, mentre que XQuery sí.

Tots els sistemes suporten algun tipus d'interfície que permet accedir-hi per mitjà d'algún llenguatge de programació, com ara la interfície XML:DB, o bé per mitjà d'algún protocol de xarxa com HTTP, WebDAV, SOAP...

#### Organització i emmagatzematge

L'organització més habitual de les dades és mitjançant col·leccions de documents XML que sovint intenten generar una estructura semblant a la que ofereixen els directoris dels sistemes operatius. En aquests sistemes una col·lecció és com una carpeta que conté un conjunt de documents XML o fins i tot altres carpetes (figura 3.9).

**FIGURA 3.9.** Les col·leccions en eXist són carpetes que contenen fitxers

El tractament de les col·leccions sol ser una de les diferències que hi ha entre bases de dades XML. Algunes迫en que dins d'una col·lecció hi hagi documents del mateix tipus, mentre que altres permeten que s'hi posi qualsevol tipus de document.

L'emmagatzematge de les dades sol ser el que diferencia més les NXD, ja que la naturalesa dels fitxers XML no els fa ideals per ser emmagatzemats de manera eficient (tenen repeticions, ocupen molt d'espai...). Per aquest motiu totes les NXD intenten optimitzar d'alguna manera aquest emmagatzematge: convertint les dades en fitxers binaris (amb simples compressions de dades o codificant-les), creant estructures d'arbre, o simplement emmagatzemant els fitxers XML sense fer-hi res confiant que la reducció de costos de l'emmagatzematge farà que ocupar espai no sigui un problema.

## Indexació

A l'hora de fer servir una base de dades el que importa realment a l'usuari és que faci la cerca en un temps acceptable. Els fitxers de text no són la millor manera d'emmagatzemar dades si l'objectiu és fer-hi cerques, i per tant les bases de dades XML s'han esforçat a intentar crear algun sistema per fer aquestes cerques més eficients.

El més habitual sol ser:

- Organitzar de manera eficient les col·leccions
- Fer servir índexos

Com passa en el món real, pot ser més senzill trobar el que busquem si tenim les dades ben organitzades i ben classificades que si les tenim de manera caòtica. Per

---

Si som prou organitzats a l'hora de desar els documents XML podria ser que ni ens fes falta tenir una base de dades XML.

tal per buscar les dades només caldrà accedir a la col·lecció adequada i des d'allà fer la cerca.

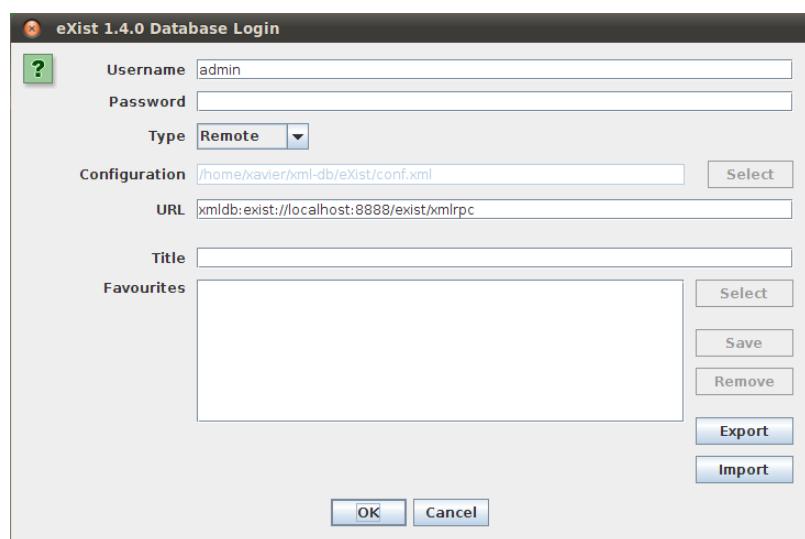
Un altre punt de vista consisteix a indexar les dades d'alguna manera per poder aconseguir recórrer la informació en un ordre determinat que permeti accedir a les dades d'una manera més ràpida. A diferència del que passa amb les bases de dades relacionals, en les d'XML no hi ha desenvolupades gaire teories sobre com s'han de crear els índexs, i això fa que els diferents productes facin servir filosofies d'indexació totalment diferents.

A pesar d'això el mètode d'indexació més corrent és crear una estructura d'arbre paral·lela als documents XML que contingui les dades que cal que siguin indexades.

## Seguretat

La seguretat és un altre aspecte per tenir en compte en els sistemes NXD. Generalment en un entorn empresarial no tothom pot accedir a les dades que vulgui sinó que a la informació poden accedir només les persones autoritzades (figura 3.10).

**FIGURA 3.10.** eXist demana autorització per permetre la connexió



La identificació d'usuaris permet implementar els mecanismes bàsics per permetre controlar a quins documents pot accedir cada un dels usuaris de l'organització (figura 3.11).

**FIGURA 3.11.** Permisos de fitxers

The screenshot shows the eXist Admin Client interface. At the top is a menu bar with File, Tools, Connection, Options, Help, and a toolbar with various icons. Below is a table listing file permissions:

Resource	Date	Owner	Group	Permissions
comics		admin	dba	rwur-ur-u
library		admin	dba	rwur-ur-u
lligues		admin	dba	rwur-ur-u
mods		admin	dba	rwur-ur-u
shakespeare		admin	dba	rwur-ur-u
system		admin	dba	rwnwv---
todo		admin	dba	rwur-ur-u
twitter		guest	guest	rwur-ur-u
www		admin	dba	rwur-ur-u
xinclude		admin	dba	rwur-ur-u
xproc		admin	dba	rwur-ur-u
xqdocs		admin	dba	rwurvur-
build.xml	Sat Dec 11 00:44:32 CET ...	admin	dba	rwur-ur-u
examples.xml	Sat Dec 11 00:44:32 CET ...	admin	dba	rwur-ur-u

At the bottom left, there's a command line interface with the prompt "exist:/db>". The status bar at the bottom right says "connected to xmldb:exist://localhost:8888/exist/xmlrpc as user admin".

## Actualitzacions de les dades XML

La falta d'un sistema d'actualitzacions dels fitxers XML en la primera versió d'XQuery ha fet que molt sovint els sistemes NXD, en comptes d'actualitzar les dades que han canviat, optin per substituir totalment el document cada vegada que s'hi produueixin canvis.

Per aquest motiu se solen oferir sistemes de transferència de fitxers via HTTP com WebDAV (figura 3.12) (*Web-based distributed authoring and versioning*) o per mitjà de serveis web (SOAP o REST).

**FIGURA 3.12.** Connexió amb eXist per mitjà de WebDAV

The screenshot shows the "db - Navegador de fitxers" (File Navigator) interface. At the top is a menu bar with Fitxer, Edita, Visualitza, Vés, Adreces d'interès, Ajuda, and a toolbar with icons. The address bar shows "Ubicació: dav://localhost:8080/exist/webdav/db". The left sidebar shows a tree view of local drives and network locations, including "WebDAV a loc...", "Papeleria", "Documents", "Música", "Imatges", "Vídeos", "Baixades", and "Oficial". The main pane displays a file system structure with four items: "lligues", "system", "xqdocs", and "classe1.xml". A status bar at the bottom right says "4 elements".

A pesar d'això la tendència és que els sistemes s'acullen a alguna de les propostes d'estàndards d'actualització de dades XML com **xquupdate** o **XML:DB XUpdate**.

### 3.3.2 Exemples de bases de dades natives XML

Hi ha moltes bases de dades XML que s'estan fent servir actualment en entorns professionals:

- **Tamino**: permet tenir un sistema híbrid d'emmagatzematge XML i relacional.
- **eXist-db**: es tracta d'un sistema de codi obert que fa servir una indexació de dades amb BTree en fitxers dividits en pàgines.
- **TigerLogic XDMS**: es pot fer servir tant per emmagatzemar documents XML com dades multimèdia.
- **EMC xdb**: una base de dades per a documents XML molt escalable.
- **Apache Xindice**: es tracta d'un sistema pensat per emmagatzemar grans col·leccions de petits fitxers XML. És de codi obert.
- **MarkLogic**: ofereix una solució empresarial per emmagatzemar qualsevol tipus de documents, metadades, RSS, correus electrònics, XML...

Generalment totes ofereixen:

- Emmagatzematge eficient de documents XML
- Suport per a consultes XQuery o XPath
- Alguna forma orientada a recursos d'accés a les dades: HTTP, webDAV, serveis web...
- Transformació de documents
- Diverses interfícies d'usuari

### 3.3.3 eXist-db

De les bases de dades XML de codi obert disponibles, probablement eXist-db és la que està més preparada per ser usada en un entorn professional. Es tracta d'una base de dades nativa XML que està desenvolupada en llenguatge Java, i que:

- Permet fer operacions amb els llenguatges XQuery 1.0, XPath 2.0 i XSLT 2.0.
- Pot funcionar tan com una biblioteca que s'incorpora als programes com ser executada per mitjà d'un servidor.

- Es pot accedir a la base de dades per mitjà de múltiples interfícies estàndards com REST, WebDAV, SOAP, XML-RPC o ATOM.
- Permet actualitzacions de dades per mitjà de l'API XMLDB, XUpdate i XQuery Update Facility.
- Porta incorporat un sistema de gestió d'usuaris de la base de dades i dels permisos d'accés a les col·leccions semblant al dels sistemes Unix (figura 3.13).

**FIGURA 3.13.** Cada fitxer té permisos d'usuari i de grup

The screenshot shows the eXist Admin Client interface. At the top is a menu bar with File, Tools, Connection, Options, Help, and several icons. Below the menu is a toolbar with icons for creating, deleting, and managing collections. The main area contains a table with columns: Resource, Date, Owner, Group, and Permissions. The table lists various XML files and their metadata. At the bottom of the interface, there is a command-line prompt window with the text "type help or ? for help." and "exist:> /db> |". A status bar at the bottom right indicates the connection details: "connected to xmldb:exist://localhost:8888/exist/xmlrpc as user admin".

Resource	Date	Owner	Group	Permissions
comics		admin	dba	rwur-ur-u
library		admin	dba	rwur-ur-u
lligues		admin	dba	rwur-ur-u
mods		admin	dba	rwur-ur-u
shakespeare		admin	dba	rwur-ur-u
system		admin	dba	rwurwru---
todo		admin	dba	rwur-ur-u
twitter		guest	guest	rwur-ur-u
www		admin	dba	rwur-ur-u
xinclude		admin	dba	rwur-ur-u
xproc		admin	dba	rwur-ur-u
xqdocs		admin	dba	rwurwur-
build.xml	Sat Dec 11 00:44:32 CET ...	admin	dba	rwur-ur-u
examples.xml	Sat Dec 11 00:44:32 CET ...	admin	dba	rwur-ur-u

L'organització de les dades és per mitjà de col·leccions de fitxers XML que s'organitzen d'una manera pràcticament idèntica a com es fa en els fitxers d'un sistema operatiu. Igual que en els sistemes operatius, es poden fer col·leccions dins de col·leccions i s'hi pot navegar amb un sol clic (figura 3.14).

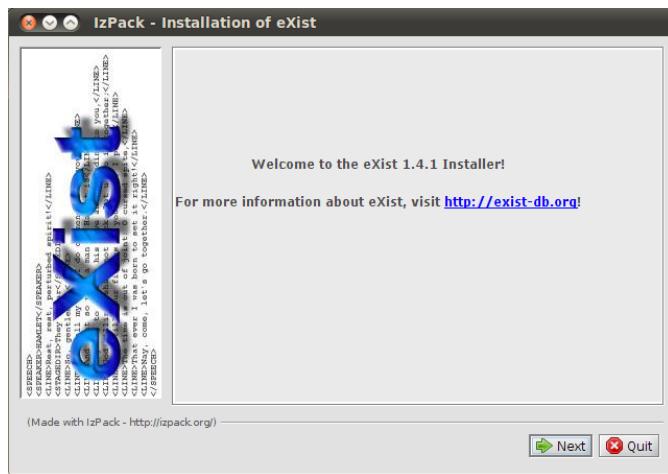
**FIGURA 3.14.** L'organització es fa en col·leccions i són navegables

The screenshot shows the eXist browser interface. On the left is a sidebar titled "Select a Page" with a tree view of the database structure. The main area is titled "Browsing Collection: /db" and shows a table of collections. The table has columns: Name, Permissions, Owner, Group, Created, Modified, Size (KB), and Revision. It lists five collections: "Up", "system", "todo", "twitter", and "xavier". Below the table are buttons for "Remove Selected", "Create Collection", "Upload", "New collection:", "Store as:", and "Navega...". A status bar at the bottom left says "Logged in as: xavier".

## Instal·lació

La instal·lació es fa per mitjà d'un senzill sistema d'instal·lació basat en finestres (figura 3.15).

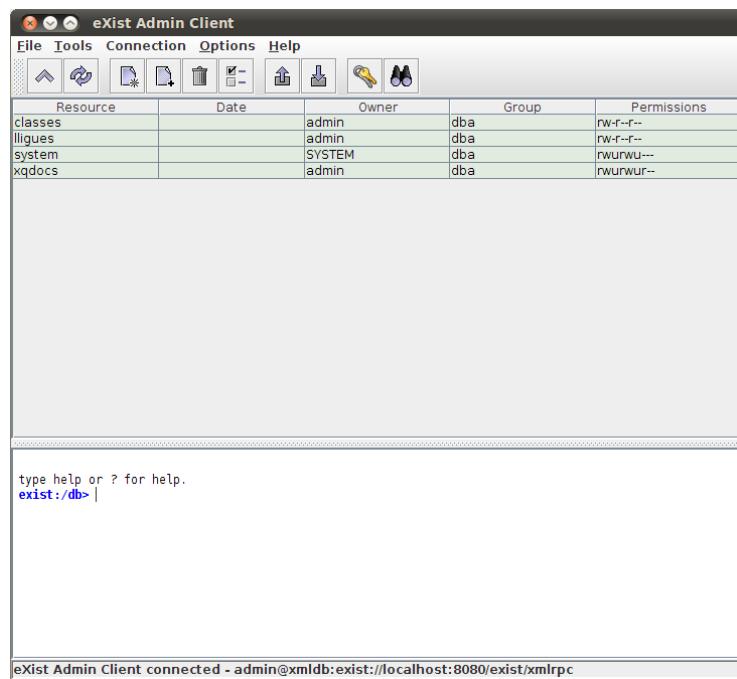
**FIGURA 3.15.** Instal·lació d'eXist



En la instal·lació per defecte a més de l'eXist s'hi instal·la el servidor Jetty, que permetrà accedir a la gestió web del sistema amb un navegador en l'adreça <http://localhost:8080/exist/>.

Si es fa servir el servidor, l'accés a les consultes i l'administració del sistema es pot fer per mitjà d'un client Java que proporciona, o bé per mitjà de l'entorn web (figura 3.16).

**FIGURA 3.16.** Client Java per connectar amb la base de dades



Tant el client com el client web disposen d'un entorn per poder elaborar consultes

XQuery (figura 3.17).

**FIGURA 3.17.** Consultes Xquery des del client

Query Dialog

Query Input:

```
History: 2. for $selec in doc("lliga.xml")//equip
for $selec in collection("//db/lligues")//partit.
let $gol := concat($selec/equip[1]/nom, " ",$selec/equip[1]/resultat..
    . . .
    ,$selec/equip[2]/nom, " ",$selec/equip[2]/resultat).
return <partit>{$gol}</partit>.
```

Context:/db/lligues

Results:

XML	Trace
<partit>Verds 2 - Vermells 0</partit>.	
<partit>Blaus 1 - Blancs 1</partit>.	
<partit>Verds 2 - Verds 1</partit>.	
<partit>Blancs 0 - Vermells 4</partit>.	
<partit>Verds 0 - Blancs 0</partit>.	
<partit>Blaus 1 - Vermells 2</partit>.	
<partit>Blancs 0 - Verds 1</partit>.	
<partit>Vermells 5 - Blaus 1</partit>.	
<partit>Verds 2 - Blaus 2</partit>.	
<partit>Blancs 0 - Vermells 1</partit>.	
<partit>Vermells 3 - Verds 1</partit>.	
<partit>Blancs 0 - Blaus 2</partit>.	
<partit>Verds 0 - Vermells 1</partit>.	
<partit>Blaus 3 - Grcos 1</partit>.	
<partit>Blaus 2 - Verds 1</partit>.	
<partit>Grcos 1 - Vermells 8</partit>.	
<partit>Blancs 0 - Verds 1</partit>.	
<partit>Blaus 1 - Vermells 1</partit>.	
<partit>Grcos 3 - Verde ></partit>.	

Found 48 items. Compilation: 14ms, Execution: 48ms

Line: 4 Column:13

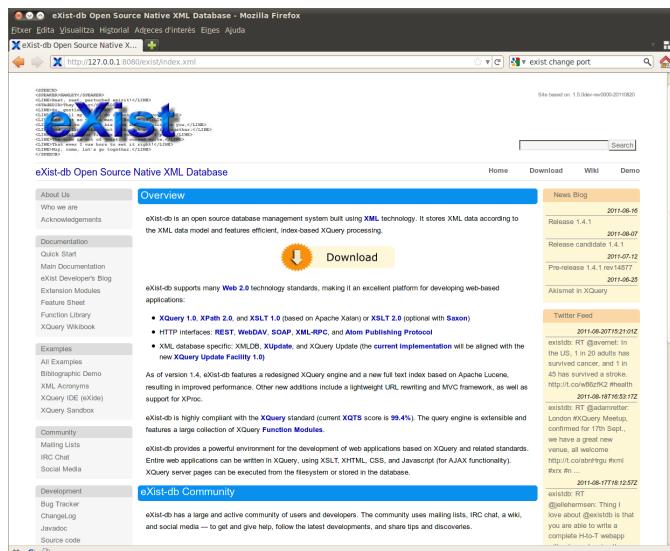
Treballar interactivament amb l'eXist

Abans de poder començar a treballar amb l'eXist cal que s'hagin entrat les dades en la base de dades. En aquest exemple ens concentrarem en com es pot fer des de l'entorn web però es pot fer d'una manera semblant des del programa client.

## Crear una col·lecció

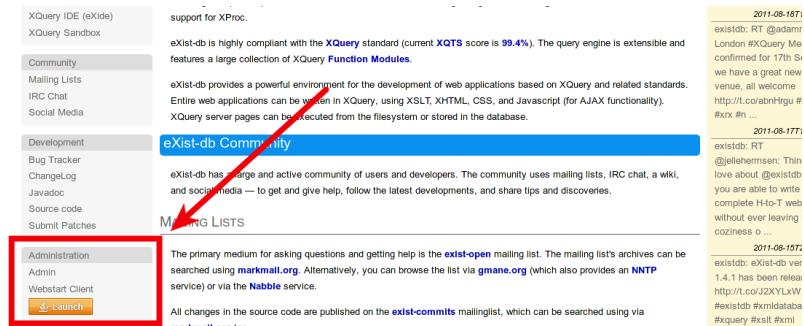
Amb l'eXist en marxa, des de la mateixa màquina on s'ha instal·lat l'eXist obriu un navegador i aneu a l'adreça <http://localhost:8080/exist/> (figurafigura 3.18).

**FIGURA 3.18.** Pàgina inicial de l'eXist



En la part inferior de la columna de l'esquerra hi ha un enllaç dins de la secció “Administration” que es diu “Admin” (figura 3.19).

FIGURA 3.19



En clicar-hi us portarà a una finestra on es demana la identificació de l'usuari. El primer cop que s'hi accedeix no hi ha usuaris, i per tant caldrà entrar amb l'usuari *admin* i la contrasenya especificada en la instal·lació (figura 3.20).

FIGURA 3.20

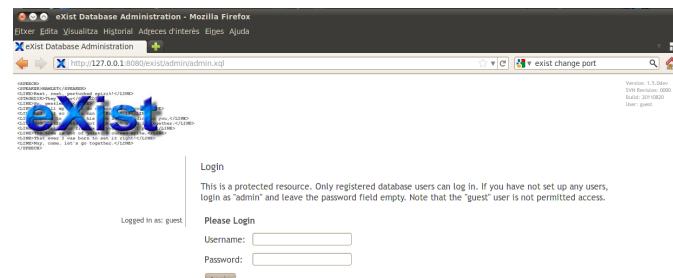
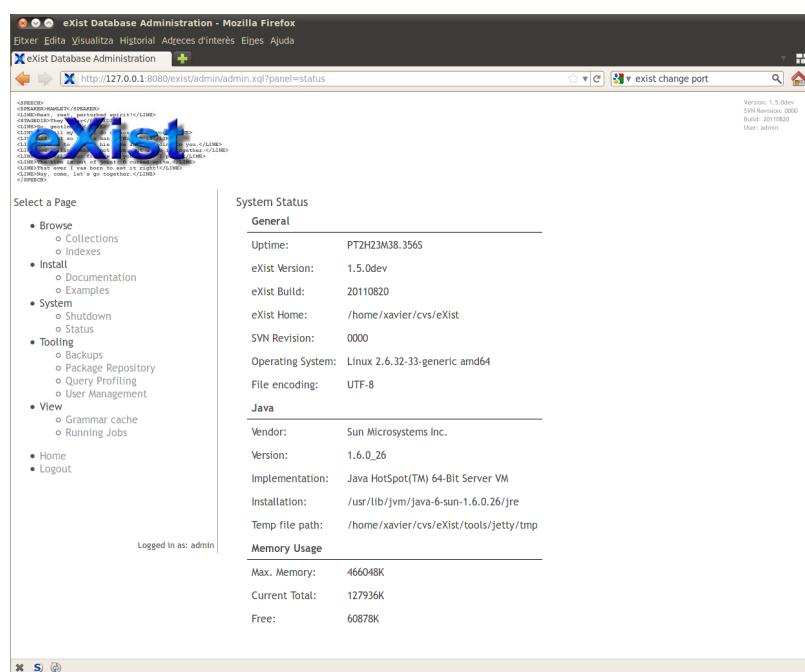


FIGURA 3.21. Pantalla d'administració de l'eXist



Un cop esteu identificats, entreu en la zona d'administració; al costat esquerre hi ha un menú que permet fer diverses coses, com navegar per les col·leccions i els índexs, aturar el servidor, instal·lar els fitxers d'exemple, crear usuaris... (figura 3.21).

Per crear una col·lecció nova cal anar al menú de l'esquerra i clicar en l'opció “Browse collections”. Això farà que aparegui tota la llista de les col·leccions instal·lades en el sistema. Per crear una nova col·lecció anomenada *classes*, cal que escrigueu el nom en el quadre de text i premeu el botó (figura 3.22).

**FIGURA 3.22.** Fer una llista de les col·leccions

En entrar-hi la col·lecció no té cap fitxer XML: els podeu afegir amb l'ajuda del botó “Upload”; i ja tindreu la col·lecció creada (figura 3.23).

**FIGURA 3.23.** Creada la col·lecció //classes//

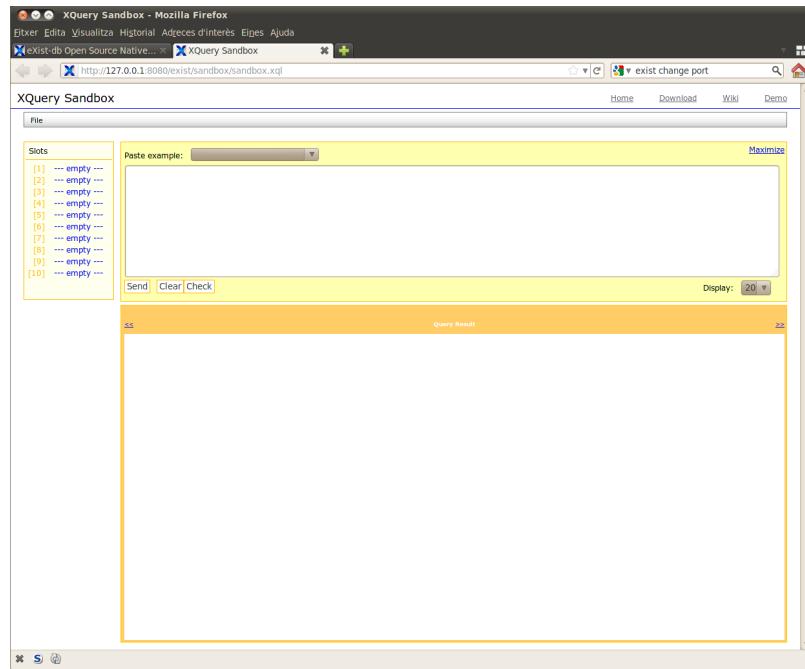
## Fer consultes XQuery

Les consultes XQuery també es poden fer des de qualsevol dels entorns que proporciona l'eXist-db. Per fer-ho des de l'entorn web cal anar a la pàgina principal i en el menú de l'esquerra, dins de l'opció “Example”, escollir l'opció “XQuery SandBox”.

Aquesta opció us portarà a una pàgina en la qual es poden fer consultes XQuery

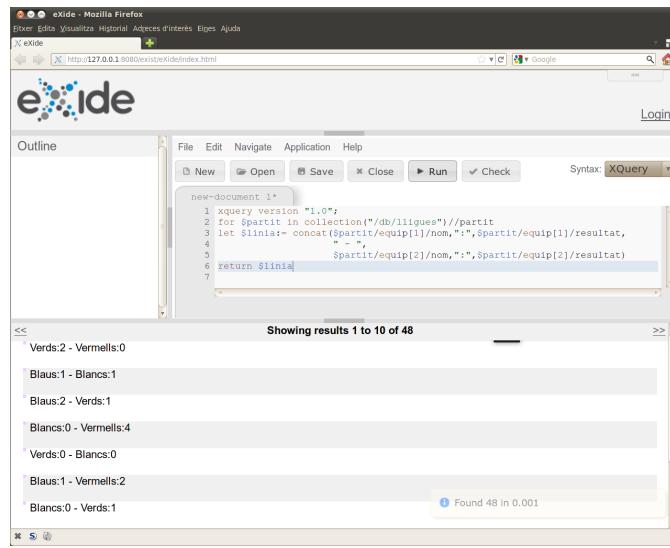
de les col·leccions o els fitxers XML que hi ha en el sistema (figura 3.24).

**FIGURA 3.24.** XQuery SandBox



A partir de la versió 1.5 s'ha afegit l'opció “XQuery IDE (eXide)”, que és un sistema molt més potent que ofereix tota una sèrie d'avantatges afegits: comprova interactivament els errors, permet autoemplenar en temps de disseny, etc. (figura 3.25).

**FIGURA 3.25.** eXide



```
1 for $alumne in collection("/db/classes")//alumne/nom  
2 return $alumne/text()
```

Que donarà el que es veu en la figura 3.26.

**FIGURA 3.26.** Noms dels alumnes de la col·lecció

The screenshot shows the XQuery Sandbox interface in Mozilla Firefox. The URL is <http://127.0.0.1:8080/exist/sandbox/sandbox.xql>. The main area contains the XQuery code:

```
for $alumne in collection("/db/classes")//alumne/nom  
return $alumne/text()
```

Below the code, there are buttons for "Send", "Clear", and "Check". To the right, a "Display:" dropdown is set to 20. The results section shows the output:

Found 6 in 0.002 seconds. Showing Items 1 to 6

Rank	Name
1	Frederic
2	Filomeno
3	Manel
4	Frederic
5	Maria
6	Bernat