Projecte final de carrera

Bases de Dades XML natives per gestionar documents MPEG-7

ALUMNE: DOMINGO PERALES LORENTE

ESTUDIS: ENGINYERIA INFORMÀTICA DE SISTEMES

CONSULTOR: ÒSCAR CELMA HERRADA

DATA: 9 de Gener del 2005

DEDICATORIA:

Especialment per la meva dona que m'ha suportat durant tot el temps que han durat aquests estudis...i per tots aquells que han aportat el seu petit gra de sorra.

Resum del projecte

Aquest projecte centra el seu estudi en el llenguatge XML i en el seu ús com a mitjà per compartir informació entre sistemes i aplicacions diferents gràcies a l'ús del format text com a base de creació.

Partint de la creació de documents ben formats i la seva validació amb els DTD's o els XML Schema, ens mostra la seva utilitat com a contenidors de dades i s'endinsa tant en les transformacions d'aquests documents en d'altres seguint els patrons de transformació creats segons les fulles d'estil XSLT, com en les cerques de parts dels documents que es poden fer amb el llenguatge de consulta XQuery amb l'ajut del llenguatge XPath per localitzar les dades interessants, passant pel llenguatge XUpdate utilitzat per fer modificacions sobre documents XML.

Un altre punt important d'aquest treball és l'estudi de dos sistemes gestors de bases de dades que utilitzen els documents XML com a magatzems de dades de la mateixa manera que els sistemes gestors de bases de dades relacionals tradicionals fan amb les taules. Aquests dos SGBD estudiats, implementats en llenguatge java, són eXist i Xindice, dels quals s'estudien avantatges e inconvenients en la seva implementació i manera de treballar.

També s'estudia l'estàndard MPEG-7 de representació d'informació relacionada amb el món audiovisual i multimedia, basada en XML Schema i XML.

I finalment, per mostrar la utilitat de totes aquestes eines treballant conjuntament, s'ha desenvolupat una aplicació que permet realitzar cerques de documents MPEG-7 via web utilitzant com a SGBD la base de dades eXist.

Índex de continguts	Pàg.

Índex de figures

Introducció

Aquest projecte es divideix en quatre apartats ben diferenciats:
El primer fa referència exclusivament al llenguatge XML i a les seves regles seguint els estàndards publicats pel W3C. El segon tracta sobre els llenguatges utilitzats per transformar els documents XML, moure's pel seu contingut i presentar-los depenent de l'aplicació. El tercer apartat és sobre les bases de dades XML natives (BD:XML natives), en concret; Xindice i eXist. L'anàlisi d'aquestes bases de dades inclou una avaluació i seguiment de la seva instal.lació a un PC, amb el sistema operatiu Linux (SUSE 9.0 professional). El quart apartat és sobre l'estàndard MPEG-7, per a descriure documents multimèdia, el qual està basat en XML. Finalment, com a cas d'ús s'ha creat una aplicació web que permet realitzar consultes sobre documents MPEG-7, descrits en XML, els quals estan emmagatzemats en una BD:XML nativa..

Resultats esperats

En quant als objectius o fites que espero aconseguir són:

- Recopilació de informació sobre XML (DTD, XML Schema, XSLT, XPath i XQuery) tant a Internet com a qualsevol font fiable que sempre ha de ser comparada amb els estàndards oficials.
- Donar una explicació clara de la utilitat de l'estàndard XML en quant a intercanvi d'informació entre diferents sistemes.
- Recopilació sobre l'estàndard MPEG-7 i comprensió del seu funcionament per realitzar un cas pràctic.
- Un anàlisi que inclogui la interrelació dels objectius anteriors. La major part de la informació tindrà com a origen Internet, ja que mitjançant cerques espero trobar tota la informació necessària.
- Aprendre a utilitzar les diferents eines i BD explicades.
- Explicar com totes aquestes noves eines i BD:XML es poden utilitzar aplicant de forma concreta al cas de consultes i treball amb documents basats en l'estàndard MPEG-7.
- Explicació del perquè de la utilitat de les BD:XML natives en front de les tradicionals en quant a el seu ús per treballar amb documents XML.

Riscos esperats

- Aquest projecte és un gran repte per mi, pel fet que quasi tots els temes que tractaré són nous i totalment desconeguts, i puc dir que cadascun d'ells mereixen un estudi a fons, que jo no puc plasmar en aquest projecte. Caldrà doncs, fer un exercici de síntesi que permeti mostrar els aspectes més rellevants de cadascun dels temes a parlar.
- Encara que puc suposar que ja comença a haver una quantitat important de documentació referent als temes que tractaré, tota està en una ràpida i constant evolució el que suposarà moltes vegades haver de comparar les diferents fonts d'informació que pugui trobar. Aquest fet m'obligarà a tenir com a única referència, o quasi exclusiva, els llocs web oficials generadors de documentació sobre els temes considerats, els quals estan en anglès. Encara que domino raonablement aquest llenguatge sempre serà una barrera alhora d'extreure tot el significat i conclusions que es poden treure dels textos llegits com a referència.
- Problemes típics deguts a la incompatibilitat de software que pugui utilitzar.
- Espero que el fet que la comunicació no presencial, amb els retards que això pugui implicar no afecti a les consultes sobre dubtes varis que puguin aparèixer.

Abast de la proposta

La següent proposta cobreix els següents continguts:

1. Recopilació i estudi del llenguatge XML
2. Recopilació i estudi de XSLT, XPath, XQuery i XUpdate
3. Recopilació i estudi de les BD:XML natives
4. Recopilació i estudi sobre documents MPEG-7
5. Cas pràctic: Documents MPEG-7

Productes Obtinguts

El producte que s'ha creat com a cas d'ús per aquest TFC és un programari, basat en l'entorn web, de cerca de documents MPEG-7. El seu aspecte és senzill i utilitza la base de dades XML nativa eXist com a motor de BD. El programari desenvolupat fa ús dels llenguatges HTML i javascript pel seu funcionament al costat del client, i el llenguatge PHP i la BD eXist del costat del servidor, de manera que l'aplicació final funciona servint dades en funció de el que demana el client.

Aquesta aplicació, degut al poc temps per la seva planificació i desenvolupament, és molt limitada, encara que amb el seu ús es poden veure algunes de les capacitats que ens ofereixen les BD:XML natives i el llenguatge XML com a eines per compartir informació entre aplicacions molt diverses.

Els documents utilitzats per fer les consultes han estat subministrats pel consultor del projecte.

1.El Llenguatge XML

Per començar a entendre una mica el que és el llenguatge XML direm és un llenguatge basat en etiquetes i també és un metallenguatge, o sigui, un llenguatge que pot ser utilitzat per descriure altres llenguatges.

Per ajudar-nos a comprendre el compararem amb un altre llenguatge d'etiquetes més senzill com és HTML. Direm que les etiquetes dels documents HTML només indiquen com s'ha de mostrar el contingut de les pàgines web, o sigui, quin serà el seu format un cop surtin per pantalla al navegador o a l'intèrpret web que les mostri. Les etiquetes que componen la pàgina web en cap moment no diuen res del significat d'una dada en concret.

Una solució al "problema" de falta d'informació (semàntica) sobre les dades ho donen els documents XML ja que en ells les dades "s'organitzen" segons el seu significat. A més a més, les etiquetes no venen ja definides com és el cas del HTML (per exemple <HTML>, <HEAD>,<BODY> ...), sinó que les etiquetes les defineix l'usuari en funció del significat que li vol donar (sempre seguint unes normes sintàctiques bàsiques).

D'aquesta manera, suposem que volem estructurar les dades d'una persona, per exemple el seu nom i la seva adreça, ho podríem definir així:

```
<persona>
  <nom>Ramon</nom>
  <cognoms>Martí Gran</cognoms>
  <adreça>
    <carrer>La Font</carrer>
    <numero>25</numero>
  </adreça>
</persona>
```

Com podem veure, cadascuna de les dades contingudes en aquest fragment s'estructuren jeràrquicament segons el seu significat.

- Les etiquetes en principi poden no significar res per a cap programa degut a que han estat creades segons el significat que s'ha cregut convenient per elles.
- Les etiquetes s'han creat "al gust" tenint en compte només el significat que se li ha volgut donar. Un cop creat el document XML hi ha aplicacions (o processadors XML) que permeten llegir els documents XML i accedir al seu contingut i a la seva estructura.

Els documents XML estan compostos per unitats d'emmagatzemat anomenades entitats, que contenen tant dades analitzades com no analitzades. Les dades analitzades estan compostes de caràcters, alguns dels quals, en la forma de dades caràcter, i altres en la forma marques (TAGS). Les marques descriuen l'estructura d'emmagatzematge del document i la seva estructura lògica. L'estàndard XML aporta una sèrie de restriccions a l'emmagatzematge i a l'estructura lògica del document.

1.1 Origen i Objectius

El llenguatge va ser desenvolupat per un grup de treball del Consorci World Wide Web (W3C) a l'any 1996. Va ser presidit per Bosak de Sun Microsystem amb la participació activa d'un grup especial d'interès en XML (que prèviament era conegut com a Grup de Treball SGML) també sota la batuta del W3C.

Els principals objectius que es van marcar alhora de descriure el futur llenguatge que s'estava dissenyant van ser:

1. XML ha de poder ser directament utilitzable sobre Internet.

2. XML ha de suportar una àmplia varietat d'aplicacions.
3. XML ha de ser compatible amb SGML.
4. L'escriptura de programes que processin XML ha de ser fàcil.
5. El número de característiques opcionals de XML ha de ser mínima, idealment zero.
6. Els documents XML han de ser llegibles per humans i raonablement clars.
7. El disseny XML ha de ser preparat ràpidament.
8. El disseny de XML ha de ser formal i concís.
9. Els documents XML han de poder ser creats fàcilment.
10. La concisió de les marques XML és de mínima importància.

El llenguatge de marques XML és un estàndard acceptat des del 10 de Febrer del 1998 en que va ser publicat el document amb la primera especificació oficial que va treure el W3C, "Extensible Markup Language (XML) 1.0." A continuació van publicar la segona edició de la versió 1.0 amb data del 6 d'Octubre del 2000 i la tercera edició que va ser publicada el 4 de Febrer del 2004.

A més d'aquesta primera versió, el W3C ha tret la versió 1.1 d'aquest estàndard amb la mateixa data que la última edició de la versió anterior, 4 de Febrer del 2004 que va ser publicada el 15 d'Abril del mateix any incloent canvis varis sobre la primera versió.

El llenguatge de marques XML és, en sí mateix, un llenguatge amb el que es poden crear tot tipus de documents basats en text, dotant-los d'una estructura interna que depèn de l'aplicació final a que van destinats. Als documents que creem se els ha d'exigir que estiguin ben formats, i si interessa se li poden afegir restriccions de creació per tal de crear uns documents que siguin vàlids respecte a aquestes restriccions per que s'ajustin a un format determinat de construcció.

Amb això ja se li veu una utilitat de funcionament interessant, però no és aquesta utilitat aïllada la que està fent que el llenguatge XML s'utilitzi cada cop més arreu del mon.

El valor ocult dels documents XML està en que en sí mateixos es poden convertir en transports d'informació entre aplicacions, ja que per la seva estructura i flexibilitat permeten que diferents aplicacions aportin un format als seus documents seguint patrons ja creats, i que de vegades són públics, permetent així intercanvis de dades entre aplicacions que d'altra manera requeririen de programes traductors d'informació per tal d'adaptar-se a qualsevol altra aplicació que volgués les seves dades, i això per cadascuna de les diferents aplicacions.

Sense fer la traducció de la informació a XML amb instruccions de processament per les diferents aplicacions els intercanvis d'informació es fan com mostra la figura següent:

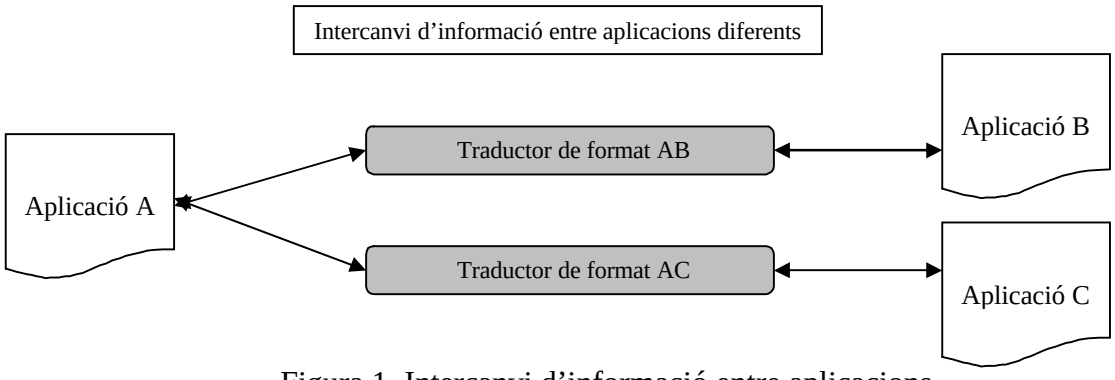

Figura 1. Intercanvi d'informació entre aplicacions

Si se fa una traducció a XML seguint patrons establerts es poden crear documents en un format comú que pot ser utilitzat directament per les altres aplicacions que també han de conèixer l'estructura amb que ha estat creat el document.

Aquest format és XML, que no deixa de ser text, per el que les aplicacions que han de llegir aquestes dades només han de conèixer el significat de les etiquetes que li donen l'estructura i llegir el seu contingut per interpretar-lo correctament.

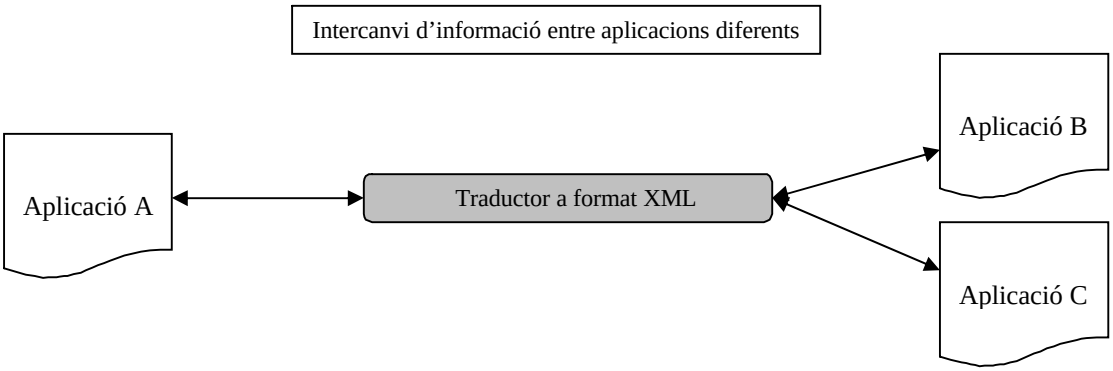

Figura 2. Intercanvi d'informació entre aplicacions utilitzant documents XML

Com que els documents XML estan en format text es poden obrir sense cap problema en diferents sistemes operatius. Una aplicació com per exemple el 'bloc de notes' de Windows o el 'vi' de Linux poden examinar-ne el seu contingut. Aquest contingut pot recordar al codi HTML d'una pàgina web, però encara que poguessin semblar iguals són ben diferents, i el seu significat també és diferent.

1.2 Diferències entre les versions 1.0 i 1.1

En aquest apartat es farà referència a les diferències més importants entre les versions 1.0 i 1.1 dels estàndards de XML.

- Degut als canvi de versió de l'estàndard UNICODE de la 2.0 a la 4.0 alguns dels nous caràcters no es poden utilitzar en la versió 1.0 de XML per definir noms d'etiquetes, atributs, etc. La versió 1.0 té una definició molt rígida en quant a la definició de noms. Així aquesta versió considera que tot el que no està permès està prohibit per definir noms.
  La versió 1.1 de XML modifica aquesta forma d'actuar i considera que tot allò que no està prohibit explícitament per definir noms està permès.
- XML 1.0 s'adapta als finals de línia de la majoria de sistemes operatius moderns deixant de costat el format utilitzat per IBM i els seus compatibles, que requereixen una traducció per poder treballar amb XML.
  XML 1.1 soluciona aquest problema afegint el codi NEL (#x85) entre els finals de línia acceptats. A més també afegeix el separador ce caràcters #x2028.
- XML 1.1 permet l'ús de caràcters de referència als caràcters de control #x1 fins al #x1F que XML 1.0 no permetia.
- Els caràcters de control #x7F fins a #x9F, que eren permesos en XML 1.0, en XML 1.1 només poden aparèixer com a referència a caràcters.

Aquests canvis provocats pel canvi de versió dels caràcters UNICODE són els que han fet canviar el número de versió de l'estàndard XML de la 1.0 a la 1.1, ja que aquest canvi afecta a la definició de bona formació dels documents XML.

1.3 Documents XML

Respecte un document XML s'hauria de diferenciar la seva estructura lògica i la seva estructura física.

L'estructura física fa referència a les diferents unitats d'emmagatzematge del contingut dels document XML, les quals s'anomenen entitats, i cadascuna d'elles ha de tenir contingut associat. Cada

document XML està contingut com a mínim dintre d'una entitat diferent, de la mateixa manera que varies entitats poden pertànyer, per estar referenciades, a un sol document XML.

L'estructura lògica fa referència a la construcció dels elements que el composen: les etiquetes d'inici, de fi, els comentaris, etc. Els documents XML han de començar amb una declaració XML per tal que es pugui especificar la versió a la qual pertany el document i per especificar si el document necessita d'altres per tal de validar-lo.

Una construcció d'un document XML ben format seria per exemple:

```
<?xml version="1.0"?>
<inici>projecte final de carrera</inici>
```

La versió (version="1.0") s'indica perquè es pugui fer una validació del contingut respecte a la versió de l'estàndard XML. Ens donarà error si el contingut no correspon con la especificació que hem indicat al nostre document.

1.3.1 Marques i dades caràcter

Quan s'escriu un document XML primer de tot hem de fer una primera diferenciació: el que són marques i tota la resta que no ho és.

- Marques: Prendran la forma d'etiquetes d'inici, etiquetes de fi, etiquetes d'elements buits, referències a entitats, referències a caràcter, comentaris, seccions CDATA, declaracions de tipus de document e instruccions de processament.
- La resta: La resta de caràcters del document XML constituirà les anomenades dades caràcter del document.

Quan escrivim el nostre document hem de tenir en compte que certs caràcters s'utilitzen com a delimitadors i caràcters especials segons l'especificació XML. Aquestos caràcters serien:

- El caràcter 'ampersand' (&) i el caràcter (<) només poden aparèixer d'aquesta manera quan s'utilitzem com a delimitadors de marques o dintre de comentaris, instruccions de processament, en seccions CDATA o en la definició del DTD intern. En el cas de que s'haguessin d'utilitzar amb qualsevol altre sentit, per exemple per una operació matemàtica de comparació '3<4' o en 'P&B' llavors hauríem de recórrer a escapar-los. Ho faríem de la següent manera: substituïm '&' per **"&"** i '<' per **"<"**.
- Però també tenim més casos: substituiríem **'>'** per **">"**
- Per tal que els atributs puguin contenir cometes simples o dobles podem escapar **(')** per **"'"** i (") per **"""**

1.3.2 Documents XML ben formats i documents XML vàlids

Un document XML està **ben format** si:

- Conté com a mínim un element, que serà l'element denominat arrel del document XML.
- Existeix un i només un element arrel que no torna a aparèixer a cap altre lloc del document.
- Tots els elements del document estan delimitats per unes etiquetes d'inici i final o per una etiqueta d'element buit, han de tenir un tipus (nom) i poden contenir un o més atributs els quals hauran de tenir el parell 'nom_atribut=valor_atribut'. Les etiquetes dels elements continguts dintre d'altres elements es tanquen abans que les etiquetes dels elements que els contenen.

Restriccions aplicables als elements:

- No han de començar per cap de les combinacions possibles de majúscules o minúscules de la cadena 'xml', ja que estan reservats per futures implementacions o versions d'aquest estàndard. A més han de començar per una lletra, un '_' o els dos punts ':' , qualsevol altra construcció seria incorrecta.
- El nom de l'etiqueta de final d'un element, en cas que no sigui un element buit, ha de ser el mateix que la de l'etiqueta d'inici. Ex:

```
<et1> </et1>
```

El valor de l'atribut ha d'anar obligatòriament entre cometes, ja siguin simples (') o dobles(") . No han de contenir entitats referència directes o indirectes a entitats externes. Un exemple d'element correcte amb contingut seria:

```
<casa num='21' color='vermell'>
    La casa que està a dalt del turó.
</casa>
```

Amb aquesta definició és fàcil veure un document XML com un arbre de dades en el que les etiquetes serien els nodes i les dades caràcter serien les fulles. En aquest arbre l'element arrel del document seria també l'element arrel de l'arbre, i la resta d'elements anirien continguts en funció de la seva posició dintre de l'estructura del document. Així el document XML tindria l'estructura d'arbre que es mostra a la seva dreta:

```
<arrel>
  <color>
        <blau>
              text
        </blau>
        <verd>
              text
        </verd>
  </color >
  <gust >
      text
      <salat>
        <poc>
              text
        </poc>
        text
    </salat>
  </gust >
  <tacte>
        text
  </tacte >
</arrel>
```

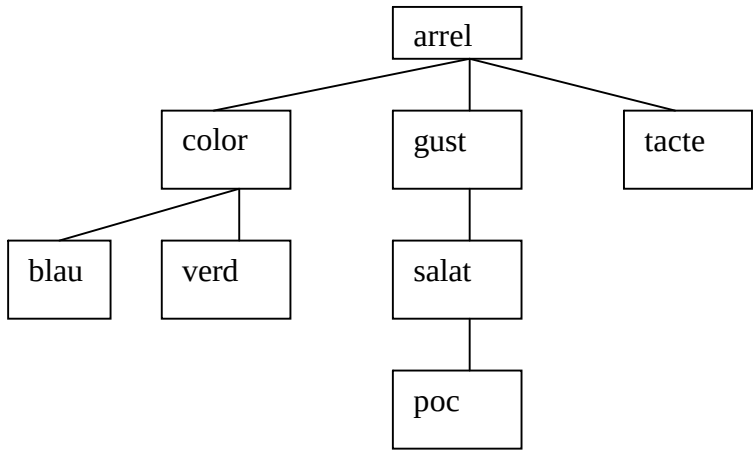

Figura 3. Arbre associat al document XML de l'exemple

A més d'aquestes propietats ha de complir l'especificació marcada segons l'estàndard del W3C sense la necessitat de seguir les indicacions de construcció de cap DTD (Definició de Tipus de Document) [veure secció 1.2.4 per més detalls]. A més, cadascuna de les entitats analitzades a què es fa referència de forma directa o indirecta també han d'estar ben formades.

Un document ben format pot a més ser **vàlid** si té associat un DTD i a més compleix les restriccions de construcció en ell indicades en quan l'estructura del contingut.

Cal tenir en compte que un document pot estar ben format però no ser un document vàlid. Però no es pot donar el cas que sigui un document vàlid no estigui ben format.

1.3.3 Declaració XML

Quan es crea un document XML, tal com s'ha comentat anteriorment, el primer que farà un processador XML és comprovar si és un document XML ben format, però per això cal que aquest conegui (tot i que no és obligatori) alguns paràmetres del document que ha d'analitzar, com per

exemple la versió de l'estàndard XML que utilitza, a més d'altres paràmetres que poden ser necessaris per una correcta definició del document.

La declaració del document XML es fa a l'inici de document, i és fa de la següent manera:

- Comença amb '<?xml' i acaba amb '?>'
- Si s'inclou la declaració, aquesta ha de portar obligatòriament el número de versió, amb la cadena 'version="número versió"' (entre cometes dobles o simples). En cas que donem un número de versió i el contingut del document no concordi amb l'especificació el processador XML ha de retornar un error.
- Opcionalment es pot incloure el nom de la codificació que s'utilitza amb encoding="nom de la codificació"
- També, de forma opcional, es pot indicar si el nostre document XML es validarà contra algun DTD tot indicant 'standalone=yes/no' segons el cas.

Una possible definició de la capçalera d'un document XML podrien ser, per exemple:

- <?xml version='1.0' encoding='UTF-16' standalone='yes' ?>
- <?xml version='1.0' encoding='US-ASCII' ?>
- <?xml version='1.0' ?>

El primer exemple indica que el document ha de complir amb les especificacions de la versió 1.0 de l'estàndard XML, que la codificació a utilitzar és UTF-16 i que el document no s'ha de validar contra cap DTD.

A continuació de la declaració XML es poden afegir comentaris i seguidament a aquests comentaris, abans dels elements, podem inserir una declaració de DTD interna [veure secció 1.2.4 per més detalls]. al propi document XML. Aquesta declaració faria la mateixa funció que un DTD extern, i en cas de complir-se les regles de construcció marcades s'obtindria un document XML vàlid.

1.3.4 Comentaris

Els significats dels comentaris és el mateix que en qualsevol altre llenguatge de programació i s'utilitzen per que es puguin fer indicacions dintre del document XML sense que això afecti en res el contingut. Els comentaris, com la resta de components d'un document han de seguir les regles de creació i han d'aparèixer només en els llocs permesos.

Els comentaris han de començar per **'<!--'** i terminar amb **'-->'**. Per exemple:

<!-- això és un comentari vàlid per un document XML -->

A causa de les restriccions de compatibilitat, dintre dels comentaris no han d'aparèixer dos guions junts '--'.

1.3.5 Instruccions de processament

Com s'ha comentat anteriorment el llenguatge XML està essent utilitzat per gran varietat d'aplicacions diferents que tractaran les dades en funció de les seves necessitats i del punt del document que estiguin tractant, i és per això que els documents XML poden incloure també marques que portaran informació específica depenent de l'aplicació que els pot tractar.

És per això que tenim les instruccions de processament (PI). La forma de construir-les és la següent, comencen amb la capçalera '<?' i terminen amb '?>'. Dintre d'aquestes marques especials que no pertanyen a les dades caràcter del document s'ha d'especificar el nom de l'aplicació cap a la que van destinades, i a continuació el cos de la instrucció que hi volem passar-li. La construcció final d'una PI seria, per exemple:

```
<? nomAplicacio instruccio_que_volem_passar ?>
<?perl lower-to-upper-case  ?>
<?web-server add-header = "universidad" ?>
```

1.3.6 Seccions CDATA

Les seccions CDATA serveixen per escapar blocs sencers de text, dintre dels quals s'entén que el que apareix són tot dades caràcter i que no pertanyen a l'estructura de marques del document XML.

Aquestes seccions comencen per '<![CDATA[' i terminen per ']]>'. D'aquesta manera, si volguéssim escriure l'expressió "el número 3<5 i el 3>1" ho podríem fer així:

```
<![CDATA[el número 3<5 i el 3>1]]>
```

En cas de no haver utilitzat una secció CDATA ho hauríem d'haver escrit així:

```
el número 3&lt;5 i el 3&gt;1
```

1.3.7 Espai de noms (Namespaces)

A mesura que els documents XML van creixent i són utilitzats per multitud d'aplicacions, un dels problemes amb que poden trobar-se aquestes aplicacions és la utilització dels mateixos noms a les etiquetes que han d'interpretar diferents aplicacions per les quals aquestes etiquetes tenen significats diferents. A tipus de problemes és el que va crear la necessitat dels anomenats espais de noms (namespaces en anglès).

La funció dels namespaces és fer una partició del conjunt d'etiquetes que composen el document XML de manera que es puguin utilitzar els mateixos noms d'etiquetes aplicant-los una petita variació en la seva construcció. Aquesta variació és la inserció de prefixos diferents per crear la separació entre els diferents grups d'etiquetes que volem utilitzar, i amb això se les està dotant de significats diferents.

Quan es defineix un espai de noms, les etiquetes queden dividides en dues parts, "namespace:etiqueta". La part del namespace és la que marca la diferència de l'espai de noms al que pertany l'etiqueta. En un document XML es poden crear tants espais de noms com sigui necessari, i a més poden estar barrejats els seus continguts. Exemple:

Amb espais de noms	Sense espais de noms
<?xml version="1.0" encoding="iso-8859-1"?> <!-- Descripció dels elements d'una casa --> <micasa > <nom>Villa Maria</nom> <habitacio id="menjador"> <moble>aparador</moble> <moble> <nom>Sofà</nom> <descripcio>Pelut</descripcio> <mida>enorme</mida> </moble> </habitacio> </micasa>	<?xml version="1.0" encoding="iso-8859-1"?> <!-- Descripció dels elements d'una casa --> <mc:micasa xmlns:mc='http://www.adreça.org/micasa' xmlns:moble='http://www.adreça.org/moble'> <mc:nom>Villa Maria</mc:nom> <mc:habitacio id="menjador"> <mc:moble>aparador</mc:moble> <mc:moble> <moble:nom>Sofà</moble:nom> <moble:descripcio>Pelut</moble:descripcio> <moble:mida>Enorme</moble:mida> </mc:moble> </mc:habitacio> </mc:micasa>

En aquest segon exemple es pot veure que els dos espais de noms comencen la seva declaració amb "xmlns:nom_de_l'espai_de_noms" i a continuació la URI que els distingeix universalment.

També es pot crear un espai de noms per defecte. Per fer-ho quan es fa la declaració de l'espai de noms no s'indicarà cap nom per l'espai de noms. Si s'agafes la declaració d'espai de noms anterior i es

volgués convertir en un espai de noms per defecte es faria sense el prefix de l'espai de noms així <micasa xmlns='http://www.adreça.org/micasa'> [comprovar diferència amb la declaració anterior], i un cop fet això tots els elements que queden entre les etiquetes d'inici i final d'aquest element i que no contenen cap prefix d'espai de noms passarien a pertànyer a aquest espai de noms per defecte.

1.4 Declaració del tipus de document (DTD)

Anàlogament als documents XML, els documents DTD estan formats per text.

A mesura que es creen les marques i s'afegeixen els atributs corresponents dintre del document, s'està modificant el seu contingut, però alhora es pot o no estar modificant la seva estructura lògica. De vegades interessarà mantenir un cert format, una certa estructura lògica de creació del nostre document amb una sèrie de regles prèviament definides per l'usuari o que ja venen donades per entitats externes que s'associen al document. El DTD, doncs, conté l'esquelet (gramatical) sobre el qual s'aguantarà el document XML.

Gràficament seria:

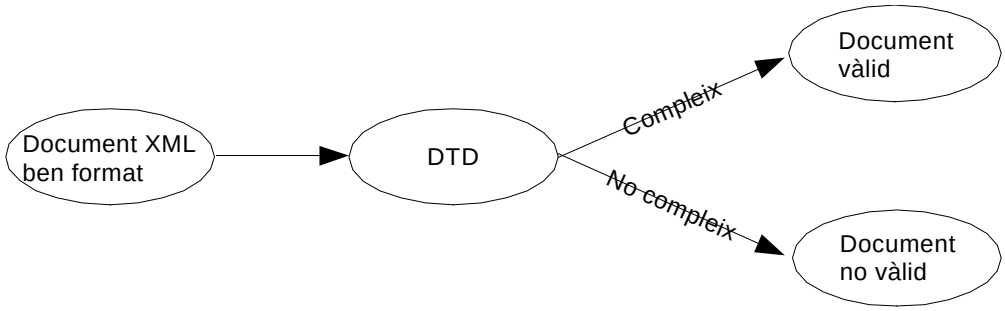

Figura 4: Validació de documents XML amb un DTD

Aquests DTD contra els que es pot validar el document XML poden ser:

- Interns al propi document, tal com s'ha comentat.
- Extern, cas en que estiguin definits en una entitat física diferent i que per tant haurà de ser cridada abans de poder fer la validació. Aquests documents externs poden ser de dos tipus:
  - Públics: Són accessibles per tothom.
  - No públic: Lo contrari dels anteriors.

La declaració del DTD es fa després de la declaració XML i abans del primer element del document.

El DTD conté una descripció per la construcció del document mitjançant una correcta organització dels elements que ha de contenir, dels quals s'ha de indicar el tipus mitjançant l'assignació d'un nom concret a cadascun d'ells i la seva cardinalitat.

Els DTD poden ser: Interns al document XML, externs, interns i externs alhora, públics i no públics.

1.4.1 DTD intern

Tot seguit es mostra un exemple de declaració d'un DTD intern:

```
<?xml version='1.0' standalone='yes'?>
<!DOCTYPE ORDEN_DE_COMPRA[
  <!ELEMENT ORDEN_DE_COMPRA (CLIENT)>
  <!ELEMENT CLIENT (NUMERO_DE_COMTE, NOM_COMPLET)>
  <!ELEMENT NUMERO_DE_COMTE (#PCDATA)>
  <!ELEMENT NOM_COMPLET (#PCDATA)>
]>
```

En aquest DTD intern es pot veure la declaració del elements i l'ordre correcte d'aparició que han de tenir els elements. L'element ODREN_DE_COMPRA serà l'element arrel del document i dintre d'ell trobarem elements del tipus CLIENT que a la seva vegada estarà compost pels elements del tipus NUMERO_DE_COMPTE i NOM_COMPLET, amb aquest ordre i amb aquestos noms. El contingut d'aquestos dos últims elements seran dades, o sigui dades caràcter.

Cal tenir molt en compte l'espai que es deixa entre 'NOM_COMPLET' i '(#PCDATA)', ja que si no el processador XML mostrarà un error. A continuació la resta del document que segueix el DTD anterior:

```
<ORDEN_DE_COMPRA>
   <CLIENT>
         <NUMERO_DE_COMTE>12345678 </NUMERO_DE_COMTE>
         <NOM_COMPLET> Manel pèrez</NOM_COMPLET>
   </CLIENT>
</ORDEN_DE_COMPRA>
```

1.4.2 DTD extern

El DTD estarà en un document diferent, en aquest cas tenim un DTD extern.

La forma d'indicar al processador XML que la validació del document és farà contra un DTD extern és amb la l'atribut 'standalone' de la declaració XML, ara el valor és 'standalone=no'. Fent això el processador XML ja sap que cal obtenir un document extern per tal de poder fer la validació. Per especificar un DTD extern s'utilitza una declaració DOCTYPE afegint la localització del DTD corresponent.

Per que el DTD de l'exemple anterior sigui extern es pot haver agafat tot el que hi ha entre les marques '[' i ']' i es pot guardar en un arxiu amb el nom 'compra.dtd' (al mateix directori, per exemple). En aquest cas per fer la validació del document de forma correcta s'ha de modificar el document deixant-lo de la següent manera.

Primer es crearia el DTD en l'arxiu 'compra.dtd':

```
<!ELEMENT ORDEN_DE_COMPRA (CLIENT)>
<!ELEMENT CLIENT (NUMERO_DE_COMTE, NOM_COMPLET)>
<!ELEMENT NUMERO_DE_COMTE (#PCDATA)>
<!ELEMENT NOM_COMPLET (#PCDATA)>
```

I a continuació es crea el document XML que se validarà contra el DTD que s'acaba de crear:

```
<?xml version='1.0' standalone='no'?>
<!DOCTYPE ORDEN_DE_COMPRA SYSTEM "directori_ treball\compra.dtd">
<ORDEN_DE_COMPRA>
   <CLIENT>
```

```
    <NUMERO_DE_COMTE>12345678 </NUMERO_DE_COMTE>
    <NOM_COMPLET> Manel pèrez</NOM_COMPLET>
  </CLIENT>
</ORDEN_DE_COMPRA>
```

1.4.3 DTD intern i extern

Aquests DTD són una combinació dels dos anteriors, per tant tindrem una referència externa a un DTD que especificarà una part de la gramàtica que haurà de complir el document XML, i la part del DTD intern especificarà la resta de la gramàtica que haurà de complir el document.

Per combinar-los correctament s'ha de tenir en compte que:

- Els DTD siguin compatibles entre si, que siguin complementaris.
- Que un DTD no anul·li declaracions fetes a l'altre. Els DTD no haurien de definir els mateixos tipus a les declaracions per que es podrien crear incompatibilitats no desitjades.
- Es poden tornar a definir elements dels DTD extern en el intern, ja que el que s'especifica en el DTD intern té valor sobre el que s'especifica en el extern.

Per fer una declaració de DTD mixta incloent tant una referència externa com una interna es farà posant primer la referència al DTD extern i a continuació fer la del intern tal com s'havia fet quan aquest anava com a única alternativa.

Per fer un exemple ampliarem l'anterior exemple extern afegint informació addicional a l'ordre de compra. Ara s'inclouran dos elements nous, que ampliarà la definició feta del client afegint també el número de telèfon i l'adreça de correu electrònic. Així la definició completa del DTD i la creació d'un element quedaria com segueix:

```
<?xml version='1.0' standalone='no'?>
<!DOCTYPE ORDEN_DE_COMPRA SYSTEM "directori_ treball\compra.dtd" [
<!ELEMENT CLIENT (NUMERO_DE_COMTE, NOM_COMPLET, TELEFON,MAIL)>
<!ELEMENT TELEFON (#PCDATA)>
<!ELEMENT MAIL (#PCDATA)>
]>

<ORDEN_DE_COMPRA>
  <CLIENT>
    <NUMERO_DE_COMTE>12345678 </NUMERO_DE_COMTE>
    <NOM_COMPLET> Manel pèrez</NOM_COMPLET>
    <TELEFON>977555555</TELEFON >
    <MAIL>manel@correu.org</MAIL >
  </CLIENT>
</ORDEN_DE_COMPRA>
```

1.4.4 DTD extern públic

L'objectiu dels documents DTD públics és que tothom pugui construir documents XML basats en un mateix DTD. Es diferencien dels privats utilitzant la paraula clau 'PUBLIC' a continuació del tipus de l'arrel del DTD que estem especificant. Un exemple:

```
<!DOCTYPE spec PUBLIC "-//W3C//DTD Specification V2.0//EN"
"http://www.w3c.org/XML/xmlspec-v20.dtd">
```

En aquesta especificació del DTD es poden veure els elements següents:

- L'element arrel és del tipus 'spec'.
- El DTD al que fem referència és d'accés públic.
- "-//W3C//DTD Specification V2.0//EN" que ens informa de el següent:
  - Les contrabarres '//0 que separen les diferents categories que la composen.
  - El signe menys '-' ens indica que DTD és un estàndard no reconegut, no oficial. En cas de ser un símbol més '+' hauria informat d'un estàndard oficial.
  - Informació sobre el propietari d'aquest DTD, en aquest cas 'W3C'.
  - Una descripció del contingut del DTD 'DTD Specification V2.0'.
  - Dos lletres que ens indiquen del llenguatge del documents XML pels quals s'utilitza aquest DTD 'EN', per indicar que estaran en anglès.
- La URL: "http://www.w3c.org/XML/xmlspec-v20.dtd", que ens diu a on poden trobar el DTD anomenat 'xmlspec-v20.dtd'

1.4.5 DTD extern NO públic

Quan ens troben en el cas que els documents que volem validar o han de fer contra un DTD privat a una empresa o simplement a algú que no vol que sigui públic llavors el que es fa és substituir la indicació "PUBLIC' per 'SYSTEM'. Per tant si la declaració anterior hagués estat per a un DTD no públic se especifica d'aquesta altra manera [comparar diferencies amb l'anterior]:

```
<!DOCTYPE spec SYSTEM "http://www.w3c.org/XML/xmlspec-v20.dtd">
```

1.4.6 Definició d'elements al DTD

Per definir elements s'ha de fer de la manera següent:

- Declaració d'inici de la definició: '<!ELEMENT'
- Opcionals:
  - Es pot indicar si existeix una definició externa per aquest element, que se indicaria posant 'SYSTEM "URI_del_DTD"'. Un exemple seria: <!DOCTYPE pag_web SYSTEM "http://www.la_meva_adreça.org">
  - Declaracions de marcació internes al propi document o declaracions d'entitats per referència entre les marques '[' i ']' . Aquesta serà una declaració de tipus d'element, de llista d'atributs, d'elements o de notació.
- Etiqueta de fi de declaració: '>'
- Entre les etiquetes d'inici i de fi es declararan els diferents elements i la seva cardinalitat.

El nom de l'element serà el nom que se li posa és el del tipus que se li assigna a l'element. Cap tipus d'element no es pot declarar més d'una vegada.

El contingut de l'element que se indica depèn de el que contindrà l'element. Els diferents tipus de contingut d'un element poden ser:

Tipus de contingut	Declaració al DTD	Al document XML
EMPTY La etiqueta estarà buida	<!ELEMENT casa EMPTY>	<casa></casa>, o també <casa/>
Només elements Contindrà altres elements dintre seu	<!ELEMENT casa (habitació1, habitació2)> <!ELEMENT habitació1 EMPTY> <!ELEMENT habitació2 EMPTY>	<casa> <habitació1></habitació1> <habitació2/> </casa>
Només text Només contindrà dades caràcter, no altres elements. Són elements	<!ELEMENT habitació (#PCDATA)>	<habitació> Aquesta habitació m'agrada </habitació>

terminals, nodes fulla.		
Contingut barrejat Contindrà primer dades caràcter i després altres elements.	<!ELEMENT casa (#PCDATA, habitació)> <!ELEMENT habitació1 (#PCDATA)>	<casa> Aquesta es una casa gran <habitació> Habitació verda </habitació> </casa>
Qualsevol contingut Tot tipus de contingut sense ordre establert.	<!ELEMENT mon ANY>.	<mon> El mon és gran <país>i el meu país petit</país> <poble/> </mon>

Quan és crea un DTD a més del contingut que ha d'aparèixer al document XML també es poden fer indicacions d'ordre i cardinalitat d'aquests elements. A la taula següent és mostren els indicadors d'ordre i de cardinalitat que podem utilitzar en la definició del DTD.

Taula d'indicadors d'ordre i qualificadors d'elements			
Tipus	Valor	Acció	Descripció
Indicador d'ordre ','	,	Seqüència	Un element ha d'anar a continuació de l'anterior
Indicador d'elecció	|	Elecció	O un element o l'altre, però no els dos
Indicador d'agrupació	()	Agrupació	Els elements van junts
Sense qualificador	Absència de qualificador en un element	Requerit i únic	Fins ara els elements no tenien qualificadors, això indicava que havien d'aparèixer un sol cop. Els qualificadors quan apareixen indiquem com han d'aparèixer.
Qualificador '?'	?	Opcional i únic	L'element apareix opcionalment, i si ho fa serà un sol cop.
Qualificador '*'	*	Opcional i repetible	L'element apareix opcionalment, i ho pot fer de manera repetida.
Qualificador '+'	+	Requerit i repetible	L'element ha d'aparèixer almenys un cop i es pot repetir.

Exemples d'elecció i agrupació:

Indicador	Definició al DTD	Creació d'elements al document XML
Elecció '|'	<!ELEMENT A (B | C | D) > <!ELEMENT B (#PCDATA) > <!ELEMENT C (#PCDATA) > <!ELEMENT D (#PCDATA) >	Són declaracions vàlides d'elements les següents: • <A> <B> Element B </B> </A> • <A> <C> Element C </C> </A> • <A> <D> Element D </D> </A> Qualsevol altra construcció no serà vàlida
Agrupació '()'	<!ELEMENT A (B, (C | D)) > <!ELEMENT B (#PCDATA) > <!ELEMENT C (#PCDATA) > <!ELEMENT D (#PCDATA) >	Es te una agrupació entre els elements C i D, i entre ells un símbol d'elecció que només els afecta a ells. Construccions que donarien documents XML vàlids. • <A> <B> Element B </B> <C> Element C </C> </B> </A> • <A> <B> Element B </B> <D> Element D </D> </B> </A> Qualsevol altra construcció donarà documents XML no vàlids.

Com a exemple complet mostraré possibles documents XML pel DTD següent:

```
<!ELEMENT casa ( num_porta, num_pis, telefon, codi_postal ?, ( email_casa | (email_mati +, email_tarda * ) ) >
<!ELEMENT num_porta ( #PCDATA) >
<!ELEMENT num_pis ( #PCDATA) >
<!ELEMENT telefon ( #PCDATA) >
<!ELEMENT codi_postal ( #PCDATA) >
<!ELEMENT email_casa ( #PCDATA) >
<!ELEMENT email_mati ( #PCDATA) >
<!ELEMENT email_tarda ( #PCDATA) >
```

```
<casa>
  <num_porta> 25 </num_porta>
  <num_pis> 3 B </num_pis>
  <telefon> 555-2316121
</telefon>
  <codi_postal> 57841
</codi_postal>
  <email_casa> casa@correu.net
</email_casa>
</casa>
```

```
<casa>
  <num_porta> 25 </num_porta>
  <num_pis> 3 B </num_pis>
  <telefon> 555-2316121
</telefon>
  <email_mati> mati@correu.net
</email_mati>
  <email_tarda> tarda@correu.net
</email_tarda>
</casa>
```

```
<casa>
  <num_porta> 25 </num_porta>
  <num_pis> 3 B </num_pis>
  <telefon> 555-2316121
</telefon>
  <codi_postal> 57841
</codi_postal>
  <email_mati> mati1@correu.net
</email_mati>
  <email_mati> mati2@correu.net
</email_mati>
  <email_mati> mati3@correu.net
</email_mati>
</casa>
```

1.4.7 Declaracions de llistes d'atributs.

Els atributs s'utilitzen per donar propietats o característiques especials a elements concrets. Els atributs no pertanyen a les dades caràcter de l'element. Aquests també s'han de definir dintre del DTD com a llistes d'atributs tal com apareixeran dins dels documents XML si volem aquests siguin vàlid. La manera de definir una llista d'atributs és la següent:

- Inici de la declaració de la llista d'atributs: '<!ATTLIST'
- Especificar a quin element pertany l'atribut
- Donar un nom a l'atribut i decidir el seu tipus
- Especificar la seva utilització (CDATA, ENTITY... a continuació es fa referència)
- Final de la definició de la llista d'atributs: '>'

La llista de tipus d'atributs se inclou a continuació

Tipus de l'atribut	Descripció del contingut
CDATA	Cadenes de text
ENTITY	Nom d'una entitat definida prèviament. Una entitat és un component que ja existeix i que por ser substituït per un document. Per exemple podem substituir una entitat per un bloc de text, un vídeo, un arxiu de so...
ENTITIES	Llista de noms ENTITY
ID	Identificador únic per a un element. Per poder referenciar un element inequívocament dintre del document. Actua com un índex de valors únics. Només pot haver un per element
IDREF	Fa referència a un atribut ID definit prèviament. El valors de IDREF han de ser iguals a algun dels valors ID a que fan referència
IDREFS	Llista de IDREF
NOTATION	Conté el nom d'una notació declarada al DTD. Les notacions s'utilitzen per identificar el format utilitzat per la informació que no és XML
NMTOKEN	Una cadena de caràcters que no pot incloure espais en blanc
NMTOKENS	Llista de NMTOKEN
ENUMERATION	Llista de possibles valors que por tenir un atribut. Quan apareixen, el valor de l'atribut només pot incloure un d'aquests valors

Aquests serien els tipus que es poden indicar per un atribut, però a més del tipus també es pot indicar l'ús que anem a fer d'aquest atribut per defecte. A continuació ve una taula amb els diferents usos que se li poden donar a un atribut

Paraula clau	Ús de l'atribut	Valor per defecte	Descripció
#IMPLIED	Atribut opcional	No té , no permès	El valor de l'atribut és opcional
#FIXED	Atribut fix	Predeterminat, necessari i fix	El valor predeterminat de l'atribut s'ha de posar de manera obligatòria, i és l'únic que pot prendre aquest atribut. Es pot veure com una constant per l'element. Si un atribut ha estat declarat #FIXED el processador XML considerarà que té aquest valor com si estigués present
#REQUIRED	Atribut obligatori	No té valor per defecte ni és permès	El valor de l'atribut és obligatori i ha d'aparèixer quan és declara l'element.

1.5 XML Schema

El XML Schema va ser originalment proposat per Microsoft, però es va convertir en una recomanació del W3C al maig del 2001, i disposa d'una versió estable revisada pels seus membres.

Problemes trobats a l'ús dels DTD:

- Llenguatge propi amb una sintaxi diferent al llenguatge XML.
- No permet l'ús d'espais de noms [namespaces, apartat 1.3.7].
- Quantitat molt limitada de tipus de dades.
- Els mecanisme d'extensió és complex i fràgil.

Per solucionar en part aquests problemes es va crear el XML Schema . Avantatges del XML Schema en front als DTD:

- Esta escrit en XML.
- Té un conjunt molt més ample de tipus base ja definits.
- Permet tipus de dades definits per l'usuari, anomenats arquetipus.
- Es poden validar documents que contenen espais de noms.
- Permet herència entre arquetipus, aprofitant altres ja existents.

Gràficament es pot mostrar la funció del XML Schema, similar als DTD.

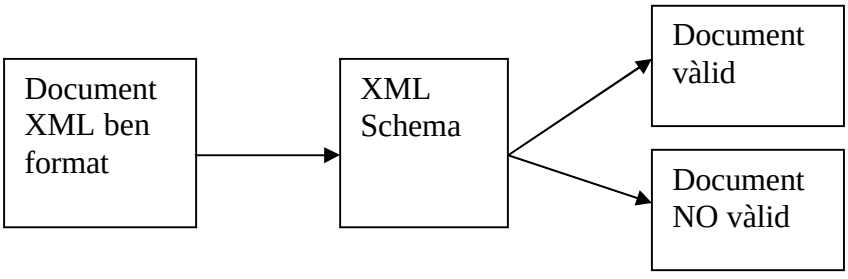

Figura 5. Validació d'un document XML contra un XML Schema

Tindrem un espai de noms XML en el que estan definits els diferents tipus de dades bàsics, i l'espai de noms d'usuari, en el que es pot definir, els seus propis tipus de dades complexos derivant-los d'aquestos tipus bàsics i ampliant les restriccions d'aquests.

En la definició de l'esquema s'indica quin pot ser l'element arrel, i es defineixen els elements indicant quins seran elements complexos formats a partir dels que ja no contindran més elements al

seu interior. També se indicaran els noms i tipus dels atributs que tindran els elements definint de igual manera les seves restriccions d'us i cardinalitat.

Document XML	**Schema XML per validar el document XML**
<?xml version="1.0"?> <note xmlns=http://www.w3schools.com xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:schemaLocation="http://www.w3schools.com/nota.xsd"> <note> <to>Pere</to> <from>Domingo</from> <heading>Recorda</heading> <body>Anar al cine el dissabte!</body> </note>	<?xml version="1.0"?><xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.w3schools.com" xmlns="http://www.w3schools.com" elementFormDefault="qualified"> <xs:element name="note"> <xs:complexType> <xs:sequence> <xs:element name="to"type="xs:string"/> <xs:element name="from"type="xs:string"/> <xs:element name="heading" type="xs:string"/> <xs:element name="body" type="xs:string"/> </xs:sequence> </xs:complexType> </xs:element> </xs:schema>

1.6 Fulles d'estil en cascada i XML

Les fulles d'estil s'utilitzen amb els documents XML per dotar-los d'un significat davant dels navegador, ja que per si mateixos, si s'utilitzen etiquetes diferents de les de HTML, no entenen ni l'estructura ni el contingut de uns documents que no els hi proporcionen cap informació de com presentar-los.

Les fulles d'estil en cascada (CSS, Cascading Style Sheets) permeten crear i utilitzar estils propis del HTML que modifiquin la presentació de les dades contingudes en el document XML. La manera de indicar-li al navegador que el document que està carregant utilitza una fulla d'estil es fa amb la instrucció de processament

```
<?XML:stylesheet type="text/css" href="fulla_estils.css"?>
```

Amb aquesta instrucció se li està indicant al navegador que quan carregui el document XML el mostri segons els estils continguts a la fulla d'estil anomenada "fulla_estils.css". El document XML d'exemple seria:

```
<?xml version="1.0" encoding="UTF-8"?>
<?XML:stylesheet type="text/css" href="fulla_estils.css"?>
<document>
  Aquest document és un exemple per veure el funcionament de les CSS
  <titol>un titol qualsevol
   <contingut>i aquest seria el contingut
   </contingut>
  </titol>
</document>
```

Amb l'ús de diferents fulles d'estil es poden crear diferents presentacions

CSS utilitzat	Resultat al navegador
document { display: inline } titol, contingut { display: block }	Aquest document és un exemple per veure el funcionament de les CSS un titol qualsevol i aquest seria el contingut
document { display: inline } titol, contingut { display: block } titol { font-size: 1.3em } contingut { font-style: italic } titol, contingut { margin: 0.5em }	Aquest document és un exemple per veure el funcionament de les CSS **un titol qualsevol** *i aquest seria el contingut*
document { display: inline } titol, contingut { display: block } titol { color: blue } contingut { color: red }	Aquest document és un exemple per veure el funcionament de les CSS un titol qualsevol i aquest seria el contingut

2 XSL

Els documents XML poden ser utilitzats en combinació d'altres eines per tal de poder accedir de maneres diferents a les dades i poder-les mostrar segons interessi, fins al punt que la informació que s'obtingui estigui personalitzada per la persona o l'aplicació que la demana.

Aquesta és la finalitat dels XLS (Extensible Stylesheet Language Family), la família de les fulles d'estil extensibles, que esta composta de tres parts diferents:

- XSLT, un llenguatge per transformar XML i mostrar-lo amb formats diferents segons estigui definit.
- XPath, un llenguatge d'expressions utilitzat pes XSLT per accedir o fer referència a certes parts del document XML.
- XSL-FO, un llenguatge per especificar semàntiques de format per l'XML.

Aquestes tres parts permeten definir polítiques d'accés a la informació i la forma de mostrar-la per pantalla o de crear documents diferents partint d'un conjunt de dades comú en funció de el que es demana.

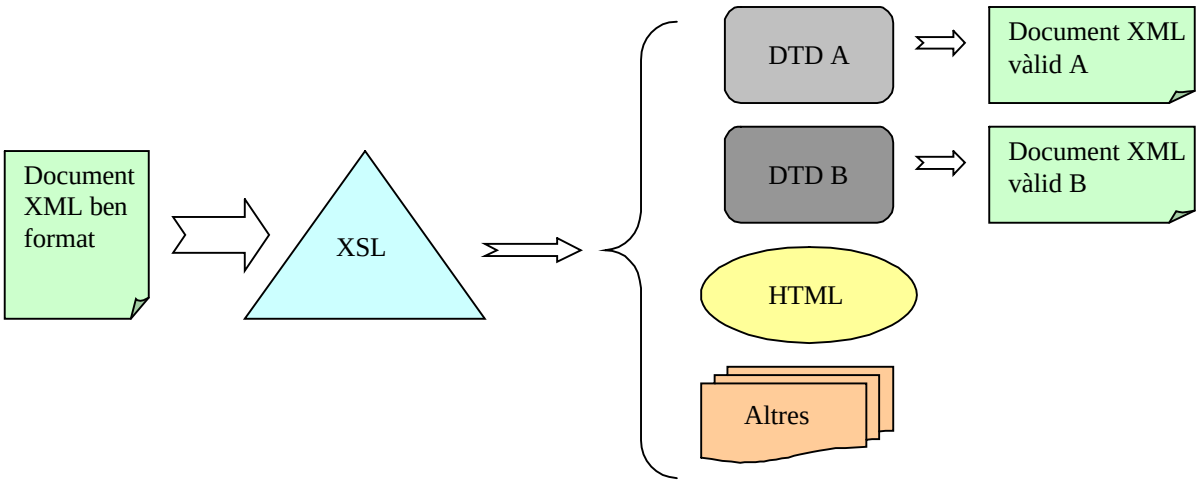

Figura 6 Gràfic de possibles resultats utilitzant XSL i XML

2.1 XSLT

2.1.1 Introducció a XSLT

XSLT és un llenguatge utilitzat per convertir documents XML en altres documents XML. Les mateixes fulles d'estil XSLT també han de ser documents XML ben formats. Per fer aquestes transformacions s'apliquen una sèrie de regles que indiquen que s'ha de fer en funció del que és. Es parteix d'un arbre XML origen i es converteix en un altre arbre XML resultant utilitzant les fulles d'estil XSLT com a plantilla pels canvis. Les fulles d'estil XSL fan una cerca pel document XML en funció dels paràmetres de cerca fins trobar el patró desitjat, i un cop el troba aplica la plantilla corresponent que produirà el resultat final.

XSLT utilitza el llenguatge d'expressions XPath **[veure apartat 2.2]** alhora de fer les seleccions d'elements dintre de l'arbre XML, per processament condicional i per generar text. Un exemple d'ordenació alfabètica utilitzant XSLT.

Document XML	Fulla d'estil XSLT	Resultat
<?xml version="1.0" encoding="UTF-8"?> <?xml-stylesheet href="ordena.xsl" type="text/xsl"?> <raiz> <cosa>Pepe </cosa> <cosa>Juan </cosa> <cosa>Enrique </cosa> <cosa>Xabier </cosa> <cosa>Aarón </cosa> </raiz>	<?xml version="1.0" encoding='ISO-8859-1'?> <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template match='raiz'> <html> <head><title>Ordenando</title></head> <body> <xsl:apply-templates select='cosa'> <xsl:sort select='.' data-type='text'/> </xsl:apply-templates> </body> </html> </xsl:template> <xsl:template match='cosa'> <xsl:value-of select='.'/> </xsl:template> </xsl:stylesheet>	1. Aarón 2. Enrique 3. Juan 4. Pepe 5. Xavier

2.1.2 Elements XSLT

A continuació es donarà una visió dels elements XSLT que es poden incloure en una fulla d'estil:

XSLT	Descripció
<xsl:stylesheet>	Aquest element és l'arrel de quasi totes les fulles d'estil XSLT.
<xsl:template>	Element utilitzat per trobar patrons dintre del document XML.
<xsl:apply-templates>	S'utilitza per processar tots els nodes que l'anterior element ha trobat.
<xsl:value-of>	S'utilitza per obtenir només el valor del contingut PCDATA contingut dintre de les etiquetes del node cercat per la expressió Xpath.
<xsl:output>	Permet definir propietats sobre el tipus de sortida que es vol generar
<xsl:element>	Permet crear elements nous de manera dinàmica dintre de l'arbre de resultats obtinguts en una expressió de cerca.
<xsl:attribute>	Funciona de manera similar al element <xsl:element>, i el que fa és afegir un atribut de manera dinàmica a l'arbre de resultats.
<xsl:text>	Permet inserir text a l'arbre resultant en funció de quin sigui el node de context.
<xsl:copy>	Copia el node actual sobre el que es fa la crida, però no es copien ni els

	seus atributs ni els seus elements fills.
<xsl:copy-of>	Permet copiar porcions de l'arbre origen a l'arbre resultat, incloent elements i atributs.
<xsl:sort>	Permet ordenar un conjunt de nodes en funció del seu contingut o atributs.
<xsl:variable>	Permet definir variables dintre dels documents XSL, de manera que si després es vol incloure aquest resultat en un altre lloc del document es pot fer invocant el nom que se li ha assignat a aquest valor.
<xsl:if>	Permet definir una estructura condicional simple que serà tinguda en compte en cas que es compleixi la condició que presenta en la seva etiqueta.
<xsl:choose>	Aquest element es pot considerar com la versió múltiple de l'element anterior, ja que dintre d'ell es poden agrupar varies opcions de comparació amb la possibilitat d'una opció que s'executarà per defecte en cas que totes les demés siguin errònies.
<xsl:for-each>	Permet aplicar les mateixes instruccions a una sèrie d'elements que compleixen una condició comuna.

2.2 XPath

2.2.1 Introducció a XPath

Els elements de les fulles d'estil XSLT de l'apartat anterior treballaven sobre els elements dels documents XML d'entrada. Per localitzar els elements o nodes XML i poder desplaçar-se per ells necessita d'alguna indicació, i això ho fa utilitzant XPath, que proporciona la versatilitat necessària per tal que l'accés als elements es pugui fer seguint regles més potents i complexes que la simple elecció d'un element en concret tal com es troba al document XML.

XPath és un llenguatge per adreçar parts d'un document XML designat per ser utilitzat per XSLT i XPointer. Per fer-ho converteix el document XML en un arbre de nodes a on tots els elements que componen el document tenen un node que els representa. D'aquesta manera XPath treballa sobre una estructura lògica abstracta i no directament sobre el document XML. Així, utilitzant una sèrie de regles de localització creades per aquest llenguatge, i que no són XML, pot accedir a qualsevol element en funció de l'element de context considerat amb l'ús d'una notació de camins, anomenats camins de localització, juntament amb capacitats per comparar si els nodes obtinguts gràcies a aquesta notació encaixen en els patrons desitjats per tal d'escollir o rebutjar els elements de manera individual.

A més de les capacitats bàsiques de localització i de selecció de nodes de que disposa també proporciona facilitats bàsiques per la manipulació de tipus bàsics.

2.2.2 Arbre del document XML

El que fa XPath és convertir el document XML en un arbre de nodes lògic en el que els elements es distribueixen en funció de la seva aparició al document. Així construeix una estructura amb uns camins que no deixin lloc a errors o dobles interpretacions com per exemple possibles bucles, i tots els elements tenen una posició única. Els tipus de nodes en que es poden convertir els diferents elements d'un document XML són:

Tipus de node	Propietats del node
Arrel	• Node arrel de l'arbre. • Només n'hi ha un i està al principi de l'arbre i correspon al primer element que obre el document XML després de la declaració d'aquest.

	• Tots els altres, siguin del tipus que siguin són descendents d'ell. • No tenen noms expandits seguint el format dels espais de noms definits per XML
Element	• Hi ha un per cada element del document XML. • Tots els nodes element tenen un i només un element pare i zero o més elements fills. • Tenen nom expandit.
Atribut	• Es corresponen amb els atributs dels element que apareixen al document XML. • El node pare d'un node atribut és un node element, però els nodes atribut no es consideren fills d'aquest node element. • El seu contingut és un valor de cadena. • Tenen un nom expandit.
Espai de noms	• Es corresponen als diferents prefixes d'espai de noms amb efecte sobre un element. • Cada node element té un d'aquest tipus de nodes associat en funció dels espais de noms amb efecte sobre ell. • L'element del que pengen aquests nodes és el seu element pare, però ells no es consideren elements fills d'aquests element. • Tenen un nom expandit • El URI del seu espai de noms és sempre nul.
Instrucció de processament	• Es corresponen a les instruccions de processament que es poden trobar al document XML. • Tenen un nom expandit. • La part local del nom expandit és el destinatari d'aquesta instrucció. • El seu URI és nul
Comentari	• Es corresponen amb els comentaris que apareixen al document XML excepte pels que apareixen dintre de la declaració del document. • No tenen nom expandit.
Text	• Corresponen a totes les dades caràcter de cadascun dels elements del document XML. • No tenen nodes germans immediatament anteriors ni posteriors que també siguin nodes text. • No tenen nom expandit.

2.2.3 Camins de localització

Els camins de localització són les expressions utilitzades per localitzar els diferents nodes que formen l'arbre lògic creat per XPath. N'hi ha de dos tipus:

Relatius:	**Absoluts:**
Utilitzen el node de context com a punt inicial de referència d'inici de la cerca, i els diferents passos que el formen estan separats pel símbol "/". Exemple: child::node Selecciona els elements "node" fills de l'element de context.	Expressen una posició dintre de l'arbre començant a partir del node arrel com a referència, que és representa per un símbol "/" a l'inici de l'expressió del camí de localització, i a continuació un camí relatiu. Exemple: /descendant::elem/child::node selecciona els nodes "node" que són fills d'un node "elem" que estan al mateix document que el node de context.

Els camins de localització comencen d'esquerra a dreta encadenats amb el símbol "/" de manera que el conjunt de nodes obtingut com a resultat d'un pas és el conjunt de context del pas següent. Cadascun d'aquests passos es divideix al seu temps per tres parts diferents:

- Un eix: Que especifica la relació jeràrquica entre els nodes que se cerquen.
- Una prova de node: Que especifica el tipus de node i el nom expandit dels nodes seleccionats pel pas de localització.

- Predicats de selecció: zero o més, que són expressions que s'utilitzen per refinar encara més les cerques dels nodes seleccionats.

2.2.4 Eixos

Els eixos corresponen a la direcció en que s'ha de fer la cerca dintre de l'arbre de nodes, de manera que hi ha eixos per tots els conjunts de node que fan referència al node de context. D'aquests eixos, com es veurà, n'hi ha que no són aplicables al node arrel, ja que no es pot fer referència, per exemple al node pare del node arrel o als seus nodes germans. La taula completa dels eixos és:

Nom de l'eix	Descripció
child	Conté els nodes fills del node contextual
descendant	Són tots els nodes que pengen d'ell, ja sigui directa o indirectament. No conté als nodes atribut ni nodes espai de noms.
parent	Conté, en cas que existeixi, el node pare del node contextual.
ancestor	Conté els nodes ascendents del node contextual. Aquest eix sempre conté el node arrel excepte quan el node de context sigui el node arrel.
following-sibling	Conté tots els nodes germans que van a continuació del node contextual, o sigui, tots els elements germans que apareixen més tard que ell al document XML. No s'inclouen ni els nodes atributs ni espai de noms.
preceding-sibling	Conté els nodes germans anteriors al node contextual. No s'inclouen ni els nodes atributs ni espai de noms.
following	Conté tots els nodes del document que van a continuació del node contextual en l'ordre que apareixen al document XML, excloent els que queden dintre de l'eix descendant, els nodes atribut, i espai de noms.
preceding	Conté tots els nodes del document que van abans que el node contextual en l'ordre que apareixen al document XML, excloent els que queden dintre de l'eix ancestors, els nodes atribut i espai de noms.
attribute	Conté els nodes atribut del node contextual, en cas que no n'hi hagi cap l'eix estarà buit.
namespace	Conté els nodes espai de noms del node contextual, en cas que no n'hi hagi cap l'eix estarà buit.
self	Només conté el propi node contextual.
descendant-or-self	Conté tots els nodes que la unió dels nodes *descendant* i *self*.
ancestor-or-self	Conté tots els nodes que la unió dels nodes *ancestor* i *self*.

L'arbre format per tots els nodes, si ignorem els nodes atribut i espai de noms, queda particionat sense superposició de nodes pels eixos ancestor, descendant, following, preceding i self. A més, també es poden indicar de manera més compacta si utilitzem la versió abreviada de molts d'ells:

Operador	Descripció
/	Selecciona els fills. Si esta al principi del camí de selecció representa l'arrel.
//	Selecció de descendents. Selecciona tots els descendents.
.	Selecció de l'element actual (el context).
*	Tots (en el sentit habitual d'aquest operador)
@	Prefix que s'anteposa al nom d'un atribut.
[ ]	Filtre sobre el conjunt de nodes seleccionats.

2.2.5 Proves de node

Les proves de node fan referència al tipus de node que cada eix pot contenir. Per tant quan se selecciona un eix a partir de l'element contextual, l'element al que se vol fer referència a de ser del tipus que pot contenir l'eix. Per exemple, l'eix *attribute* conté nodes de tipus atribut. Si no n'hi ha cap

node del tipus especificat per la prova de node com a resultat de la cerca llavors es retorna un conjunt buit com a resposta.

Com a molts altres llenguatges també es disposa d'elements "comodín" que serveixen per a indicar que se esta buscant un node qualsevol del tipus especificat per l'eix, i en aquest cas aquest element, al igual que en molts altres llenguatges de programació és el símbol "*". Altres proves de node:

Prova de node	Descripció
text()	Certa per nodes de tipus text.
comment()	Certa per nodes de tipus comentari
processing-instruction()	Certa per nodes del tipus instrucció de processament
node()	Certa per qualsevol node del tipus que sigui

2.2.6 Predicats

Els predicats s'utilitzen per acabar de filtrar un conjunt de nodes per tal d'obtenir un conjunt encara més refinat seguint les indicacions marcades. El resultat de l'avaluació del predicat per cadascun dels nodes considerats consisteix en un dels dos possibles resultats, cert o fals. Si és cert s'inclou a la llista dels nodes cercats, i en cas que sigui fals s'ignora.

Alhora d'avaluar els predicats un altre cosa que s'ha de tenir en compte és la proximitat entre els elements cercats i el node de context, ja que pot ser que s'estigui buscant, per exemple, el primer fill d'un node, l'últim germà del node de context, el tercer fill del segon fill partint des del node arrel,etc.

2.2.7 Exemples de camins de localització

A continuació es mostraran alguns exemples de camins de localització amb una breu explicació del seu significat:

- *child::a1*
  Es un camí relatiu que selecciona els elements *a1* que són fills del node contextual. L'eix d'aquest camí és *child* i la prova de node és *a1*. Aquest camí no té cap predicat.
- *self::cadira*
  Camí relatiu que selecciona el node de context si aquest és del tipus *cadira*.
- */*
  Camí absolut que selecciona l'arrel del document.
- *child::**
  Selecciona tots els elements fills del node contextual.
- */descendant::cotxe*
  Camí absolut que selecciona tots els elements cotxe que són descendents del node arrel, que estiguin al mateix document que el node de context, sempre que aquests no siguin atributs ni espais de noms.
- *child::cotxe [position()=5]*
  Camí relatiu que selecciona el cinquè element cotxe que és fill del node contextual.
- *child::*/child::cadira [position()=last()]*
  Selecciona l'últim net (fill del seus fills) de tipus cadira de l'element contextual .

2.2.8 Biblioteca de funcions de XPath

XPath, al igual que la resta de llenguatges de programació posseeix una llibreria de funcions pròpies mínima que han d'incloure tots els processadors que entenguin XPath. A continuació les inclourem dividint-les en funció del seu ús.

Les funcions estan compostes per tres parts:

- El valor que retorna la funció, és del tipus que indica el prototip de la funció.
- El nom de la funció, va entre el valor de retorn i els parèntesis del paràmetres.
- Els paràmetres que rep la funció. Aquestos paràmetres van entre parèntesis i han de ser del tipus que indica la crida a la funció. Un símbol "?" darrera de l'atribut indica que és opcional, en cas contrari és obligatori. Una funció pot no tenir paràmetres o tenir més d'un

Els tipus de dades que admeten i retornen aquestes funcions són:

- number, un número.
- String, un valor de cadena.
- boolean, un booleà.
- node-set, un conjunt de nodes.
- object, un objecte.

Funcions de conjunts de nodes

- number last()
  Retorna la posició de l'últim element d'un conjunt.
- number position()
  Retorna un número igual a la de la posició contextual de l'element que se li demana.
- number count(node-set)
  Retorna el número de nodes del conjunt de nodes que se li passa com a paràmetre.
- node-set id(object)
  Selecciona elements basant-se en el seu identificador únic.
- string local-name(node-set?)
  Retorna la part local del nom expandit del primer node del conjunt de nodes argument. En cas de no passar-li paràmetre pren com a paràmetre per defecte el node contextual.
- string namespace-uri(node-set?)
  Retorna el URI de l'espai de noms expandit del primer node del conjunt de nodes paràmetre. En cas de no passar-li paràmetre pren com a paràmetre per defecte el node contextual. La cadena de retorn serà sempre buida si no es tracta de nodes element o nodes atribut.
- string name(node-set?)
  Retorna el nom expandit del primer node del conjunt de nodes del paràmetre

Funcions de cadenes

- string string(object?)
  Converteix l'objecte paràmetre en una cadena depenen del tipus d'objecte.
- string concat(string, string, string*)
  Retorna la concatenació del les cadenes passades com a paràmetre.
- boolean starts-with(string, string)
  Retorna un valor cert en cas que la primera cadena argument comenci amb la segona o la segona estigui buida. Retornarà fals altrament.
- boolean contains (string, string)
  Retorna un valor cert en cas que la segona cadena estigui continguda en la primera o que la segona estigui buida, altrament serà fals.

- string substring-before(string,string)
  Retorna la subcadena que precedeix a la segona cadena de paràmetre dintre de la primera, o una cadena buida si la segona no conté la primera. Si la segona cadena està buida llavors es retorna la primera.
- string substring-after(string, string)
  Retorna la subcadena que segueix a la segona cadena de paràmetre dintre de la primera o una cadena buida si la primera no conté la segona. Si la segona cadena està buida llavors es retorna la primera.
- string substring(string, number, number?)
  Retorna una sub-cadena de la cadena d'argument que comença a la posició marcada pel primer número de paràmetre o fins al final si no te segon número.
- number string-length(string?)
  Retorna el número de caràcters de la cadena passada com a paràmetre, o, en cas de rebre paràmetres agafa com a cadena el valor del node de context convertit a cadena.
- string normalize-space(string?)
  Retorna la cadena d'argument amb l'espai en blanc normalitzat. Si s'omet el paràmetre agafa com a paràmetre el node contextual convertida com a cadena.
- string translate(string, string, string)
  Retorna la primera cadena de paràmetre en la que els caràcters de la segona han estat substituïdes pels de la tercera.

Funcions booleanes

- boolean boolean(object)
  Converteix l'objecte d'entrada a boolea.
- boolean not(boolean)
  Retorna el valor contrari al que rep per paràmetre.
- boolean true()
  Retorna un valor cert.
- boolean false()
  Retorna un valor fals
- boolean lang(string)
  Retorna un valor cert si el llenguatge del node contextual es el mateix o un sub-llenguatge del llenguatge passat com a paràmetre.

Funcions numèriques

- number number(object?)
  Converteix l'objecte rebut com a paràmetre en un número. Si s'omet el paràmetre agafa el node contextual com a únic objecte.
- number sum(node-set)
  Converteix tots els nodes argument a números y retorna la seva suma.
- number floor(number)
  Retorna el major número enter que no sigui més gran que el número passat com a paràmetre.
- number ceiling(number)
  Retorna el menor número enter que no sigui més petit que el número passat com a paràmetre.
- number round(number)
  Retorna el número enter més proper al número que se li passa com a paràmetre.

3 XQuery

Un cop tenim un llenguatge de marques com XML i un altre llenguatge com és XPath per poder localitzar parts concretes dels documents un pas lògic a això és un llenguatge de consultes com el que es tractarà a continuació.

XQuery és un llenguatge desenvolupat pel W3C per fer consultes a col.leccions de dades XML. XQuery també proporciona els mecanismes necessaris per extreure de manera fàcil i eficient, informació de BD:XML natives com es podrà veure més endavant en aquest treball.

XQuery és un llenguatge de consultes, però no es tracta pas d'un llenguatge procedural com ho són els llenguatges de programació clàssics (ex. C, pascal, etc.) sinó un llenguatge declaratiu. Cadascuna de les consultes XQuery és una expressió a ser avaluada, i així el resultat final d'una consulta complexa és igual a l'encadenament de totes les parts que composen la consulta, a on cada part treballa sobre el resultat retornat per la part anterior.

XQuery suporta els espais de noms, i per tant s'ha de tenir en compte a l'hora de cridar funcions o utilitzar variables, ja que segons l'espai de noms al que estan definits tindran un significat o un altre, i per tant el resultat final pot ser molt diferent de l'esperat si això no es té en compte.

3.1 XPath i XQuery

XQuery utilitza àmpliament el llenguatge d'expressions XPath per seleccionar parts del documents XML. De fet XQuery 1.0 i XPath 2.0 estan sent desenvolupats per mateix grup de treball del W3C, anant les seves especificacions de manera paral.lela.

A XQuery, les expressions XPath són un tipus de consulta simple, encara que normalment formen part d'una consulta més complexa. Per tant ens trobem que XPath per si sol és útil per extraccions de dades, però és amb les expressions FLWOR (podem pronunciar-lo flower en aquest context) de XQuery amb les que adquireix tota la força i capacitats que necessita per poder ser utilitzat com a llenguatge de consultes de BD:XML, que és el que estem estudiant en aquest treball. XQuery utilitza notació XML per definir les consultes i tractar els resultats.

XPath va ser en principi part de XSLT 1.0 [**veure apartat 2.1**] , però la versió XPath 2.0 s'està desenvolupant com a una especificació a part.

3.2 Expressions FLWOR

El nom de FLWOR li ve de les paraules clau de les clàusules utilitzades per fer les consultes, que són FOR, LET, WHERE, ORDER BY i RETURN. La seva utilització i significat són els següents:

- **FOR**: Associa una o més variables a expressions, creant un conjunt de tuples en el que cada tupla vincula una variable donada a cadascun dels ítems als que està associada l'expressió avaluada. Aquesta clàusula aporta un mecanisme per fer iteracions degut a la creació d'aquest conjunt de tuples.
- **LET**: Vincula una variable al resultat d'una expressió, afegint aquest vincle a les tuples generades per la clàusula FOR. Un altre ús que se li sol donar és el d'assignació de variables.
- **WHERE**: Aquest clàusula s'utilitza per filtrar les tuples inicials de manera que només accepta aquelles que compleixen les condicions de filtrat que se li associen.
- **ORDER BY**: Ordena les tuples que formen el resultat fins al moment.
- **RETURN**: Construeix el resultat de l'expressió FLWOR per una tupla donada. Aquesta clàusula és avaluada per cadascuna de les tuples que han passat el filtre imposat per la clàusula WHERE i ordenades per ORDER BY.

3.3 Accés a dades externes

Amb XQuery podem crear expressions que s'avaluaran sense la necessitat d'accedir a cap dada o document extern i obtenir un resultat d'aquestes expressions, com per exemple:

```
let $a:=1 return
let $b:=2 return
if(a<b) return <menor>a</menor>
else return <menor>b</menor>
```

Però precisament un dels aspectes més interessants de XQuery és poder accedir a les dades d'un document extern i treballar amb elles, per fer-ho disposem de les funcions "doc()" i "collection()" per poder importar XML extern dins del programa que ha de treballar amb ell.

- La funció "doc()" agafa la URI d'un document i retorna els nodes del document. Per exemple, la següent instrucció llegiria els nodes del document "exemple.xml" situats a la carpeta "exemples" de la unitat "C:"

  doc("c:\exemples\exemple.xml")

- La instrucció "collection()" agafa la URI que se li passa i retorna una llista de nodes. Aquesta instrucció interpreta aquesta URI en funció de la seva implementació.

Les clàusules FLWOR permeten la utilització de més d'un document com a font de dades alhora de fer una consulta simple. Al fer consultes de més d'un document podem crear consultes com si es tractés d'una consulta "INNER JOIN" típica de SQL amb diferents taules.

Quan fem una consulta d'aquest tipus la clàusula FOR permet crear conjunts de tuples amb tots els elements que necessitem dintre de variables independents en forma de producte cartesià amb tots els possibles valors. Un cop fet això la clàusula WHERE pot filtrar els parells d'elements de cada tupla que compleixen les condicions requerides. I al final la clàusula RETURN crea els elements finals, a partir dels elements que han superat el filtrat, donant format final per que sigui retornat com volem.

Per evitar errors s'ha de tenir molt en compte que sempre que utilitzem les clàusules LET i FOR aquestes han d'anar acompanyades d'un RETURN per que retorni un resultat cap al pas següent o com a resultat final de la consulta. Si no ho fem obtindrem una sortida d'error per part del programa, de la mateixa manera que si posem un RETURN sense haver posat anteriorment un LET o un FOR.

Un exemple de consulta sobre dos documents seria la següent:

Arxiu1.xml	Arxiu2.xml	Consulta XQuery

<?xml version="1.0"?>
<llibreria>
<llibre>
<titol>Títol 1</titol>
<preu>50 €</preu>
<pagines>200</pagines>
</llibre>
<llibre>
<titol>Títol 2</titol>
<preu>60 €</preu>
<pagines>170</pagines>
<titol>Títol 3</titol>
</llibre>
<llibre>
<preu>12 €</preu>
<pagines>20</pagines>
</llibre>
</llibreria>
	<?xml version="1.0"?>
	<botiga>
	<llibre>
	<titol>Títol 6</titol>
	<preu>15 €</preu>
	<pagines>20</pagines>
	</llibre>
	<llibre>
	<titol>Títol 9</titol>
	<preu>66 €</preu>
	<pagines>450</pagines>
	</llibre>
	<llibre>
	<titol>Títol 2</titol>
	<preu>67 €</preu>
	<pagines>170</pagines>
	</llibre>
	</botiga>
		<comparacio_preus>
		{
		for $a in doc("arxiu1.xml")//llibre, $b in doc("arxiu2.xml")//llibre
		where $a/titol=$b/titol
		return
		<llibre>
		{$a/titol}
		
		<arxiu1>{$a/preu/text()}</arxiu1>
		<arxiu2>{$b/preu/text()}</arxiu2>
		</llibre>
		}
		</comparacio_preus>

Resultat obtingut

```
<comparacio_preus>
  <llibre>
    <titol>Títol 2</titol>
    <arxiu1>60 €</arxiu1>
    <arxiu2>66 €</arxiu2>
  </llibre>
</comparacio_preus>
```

3.4 Condicionals

Aquest llenguatge de consultes ens ofereix també una expressió condicional "IF/THEN/ELSE" que actua de la manera que ho fa en els llenguatges de programació tradicionals, avaluant la primera opció si la avaluació retorna un resultat cert, o la segona opció, la de la part del "ELSE", en cas contrari.

Una diferència amb les instruccions condicionals tradicionals és que en aquest cas la clàusula "ELSE" és obligatòria, ja que en XQuery totes les expressions han de retornar un valor. Com a exemple aquest petit fragment de codi:

```
let $a:=1 return
let $b:=
      if($a=0) then "a és igual a 0"
      else "a és diferent de 0"
      return $b
```

3.5 Comentaris

Quan es defineix una consulta en XQuery, que no sempre serà de reduïdes dimensions, pot ser molt interessant posar indicacions sobre allò que fa, de manera que l'autor de la consulta o qualsevol altre ho pugui revisar i entendre. Per fer-ho tenim la possibilitat d'incloure comentaris dintre de les consultes que no es consideraran part de la consulta. La manera de fer-ho és escrivint els comentaris tancats entre els símbols "(:" i ":)", de manera que un comentari tindria l'aspecte següent:

```
(: Això és un comentari :)
```

3.6 Variables

Les variables que utilitzem a les consultes del llenguatge XQuery han de ser creades abans de ser utilitzades, en cas contrari donarà error. Tenim maneres diferents d'associar valors a una variable concreta que després poden ser utilitzades segons interessi.

Les variables que utilitzem van acompanyades del símbol "$" al davant per diferenciar-les d'altres parts de la consulta. A aquestes variables les hi podem associar valors de maneres diferents, per exemple:

- Amb l'ús de LET tenim una assignació instantània i total de tots els elements
  `let $a:=5` , associem un 5 a la variable $a
  `let $a:="hola caracola"` , associem la cadena "hola caracola" a la variable $a
  `let $a:=("hola","adeu")` , associem la variable $a a la seqüencia ("hola","adeu"), de manera que $a té tots els valors de la seqüència al mateix temps.
- Amb FOR tenim una associació iterativa dintre dels elements possibles
  `for $a in ("Joan","Pere","Josep")` , associa la variable $a a la seqüència indicada.
  `for $a in doc("arxiu1.xml")//llibre` , associa la variable "$a" a totes les tuples de tipus "llibre" del document "arxiu1.xml" de manera iterativa

Les variables tenen visibilitat dintre del bloc de codi al que van ser definides, i si una variable interna a un bloc de codi es declarada amb el mateix nom que una variable més externa, llavors la més interna amaga el valor de la més externa.

Hi ha una altra manera de declarar les variables en XQuery diferent de la que hem comentat abans, i és amb la utilització de "declare variable". Si ho fem d'aquesta manera podem incloure, apart del valor de la variable, el tipus de dades que contindrà aquesta variable, i a més podem indicar si aquesta variable ha de rebre el seu valor d'un entorn extern. Alguns exemples són:

- declare variable $x as xs:integer:=7;
- declare variable $x:=7;
- declare variable $x as xs:integer external;
- declare variable $x external;

La variable que porta la paraula "external" ha de rebre el seu valor d'un entorn extern abans de poder ser utilitzada.

Quan la variable definida també porta un tipus associat llavors el seu valor ha de ser del tipus definit, ja que en cas contrari l'aplicació retornarà un error.

En el cas que la variable no porti un tipus associat llavors se li assignarà un en funció del valor que se li assigni.

3.7 Funcions definides del llenguatge

Com tots els altres llenguatges, XQuery ja porta implementades una sèrie de funcions per fer aquelles accions més comunes i bàsiques que després és poden utilitzar per implementar d'altres de més complexes per part dels usuaris com serien funcions matemàtiques, de cadenes, de data i hora, etc.

Aquestes funcions normalment ja estan escrites dintre de l'espai de noms "fn" associat a la URI de l'espai de noms "http://www.w3.org/2002/QUERY_functions", i que com és un espai de noms que sol ser per defecte a les aplicacions normalment no caldrà indicar-lo.

3.8 Funcions definides per l'usuari

A més de les funcions que ja porta el llenguatge i que poden ser cridades en qualsevol moment, XQuery també permet a l'usuari crear les seves pròpies funcions en funció de les seves necessitats per ser utilitzades i reutilitzades directament o per crear d'altres més complexes. Aquestes funcions que poden crear poden ser recursives i també mútuament recursives, que són aquelles funcions que es criden l'una a l'altra.

Explicaré amb un exemple com definir una funció:

```
declare function local:mi_funcion($e as node()) as xs:integer
{
  --cos de la funció--
}
```

Si quan es defineix una funció no es defineix el tipus d'algun paràmetre o del valor de retorn aquest, per defecte agafa el tipus ítem.

Si declarem una funció amb el mateix nom i número d'elements es produirà un error per part de l'aplicació. La visibilitat dels paràmetres d'una funció serà dintre de la pròpia funció que els declara.

3.9 Quantificadors

De vegades és molt útil disposar de algun sistema que ens digui si dintre d'un conjunt n'hi ha algun que compleix una determinada condició o si tots la compleixen. Això en XQuery es fa amb els quantificadors existencial i universal de la manera següent:

- Quantificador existencial "some": fa una comprovació de si existeix algun element dintre d'un conjunt que compleixi les regles que li marquem. Ex:
  some $x in (1,2,3) satisfies $x>=2 resposta true

- Quantificador universal "every", que fa una comprovació de si tots els elements d'un conjunt compleixen una propietat marcada. Amb aquest quantificador s'ha de tenir una precaució especial, ja que avalua com a certa una comprovació de node que retorna una seqüència buida. Ex:
  every $x in (1,2,3) satisfies $x>=2 resposta false

4 XUpdate

El llenguatge XUpdate fou creat amb l'objectiu d'ésser un llenguatge per fer actualitzacions sobre documents XML i que, a diferència de la resta que s'han tractat en aquest treball, no és una iniciativa del W3C, sinó d'un projecte del "XML:DB initiative's working group". Els documents XUpdate han d'estar escrits en XML i per tant han de complir les regles de formació comunes a tots els documents XML.

En quant a la seva manera de treballar, estan pensats per treballar d'una manera molt similar a com ho fan quan treballem amb les fulles d'estil i XSLT, ja que és un llenguatge descriptiu que explica les modificacions que s'han de fer sobre determinats elements del document XML. Per tant, per poder funcionar d'aquesta manera fa ús del llenguatge de cerques XPath per trobar els elements sobre els que s'han d'aplicar aquestes modificacions.

Per poder treballar de manera correcta amb XUpdate hem de tenir en compte que té associat un espai de noms i que la seva URI és "http://www.xmldb.org/XUpdate", i el nom del prefix que s'ha de posar a les funcions d'aquest llenguatge és "xupdate".

Els documents XUpdate, al igual que els documents XML han de portar un número de versió que indicarà quina és la versió que es segueix. Actualment la versió que s'utilitza és la 1.0. Tenint això en compte la capçalera d'un document XUpdate quedaria de la següent manera, incloent també la definició del seu espai de noms:

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
```

A continuació descriurem els diferents tipus de funcions que incorpora aquest llenguatge que poden ser per:

- Modificar elements
- Inserir elements
- Afegir elements
- Actualitzar elements
- Eliminar elements
- Canviar el nom d'alguns elements
- Per modificar variables i el seu contingut

4.1 Modificacions

Quan fem una modificació aquesta ha d'anar dintre d'un element xupdate:modifications, que com s'ha vist anteriorment ha de contenir un atribut indicant el número de la versió. A més d'això l'element "modifications" ha de contenir un dels següents elements, depenent del tipus de modificacions a realitzar:

Instrucció	Descripció
`xupdate:insert-before`	Insereix el nou element com a germà abans de l'element que ens serveix de referència
`xupdate:insert-after`	Insereix el nou element com a germà darrera de l'element que ens serveix de referència
`xupdate:append`	Afegeix un nou node
`xupdate:update`	Actualitza un node ja existent
`xupdate:remove`	Elimina un node
`xupdate:rename`	Modifica el nom d'un node
`xupdate:variable`	Defineix una variable que conté una llista de nodes que pot tornar a ser utilitzada en posteriors ocasions
`xupdate:value-of`	Selecciona una variable
`xupdate:if`	Actualització condicional

4.2 Inserció d'elements

Les instruccions d'inserció d'elements insereixen nodes a l'arbre XML resultant. Per utilitzar-les hi ha dos de les anteriors instruccions de modificació que podem utilitzar, "xupdate:insert-before" i "xupdate:insert-before" per especificar a on aniran aquests elements inserits respecte a l'element de referència. Aquestes dues instruccions han de portar un "select" per especificar la instrucció XPath de cerca. Les instruccions d'inserció són les següents:

Les instrucció de modificació han de contenir els diferents tipus d'elements per indicar de quin tipus és el que s'ha d'inserir:

Instrucció	Descripció
`xupdate:element`	Afegeix un nou element
`xupdate:attribute`	Afegeix un nou atribut
`xupdate:text`	Afegeix text
`xupdate:processing-instruction`	Afegeix una nova instrucció de processament
`xupdate:comment`	Afegeix un nou comentari

A mode d'exemple de utilització d'aquestes instruccions posarem un codi que inserirà un nou element a un document ja existent

Codi XML origen	Codi XUpdate	Codi resultant

`<?xml version="1.0"?>`
`<nen>`
`<adreça>`
`<carrer>`
`Font`
`</carrer>`
`<numero>`
`15`
`</numero>`
`</adreça>`
`</nen>`
	`<?xml version="1.0"?>`
	`<xupdate:modifications version="1.0"`
	
	`xmlns:xupdate="http://www.xmldb.org/xupdate">`
	
	`<xupdate:insert-after select="/nen/adreça[1]">`
	`<xupdate:element name="adreça">`
	`<carrer>Raval</carrer>`
	`<numero>22</numero>`
	`</xupdate:element>`
	`</xupdate: insert-after >`
		`<?xml version="1.0"?>`
		`<nen>`
		`<adreça>`
		`<carrer>`
		`Font`
		`</carrer>`
		`<numero>`
		`15`
		`</numero>`
		`</adreça>`
		`<adreça>`
		`<carrer>`
		`Raval`
		`</carrer>`
		`<numero>`
		`22`
		`</numero>`
		`</adreça>`
		`</nen>`

4.3 Afegir elements

Quan utilitzem "`xupdate:append`" estem creant i afegint un node com a fill del node de context. Quan l'utilitzem hem de incloure una clàusula "select" per seleccionar el node de context que es convertirà en pare del nou node que s'està a punt de crear. Un cop seleccionat el node de context podem utilitzar opcionalment diferents atributs per indicar en quina posició s'inserirà aquest nou fill, i en el cas que d'ometre aquesta indicació el nou node s'inserirà darrera de tots els altres fills d'aquest node de context.

Juntament a aquesta instrucció podem afegir els següents tipus d'elements:

Instrucció	Descripció
`xupdate:element`	Afegeix un nou element
`xupdate:attribute`	Afegeix un nou atribut
`xupdate:text`	Afegeix text
`xupdate:processing-instruction`	Afegeix una nova instrucció de processament
`xupdate:comment`	Afegeix un nou comentari

4.4 Actualització d'elements

Això ho farem amb la instrucció "xupdate:update", que amb la seva clàusula "select" incorporada seleccionarà el node a modificar, i com a contingut d'aquest element portarà el nou valor d'aquest node seleccionat. Exemple:

Codi XML origen	Codi XUpdate	Codi resultant
<?xml version="1.0"?> <nen> <adreça> <carrer> Font </carrer> <numero> 15 </numero> </adreça> </nen>	<?xml version="1.0"?> <xupdate:update select="/nen/adreça[1]/numero"> 17 </xupdate:update>	<?xml version="1.0"?> <nen> <adreça> <carrer> Font </carrer> <numero> 17 </numero> </adreça> </nen>

4.5 Eliminació d'elements

La instrucció "xupdate:remove" juntament amb el "select" determina l'element que ha de ser esborrat. Un cop fet això l'element en qüestió ja no apareixerà a l'arbre resultant.

4.6 Canviar el nom d'un element

Això ho farem amb la instrucció "XUpdate:rename" i el seu "select" associat. De manera que un cop seleccionat un element aportem el nou nom que tindrà aquest node i així l'arbre resultant ja mostrarà el nou nom de node.

En el cas que no hi existeixi cap node al que se li hagi de modificar el nom el sistema retornarà un error.

4.7 Variables

Amb la instrucció "XUpdate:variable" afegirem o eliminarem variables associades a elements. Aquesta instrucció, a més de portar un "select" com la resta també porta el nom de l'atribut amb "name" que hem de crear o eliminar. Quan creem una variable se li ha d'afegir el valor que contindrà, que pot ser de objecte de qualsevol dels tipus possible que pot retornar una expressió.

5 BD:XML natives

Quan parlem de BD:XML natives, parlem de sistemes gestors de BD que treballaran directament sobre documents XML i els seus elements en funció de l'estructura interna de cadascun d'aquest documents i del tipus de consulta que es porti a terme.

Anàlogament als sistemes gestors de bases de dades tradicionals, aquests nous sistemes de BD:XML també ens permeten fer consultes sobre les dades emmagatzemades, creuar-les i mostrar-les de la manera que ens sembli, podent així utilitzar posteriorment aquests resultats obtinguts de la manera més convenient.

Una de les diferències a destacar amb els sistemes de BD tradicionals és que aquest nous gestors de BD treballen sobre documents XML, i totes les dades que tracten estan en format XML, a diferència dels SGBDR tradicionals que tenen un ampli ventall de diferents tipus de dades predefinits.

Els sistemes gestors de BD:XML natives que utilitzarem en aquest estudi són **eXist** [http://exist-db.org] i **Xindice** [http://xml.apache.org/Xindice/]. Els dos sistemes de BD han estat implementats en llenguatge java [http://www.sun.com], de manera que permeten una utilització amb independència del sistema operatiu sota el que s'executen.

L'orientació que vull donar per aquest treball és la d'un entorn plenament funcional utilitzant eines de lliure distribució en les que el cost de programari sigui mínim. Per aquest motiu la meva elecció en quant a sistema operatiu ha estat un sistema Linux, en concret la distribució "Suse 9.0 professional", descarregada directament des de Internet. Aquesta distribució, com he pogut comprovar, conté totes les eines necessàries per una correcta instal.lació de les dues BD. En concret les eines utilitzades per fer aquest treball han estat les següents:

- Servidor Web apache, versió 1.3 . Aquest servidor serà el que ens mostrarà les pàgines web amb el contingut dels documents XML i els resultats de les consultes que es puguin realitzar.
- Maquina virtual java, versió 1.4. Aquest component és essencial degut a que ambdues BD estan implementades utilitzant aquest llenguatge i necessiten d'aquest entorn per poder funcionar.
- Contenidor de servlets Tomcat versió 4.1. Aquest contenidor és necessari per poder executar la versió de les dues BD que se distribueixen com a servlets i així poder accedides directament via web.
- Intèrpret de PHP versió 4.3. Aquesta versió de l'intèrpret és plenament funcional amb el servidor web Apache instal.lat. La raó de la seva instal.lació és la creació de pàgines web dinàmiques que s'executin directament en el servidor i que pugin accedir a les dades XML que tenim emmagatzemades a la BD:XML nativa.

Tots aquests components acompanyen la distribució Linux utilitzada i a l'instal.lar-los es configuren per defecte, exceptuant alguns petits ajustos (com per exemple el canvi de port del servidor Tomcat del 8080 al 8081 per evitar problemes durant les proves [NOTA: EXPLICAR PERQUE HA CALGUT CANVIAR EL PORT]).

Un cop iniciat de nou l'ordinador després de la instal.lació (per si de cas, prevenint així que alguna de les variables d'entorn no es carregui correctament) obtenim un servidor web completament funcional per començar amb les proves necessàries d'instal.lació de les dues BD: eXist i Xindice.

5.1 eXist

Aquesta BD es distribueix sota els termes de llicència GNU LGPL (www.gnu.org/copuleft/lesser.html).

5.1.1 Instal.lació de la BD

Aquesta BD, per fer-la funcionar com a servlet dins de Tomcat, la descarreguem en format ".war",(l'arxiu descarregat ha estat "eXist-1.0b2-build-1107.war"). Un cop tenim descarregat l'arxiu el carreguem com a servlet des de la pàgina d'administració del tomcat.

A continuació, i seguint les instruccions d'instal.lació, modifiquem la variable d'entorn "EXIST_HOME" i li assignem com a valor el directori a on ha quedat instal.lat l'eXist. Tot seguit, exportem la variable d'entorn, o iniciem de nou l'usuari.

En el meu cas he fet que el tomcat es carregui automàticament, per tant després d'engegar el sistema el següent cop ja disposem del gestor de BD:XML eXist. El sistema eXist conté un seguit de pàgines web que permet el control de les col.leccions XML, així com crear noves col.leccions i esborrar-ne de velles, tot a través d'un entorn de pàgines web molt amigable i amb una sèrie d'exemples per poder veure com funciona, tot fent consultes sobre els documents de prova.

eXist també porta inclós en aquesta distribució el servidor web jetty, de manera que també podem executar-lo de forma independent sense la necessitat d'utilitzar ni l'Apache ni el Tomcat.

Un cop engegat el servidor eXist, amb els scripts que ja porta definits, podem executar el client de la BD, implementat també en java.. En un primer moment, al accedir al client ens demana un usuari (**admin)** i una contrasenya (cadena buida) per accedir, i a més a més quin és el directori base des del que penjaran les diferents col.leccions. Des del client, de la mateixa manera que des de la interfície web (en mode servlet), podem treballar sobre col.leccions, i disposa també d'una pantalla des de la que es poden fer consultes a les col.leccions instal.lades.

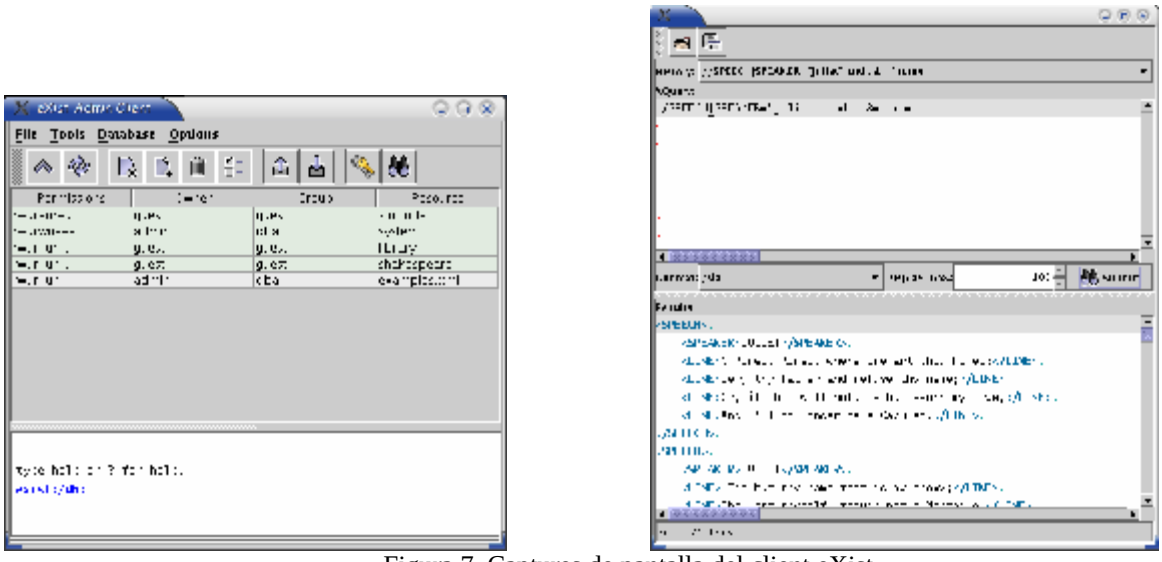

Figura 7. Captures de pantalla del client eXist

5.1.2 Funcionament

El funcionament d'aquesta DB és tal com s'havia esperat, i no mostra complicacions apreciables, cosa que fa que sigui agradable de treballar, tant via web com per la interfície gràfica que l'acompanya, i encara que no està tant treballada com la d'altres programes comercials, el seu funcionament és correcte i proporciona els resultats esperats.

A més per les proves fetes puc dir que s'agraeix tenir una interfície gràfica i no haver de treballar directament des de la línia de comandes, com és el cas de l'altra DB estudiada, Xindice (veure secció 5.2).

5.1.3 Modes de treball amb eXist

Si ens interessa crear les nostres pròpies aplicacions que accedeixin al servidor d'aquesta BD i a les diferents col.leccions que pot contenir ho podem fer de diverses maneres:

a) Creant aplicacions web que treballin amb XQuery.

En aquest cas n'hi ha dos opcions per generar directament pàgines web partint de consultes XQuery.

1.Aplicacions web amb XQueryServlet: En aquest cas eXist, a través d'una adreça web patró que es pot configurar a l'arxiu de configuració "web.xml" i unes pàgines que han de tenir una terminació concreta per que siguin reconegudes , agafa la resta de l'adreça que va darrera del patró i ho executa tractant-lo com una consulta adreçada al servidor de BD, i el resultat obtingut el retorna cap al client que li ha fet la comanda.

Per exemple, en el meu cas, l'adreça patró és "/xquery/", darrera de l'adreça base de la del servlet eXist. Llavors les pàgines que porten les consultes han de tenir una terminació ".xql" per poder ser identificades. Així, per tenir una pàgina web amb consultes que puguin ser reconegudes per eXist seria:
"http://localhost:8081/exist/xquery/exemple.xql".

Aquesta pàgina pot contenir XML i XQuery de manera que abans de ser retornada al client se substitueixen les parts de codi XQuery pel resultat de la consulta realitzada a la BD. Amb això tenim una pàgina web dinàmica que s'actualitza amb les consultes que se li poden fer online.

2.Aplicacions amb XQueryGenerator: Usant un generador com per exemple Cocoon [http://cocoon.apache.org/] que treballa com el servlet, però on el resultat és posteriorment tractat abans de ser retornat al client.

b) Amb l'API web HTTP: REST.

Aquest API proporciona unes opcions de treball que estan directament associades sobre els mètodes de consulta clàssics de les pàgines web. Aquests mètodes són: GET, PUT, DELETE i POST. El seu funcionament és molt semblant a com funcionen aquestes opcions als servidors web.

Aquesta opció de treball és la més senzilla i ràpida d'accedir directament al servidor de la BD eXist, però també és la més restringida de totes. Les consultes que podem fer utilitzant aquest mètode utilitzen com a URL una adreça web que conté un patró d'adreça per localitzar les col.leccions amb que es vol treballar, i a continuació en aquesta mateixa adreça, com opcions, se li afegeixen tant la consulta que es fa com altres opcions, que poden ser:

Opció	Explicació
_xsl=fulla_d'estil_XSL	La fulla d'estil que se li aplicarà al resultat al retornat
_query=expressió XPath/XQuery	L'expressió amb la consulta que es vol fer
_indent=yes|no	Retorna el codi amb tabulacions segons els elements
_encoding=la codificació	La codificació que se li aplicarà al resultat
_howmany=número d'elements	El número d'elements que volem que retorni
_start=punt d'inici	L'element d'inici de retorn
_wrap=yes|no	Retorna el resultat embolicat per un element concret?

c) Escrivint aplicacions java utilitzant l'API XML:DB.

Aquest seria el mètode més complet, encara que també el més complicat degut a que tractaríem amb el API java. Aquesta API conté les funcions per accedir a tots els aspectes de la BD. Per tant podríem entrar directament a la manipulació de les col.leccions i els documents XML. Això significaria fer programes java que fessin les mateixes funcions que la BD.

d) Utilitzant l'API XML-RPC

Aquest API proveeix un mètode simple per cridar procediments remots des de gran quantitat de llenguatges, per exemple PHP, JSP, etc.

e) Utilitzant Web Services: SOAP

SOAP representa una alternativa a la utilització del API XML-RPC, ja que mentre que amb XML-RPC les crides a les funcions s'han de fer completament a mà, amb SOAP moltes de les crides a les funcions de baix nivell ja es generen automàticament. Com a inconvenient de SOAP s'ha de dir que les eines SOAP són més difícils d'utilitzar, ja que estan dins l'àmbit dels Web Services.

En aquest cas, eXist ofereix dos serveis web, un per fer consultes al servidor i demanar documents, i un altre per afegir i eliminar documents i col.leccions.

5.2 Xindice

5.2.1 Instal.lació de la BD

Anàlogament al cas de la BD anterior, a l'hora de fer la instal.lació s'ha optat per fer-ho com a servlet utilitzant Tomcat com a contenidor, de manera que s'actua en igualtat de condicions.

Per aconseguir aquest objectiu des de la pàgina web http://xml.apache.org/xindice/download.cgi cal descarregar l'arxiu "xml-xindice-1.1b4-war.tar.gz". Un cop descarregat l'arxiu cal desempaquetar-lo i obtenir el fitxer amb extensió ".war". Tot seguit, des de la pàgina d'administració del tomcat s'importa aquest arxiu ".war", però el resultat no ha estat l'esperat i no funciona correctament.

Degut a que no funciona automàticament, encara que he seguit correctament les instruccions, s'ha de fer l'extracció dels arxius manualment, obtenint dues carpetes que han de ser guardades dintre del directori "xindice" ubicat a la carpeta "webapps" del tomcat.

El següent pas és afegir a les variables d'entorn la variable "XINDICE_HOME" amb la ruta fins a la carpeta "WEB-INF", que és una de les carpetes que hem guardat al directori "xindice". El pas següent és carregar de nou el tomcat, i a continuació ja podem accedir a aquesta BD.

5.2.2 Funcionament

Aquesta BD, utilitzant-la en mode servlet, es composa de dos parts diferenciades, una línia de comandes des de la que podem realitzar totes les operacions necessàries sobre la BD i després el servlet pròpiament dit que ens permet veure les col.leccions emmagatzemades, via web.

A través de la pàgina web, aquesta BD no porta cap aplicació implementada que ens permeti gestionar-la com era el cas de eXist, de manera que a través de la pàgina web, tal com ja s'ha comentat, només es poden veure les col·leccions.

A l'hora d'utilitzar-la mitjançant la línia de comandes l'únic problema trobat és que l'arxiu "xindice.sh" en principi no apareix com a executable, per tant cal donar-li els permisos per tal que ho sigui. A continuació apareixen una sèrie d'errors a l'executar-lo, en entorn Windows XP. En aquest punt l'única solució trobada ha estat eliminar totes les comprovacions que realitzava el sistema operatiu Windows XP ja que aquestes comprovacions són les que provoquen els errors.

A continuació encara apareix el problema de que al executar l'script l'aplicació fa una cerca de la BD a través del port 8888 que és el que el Xindice porta per defecte, i com que és diferent del port que

escolta el tomcat dispara un error. Per tal de solucionar aquest error li he d'indicar l'adreça i el port sota el que corre el servlet Xindice, i ho faig de la manera següent, en aquest cas per llistar les col.leccions emmagatzemades:

"xindice.sh lc –c xmldb:xindice://localhost:8081/db"

Un cop fet això ja podem treballar des de la línia de comandes sobre les col.leccions i documents i fer consultes sobre ells. Quan es treballa d'aquesta manera tots els resultats són retornats a través de l'intèrpret de comandes des del que s'ha executat la comanda.

Encara que la línia de comandes funciona correctament es troba a faltar una aplicació gràfica des de la que es puguin executar les instruccions i consultes d'una manera més amigable per l'usuari.

De la mateixa manera el treball a través de pàgina web també deixa molt que desitjar i també es troba molt a faltar alguna aplicació web bàsica per accedir a la BD tant de manera local com remota.

Com a conclusió al servlet descarregat puc dir que el seu funcionament és el correcte però no deixa de ser el mínim imprescindible per poder començar a treballar a l'hora de realitzar una aplicació més complexa, ja que no disposa de cap mena d'exemples que puguin ser utilitzats com a base d'implementació.

5.2.3 Modes de treball amb Xindice

Si es volen crear aplicacions que treballin amb Xindice, al igual que amb eXist, ho podem fer de diferents maneres.

a) Utilitzant l'API XML:DB que és l'API utilitzar per crear aplicacions java que accedeixin al servidor de Xindice.

b) A través de l'API XML-RPC de Xindice per quan es creen aplicacions amb llenguatges diferents de java.

c) Amb l'API de java, que seria l'API de més baix nivell, ja que és el mateix amb que es va crear l'aplicació Xindice.

5.3 Conclusió

Després d'haver vist com treballen les dues BD puc dir que eXist porta molt més treball acompanyant aquesta BD que Xindice, que ve escasa d'exemples, de manera que amb la primera disposem de bastant material que es pot utilitzar per veure com tirar endavant una aplicació facilitant així la feina al començament.

Per això a l'hora de crear l'aplicació d'exemple d'aquest projecte crec que és molt més convenient poder aprofitar la feina que ja està feta i per això treballaré amb eXist. Ara només falta veure quin dels modes de treball explicats és més convenient pel treball que s'ha de fer.

6 MPEG-7

6.1 Introducció

Actualment, la producció audiovisual i multimèdia genera un volum de material enorme que corre el perill de perdre's en armaris plens de DVDs. El que fora interessant seria poder cercar informació respecte tot el material anteriorment generat per a usar-lo, modificar-lo, o reciclar-lo per a futures produccions. En aquest sentit, hi ha diferents aproximacions per representar i modelar tota la informació relacionada amb una producció multimèdia, tant les dades (imatges, àudio, etc.) com les seves metadades, que expliquen el que conté la producció.

Si pensem, doncs, en les funcionalitats d'un cercador web actual, com per exemple Google [www.google.com], les cerques es realitzen per paraules, pel contingut de text d'una pàgina, pel nom d'un arxiu, etc. Però si allò que estem cercant és per exemple "una fotografia d'una posta de sol a la platja de Salou amb només dues persones dintre del aigua", la feina es complica, ja que si la fotografia que cerquem no té justament aquest nom segurament ens serà impossible trobar-la.

Aquest és el context a partir del qual es va començar a treballar en aquest estàndard, la necessitat d'un sistema de descripció del contingut audiovisual, que pugui estar emmagatzemat en diferents formats i en diferents parts del món.

6.2 Definició de MPEG-7

Aquest estàndard anomenat "Interfície de Descripció de Contingut Multimedia" (Multimedia Context Description Interface), o el que és el mateix: MPEG-7, té com objectiu substituir altres tècniques i sistemes propietaris utilitzats per portar a terme tasca d'emmagatzematge i localització de contingut multimèdia.

L'estàndard MPEG-7 especifica un conjunt de descriptors que pot ser utilitzat per descriure diferents tipus de informació multimedia, així com la manera de definir altres descriptors i estructures per aquests descriptors (esquemes de descriptors) i les relacions entre ells. A més també ha estandarditzat un llenguatge per especificar els esquemes de descripció, un llenguatge de descripció de definicions (DDL, Description Definition Language). Un cop tenim els descriptors i els esquemes de descriptors, s'associen al material audiovisual que descriuen i d'aquesta manera es poden indexar per realitzar cerques basades en el contingut. El material que es pot descriure utilitzant aquest estàndard és: Fotografies, gràfics, models 3D, àudio, converses, vídeo, així com la informació de com es combinen entre ells.

El llenguatge utilitzat com a DDL per a l'estàndard MPEG-7 ha estat l'XML Schema [veure *apartat 1.5*]. Com que aquest llenguatge no es va dissenyar específicament per continguts audiovisuals, en MPEG-7 s'han realitzat les següents extensions per ajustar-se a aquest nou estàndard:

- Extensions d'estructures: Tipus de dades "array" i "matrix"
- Tipus multimedia, incloent presentacions de audio, video i audiovisuals.
- Tipus de dades enumerats per MimeType, CountryCode,RegionCode, CurrencyCode i CharacterSetCode.
- Gestió i protecció de la propietat intel.lectual (IPMP) per Ds i DSs.

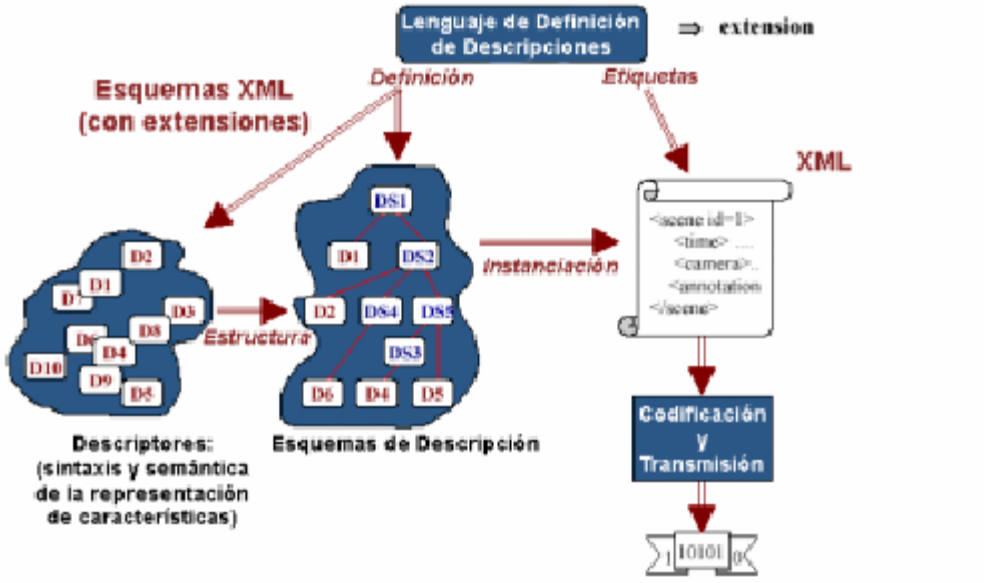

Figura 8. Esquema de creació de documents per MPEG-7 partint del DDL

En quant a la descripció d'algun contingut o algun arxiu, MPEG-7 permet diferents graus de descripció, oferint la possibilitat d'oferir diferents graus de discriminació o perfecció en aquestes descripcions, que pot ser diferent depenent de l'aplicació que l'utilitza o fins i tot de la persona que el crea. Permetent fins i tot crear, per una mateixa font diferents documents en els que aquesta informació té diferents graus d'abstracció. Aquestes descripcions es poden generar en format text (XML) o binari (BiM) o ambdós. Una mateixa fotografia, per exemple, pot ser descrita utilitzant el seu contingut en colors, les formes que la composen, la distribució dels seus elements, etc., on cadascuna d'aquestes descripcions pot anar dintre d'un descriptor diferent, i totes creades segons un esquema de descripció determinat.

La manera d'extreure aquestes descripcions del material audiovisual es pot fer de manera manual o de manera automàtica segons estigui establert per l'aplicació que ho ha de fer, encara que la manera de fer aquesta extracció no entra dintre de l'estàndard MPEG-7, ni tampoc el seu ús final.

A l'hora d'emmagatzemar aquestes descripcions seria convenient que estiguessin acompanyades del material al que descriuen, encara que això no és obligatori i mentre un fotografia, per exemple, estigui en algun servidor d'Internet, i el seu descriptor estigui al meu disc dur. L'únic que hem faria falta en aquest cas seria la localització final de l'arxiu descrit per poder arribar a la informació descrita.

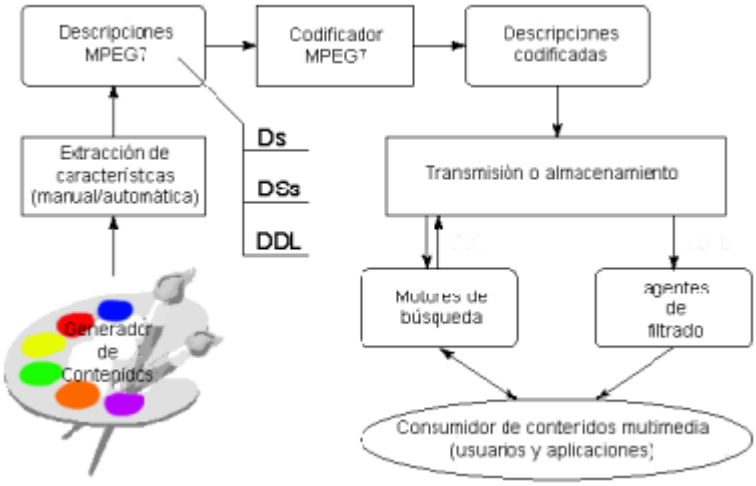

Figura 9. Descripció del procés d'ús de MPEG-7

D'aquest estàndard MPEG-7 podem dir que el seu valor principal està com a servei d'administració i control de material audiovisual, fent pujar alhora el valor d'aquest (ja que el podrem localitzar sempre que el necessitem) , ja que ha de permetre una descripció d'aquest material dividida en varis aspectes diferents (com més profunda sigui i com més punts tinguts en compte millor) que a l'hora se van subdividint per obtenir una descripció total (o tan profunda com vingui marcada per la persona o aplicació que la faci) del seu contingut.

7 Exemples de consultes XQuery per MPEG7

A continuació, un cop instal.lades les eines necessàries, passarem a fer consultes sobre els documents MPEG-7 que tenim instal.lats a la BD eXist. Tots els documents que utilitzaré estan emmagatzemats sota la col.lecció "/db/mpeg7", essent "db" l'arrel de la localització d'emmagatzematge de les col.leccions, i "mpeg7" el nom que se li ha donat a la col.lecció on s'agrupen tots els documents amb que treballarem. Els documents de treball seran les descripcions en fermat MPEG7 de les obres (les tres obres acompanyaran al document d'aquest traball comprimides en format ".zip"):

- AIDA, "opera_aida.xml"
- DON GIOVANNI, "opera_don_giovanni.xml"
- LA TRAVIATA, "opera_la_traviata.xml"

Tots els exemples que a continuació incloc han estat comprovats utilitzant la pàgina de consulta senzilla que acompanya la BD eXist, per tant en principi tots els exemples funcionen de manera correcta segons la versió que tinc instal.lada, indicada anteriorment en aquest mateix document.

7.1 Primer exemple

Per saber els documents en format MPEG7 que tenim emmagatzemats necessitem utilitzar com a caracterització per localitzar-los la seva estructura, de la qual agafem els nodes més externs per trobar-los. Per tant en aquest cas utilitzarem la cadena "/Mpeg7/Description/MultimediaContent/AudioVisual/".

Consulta

```
for $a in /Mpeg7/Description/MultimediaContent/AudioVisual
return
let $b:=doc(document-
uri($a))/Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Title[1]/text()
return
<document>
  <uri>{document-uri($a)}</uri>
  <titol>{$b}</titol>
</document>
```

Resposta

```
<document>
  <uri>/db/mpeg7/opera_aida.xml</uri>
  <titol>AIDA</titol>
</document>
<document>
  <uri>/db/mpeg7/opera_don_giovanni.xml</uri>
  <titol>Don Giovanni</titol>
</document>
```

```
    <document>
      <uri>/db/mpeg7/opera_la_traviata.xml</uri>
      <titol>La Traviata</titol>
    </document>
```

Aquest exemple consta de varies parts. Primer de tot li demanem que trobi, entre els documents emmagatzemats, els que tenen estructura que s'ajusta a l'estàndard MPEG7. Un cop trobats tots aquests documents localitzem la seva posició dintre del conjunt de la base de dades, ja que en principi, si no s'indica cap ruta específica, totes les consultes se fan sobre tots els documents que tenim emmagatzemats. Un cop tenim un document i la seva ruta absoluta ja podem localitzar cadascun dels seus elements. En aquest primer cas hem obtingut el títol de cada òpera. Al final, per cadascun dels resultats obtinguts creem un element de tipus document que conté la ruta absoluta del document i el títol de l'òpera que descriu.

La selecció del títol de l'òpera també es podia haver fet utilitzant la cadena "let $b:=(doc(document-uri($a))/Mpeg7//Title/text())[1]" substituint la que s'ha indicat a l'exemple "let $b:=doc(document-uri($a))/Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Title[1]/text()". L'única diferencia entre les dues maneres d'accedir al mateix node és que en una hem utilitzat la ruta completa fins al node en qüestió i en l'altra li estem indicant que ens retorni el contingut del primer node text que és un descendent del node arrel "Mpeg7" i que té per nom "Title".

7.2 Segon exemple

En aquest segon exemple farem una cerca dels actes en que es divideix una òpera concreta. En aquest cas farem la cerca sobre el document "opera_aida.xml", encara que també és aplicable als altres documents MPEG7 canviant només el títol del document sobre el que volem treballar.

Consulta
for $b at $a in doc ("/db/mpeg7/opera_aida.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/TemporalDecomposition/AudioVisualSegment/@id return <acte><titol pos="{$a}">{$b/attribute()}</titol></acte>
Resposta
<acte> <titol pos="1" id="AID_01_opening"/> </acte> <acte> <titol pos="2" id="AID_01_preludio"/> </acte> <acte> <titol pos="3" id="AID_01_act1"/> </acte> <acte> <titol pos="4" id="AID_01_act2"/> </acte> <acte> <titol pos="5" id="AID_01_act3"/> </acte> <acte> <titol pos="6" id="AID_01_act4"/> </acte>

Aquest cop s'ha utilitzat una variable com a comptador, la variable "$a", que indica la posició que ocupa l'element dintre del conjunt del resultat. La variable que conté els resultats obtinguts pròpiament dits és la "$b".

Al final li estem indicant que ens retorni com a resultat un element anomenat "acte" que conté un altre element anomenat "titol" amb un atribut anomenat "pos" que contindrà el valor del comptador per cadascun dels resultats obtinguts. Com a node text dintre de l'element "titol" li demanem que ens retorni el valor que conté el atribut "id" de l'element "AudioVisualSegment", però com podem veure

al resultat obtingut no ho ha fet com se li demana, sinó que ha interpretat que com és un atribut ho ha de retornar en el mateix format trobat, i per això ens torna el resultat mostrat anteriorment.

Si realment volem obtenir el resultat amb l'estructura que se li està demanat ho hem de fer convertint el node atribut a node text, i per això utilitzem la funció

"string($a as item?) xs:string".

De manera que el resultat final de la consulta serà:

Consulta

```
for $b at $a in doc
("/db/mpeg7/opera_aida.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/TemporalDecomposition/
AudioVisualSegment/@id
return
<acte><titol pos="{$a}">{string($b/attribute())}</titol></acte>
```

Resposta

```
<acte>
   <titol pos="1">AID_01_opening</titol>
</acte><acte>
   <titol pos="2">AID_01_preludio</titol>
</acte><acte>
   <titol pos="3">AID_01_act1</titol>
</acte><acte>
   <titol pos="4">AID_01_act2</titol>
</acte><acte>
   <titol pos="5">AID_01_act3</titol>
</acte><acte>
   <titol pos="6">AID_01_act4</titol>
</acte>
```

Ara ja hem aconseguit el resultat desitjat.

7.3 Tercer exemple

Ara ens interessa obtenir informació sobre els cantants que participen en l'obra. La informació que es pot extreure del document MPEG-7 és el nom i cognom del cantant, el nom i cognom del seu personatge, el tipus de veu que té i una petita definició que acompanya al seu tipus de veu. La informació sobre el seu tipus de veu i la definició es pot aconseguir en diferents idiomes, entre els que estan el català "ca", l'anglès "en", el castellà "es" i l'italià "it". En aquest cas li demanarem tota la informació possible i que tant el nom del seu tipus de veu com la definició volem la que ve en català, i al final que ens ho retorni tot creant un element anomenat cantant que contingui als elements amb la informació cercada.

La consulta que volem seria aquesta:

Consulta

```
for $a in doc
("/db/mpeg7/opera_aida.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Creator
where $a/Role[@href="OpenDramaSinger%"]
order by $a/Agent/Name/FamilyName
return
let $nom := $a/Agent/Name/GivenName/text()
let $cognom := $a/Agent/Name/FamilyName/text()
let $nomPpersonatge:=$a/Character/GivenName/text()
let $cognomPersonatge:=$a/Character/FamilyName/text()
let $veu:=$a/Role/Name[@xml:lang='ca']/text()
let $definicio:=$a/Role/Definition[@xml:lang='ca']/text()
return
<cantant>
```

```
<nom>{$nom, " ", $cognom}</nom>
<personatge> {$nomPpersonatge, " ", $cognomPersonatge}</personatge>
<veu>{$veu," : ",$definicio}</veu>
</cantant>
```

Resposta

```
<cantant>
   <nom>Adina Aaron</nom>
   <personatge>Aida </personatge>
   <veu> : Veu femenina de tessitura mes alta.</veu>
</cantant><cantant>
   <nom>Kate Aldrich</nom>
   <personatge>Amneris </personatge>
   <veu> : Veu femenina amb tessitura entre soprano i contralto.</veu>
</cantant><cantant>
   <nom>Giuseppe Garra</nom>
   <personatge>Amonasro </personatge>
   <veu>Bariton : Voz masculina con una tessitura intermedia entre tenor y bajo.</veu>
</cantant><cantant>
   <nom>Enrico Guiseppe Iori</nom>
   <personatge>Ramfis </personatge>
   <veu>Baix : Veu masculina mes greu.</veu>
</cantant><cantant>
   <nom>Micaela Patriarca</nom>
   <personatge>Sacerdotessa </personatge>
   <veu> : Veu femenina de tessitura mes alta.</veu>
</cantant><cantant>
   <nom>Paolo Pecchioli</nom>
   <personatge>Il Re Dell'Egitto </personatge>
   <veu>Baix : Veu masculina mes greu.</veu>
</cantant><cantant>
   <nom>Scott Piper</nom>
   <personatge>Radames </personatge>
   <veu>Tenor : Veu masculina mes aguda.</veu>
</cantant><cantant>
   <nom>Stefano Pisani</nom>
         <personatge>Un Messagero </personatge>
   <veu>Tenor : Veu masculina mes aguda.</veu>
</cantant>
```

En aquest cas a més he utilitzat la clàusula “order by” de manera que el resultat obtingut ha quedat ordenat per l’element “FamilyName” de l’actor, encara que podíem haver utilitzat qualsevol dels altres possibles valors per fer l’ordenació d’aquest resultat, tot va en funció de quin patró volem seguir.

En el resultat obtingut podem veure que alguns dels valors que hem demanat no apareixen. Aquest buits corresponen a aquells valors pels que demanàvem un resultat en català, ja que en aquest cas no hi deuen d’existir en aquest idioma, encara que si en altres dels possibles.

7.4 Quart exemple

Ara ampliarem l’exemple anterior en el que havíem obtingut el títol des diferents actes, i ho farem afegint informació sobre el temps que dura cadascun d’aquest actes.

Per fer-ho utilitzarem les funcions de la biblioteca “fn:minutes-from-duration($a as xdt:dayTimeDuration?) xs:integer?” que donat un temps que ve en el format “dayTimeDuration” ens retorna els minuts, i la funció “fn:seconds-from-duration($a as xdt:dayTimeDuration?) xs:decimal?” que fa lo mateix però retornant els segons.

Per fer que el resultat obtingut sigui més diferent de l'anterior agafarem una altra òpera, de manera que el resultat obtingut és el següent:

Consulta

```
for $b at $a in
doc("/db/mpeg7/opera_la_traviata.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/TemporalDecomposition/
AudioVisualSegment
return
<acte>
<titol pos="{$a}">{string($b/@id)}</titol>
<durada>{$b/MediaTime/MediaDuration/fn:minutes-from-duration( text() ), string("minuts"),
$b/MediaTime/MediaDuration/fn:seconds-from-duration( text() ), string("segons")}</durada>
</acte>
```

Resposta

```
<acte>
   <titol pos="1">Preludio</titol>
   <durada>4 minuts 26.0 segons</durada>
</acte><acte>
   <titol pos="2">act1</titol>
   <durada>27 minuts 10.0 segons</durada>
</acte><acte>
   <titol pos="3">act2</titol>
   <durada>3 minuts 6.0 segons</durada>
</acte><acte>
   <titol pos="4">act3</titol>
   <durada>27 minuts 5.0 segons</durada>
</acte>
```

Obtenint el resultat esperat que ens mostra els actes en que es descompon l'òpera "la Traviata" i el temps total que dura cada acte.

7.5 Cinquè exemple

A l'hora de fer cerques també ens pot interessar fer una cerca conjunta sobre varis documents de manera que puguem extreure el mateix tipus d'elements de tots d'una sola vegada sense haver-ho de repetir individualment per cadascun d'ells. Per això podem utilitzar "document" per "doc" alhora de definir els documents, ja que "document" admet la definició de varis documents diferents d'una sola vegada, de manera que farem la mateixa consulta a tots els documents especificats un rera l'altre i els resultats obtinguts s'afegiran al conjunt total, de manera que obtenim així la suma de resultats que obtindríem si haguéssim fet les consultes per separat.

En el següent exemple s'han volgut trobar els cantants amb una veu del tipus "Baix" de una sèrie de òperes. La cerca és fa amb una sola consulta en la que s'especifiquen tots els documents que volem consultar.

Com que un cop feta la consulta obtenim un conjunt de resultats sense cap diferència entre ells que especifiqui de quin document provenen utilitzo la funció de llibreria "fn:document-uri($a as node) xs:string?" que rebent com a paràmetre un node ens retorna com a resultat el nom i la ruta del document que el conté. Així dintre del mateix resultat ja tindrem el nom de l'òpera a la que intervé.

La consulta ens queda de la manera següent:

Consulta

```
for $a in
document("/db/mpeg7/opera_aida.xml","/db/mpeg7/opera_don_giovanni.xml","/db/mpeg7/opera_la_traviata.xml")/Mpeg7/
Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Creator
where $a/Role/Name/text()='Baix'
return
let $c:=$a/Agent/Name/GivenName/text()
let $d:=$a/Agent/Name/FamilyName/text()
```

```
let $e:=document-uri($a)
let $f:=doc($e)/Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Title
return
<cantant>
<baix>{$c," ",$d}</baix>
<opera>{($f/text())[1]}</opera>
</cantant>
```

Resposta

```
<cantant>
   <baix>Paolo Pecchioli</baix>
   <opera>AIDA</opera>
</cantant><cantant>
   <baix>Enrico Guiseppe Iori</baix>
   <opera>AIDA</opera>
</cantant><cantant>
   <baix>Felice Ponziani</baix>
   <opera>Don Giovanni</opera>
</cantant><cantant>
   <baix>Giuseppe Lolli</baix>
   <opera>Don Giovanni</opera>
</cantant><cantant>
   <baix>Mario Zorgniotti</baix>
   <opera>La Traviata</opera>
</cantant>
```

7.6 Sisè exemple

En aquest exemple volem saber el títol i el compositor de les òperes en les que hi consti el compositor. La consulta queda de la manera següent:

Consulta

```
for $a in /Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Creator
let $b := $a
where $a/Role/@href="OpenDramaRoleCS:authors:composer"
return
for $b in $a/..
let $nom:=$a/Agent/Name/GivenName/text()
let $cognom:=$a/Agent/Name/FamilyName/text()
return
<opera>
<titol>{($b/Title/text())[1]}</titol>
<compositor>{$nom," ",$cognom}</compositor>
</opera>
```

Resposta

```
<opera>
<titol>AIDA</titol>
<compositor>Guissepe Verdi</compositor>
</opera>
<opera>
<titol>Don Giovanni</titol>
<compositor>Wolfgang Amadeus Mozart</compositor>
</opera>
```

7.7 Setè exemple

En aquest últim exemple ens interessa saber el títol i el cantant de les àries de l'òpera Aida. La consulta que ens retornarà el resultat esperat serà:

Consulta

```
for $a in
doc("/db/mpeg7/opera_aida.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/TemporalDecomposition/
AudioVisualSegment/TemporalDecomposition/AudioVisualSegment/TemporalDecomposition/AudioVisualSegment
let $b := $a/CreationInformation/Creation/Title
let $c := $a/CreationInformation/Creation/Creator/Agent/Name
where $a//Name = "Aria"
return
<aria>
<titol>{$b/text()}</titol>
<cantant>{$c/GivenName/text()," ",$c/FamilyName/text()}</cantant>
</aria>
```

Resposta

```
<aria>
   <titol>Se quel guerrier io fossi!</titol>
   <cantant>Scott Piper</cantant>
</aria><aria>
   <titol>Ritorna vincitor!</titol>
   <cantant>Adina Aaron</cantant>
</aria><aria>
   <titol>Qui Radames verra...</titol>
   <cantant>Adina Aaron</cantant>
</aria><aria>
   <titol>Gia i Sacerdoti adunansi</titol>
   <cantant>KateScott AldrichPiper</cantant>
</aria><aria>
   <titol>O terra, addio</titol>
   <cantant>ScottAdinaKate PiperAaronAldrich</cantant>
</aria>
```

8 Aplicació web de consulta de documents MPEG-7

Com a cas d'ús de tot el que s'ha tractat fins ara, toca fer una aplicació senzilla, que permeti fer cerques sobre documents MPEG-7 emmagatzemats a una base de dades XML nativa. La BD escollida ha estat eXist, ja que ha mostrat un funcionament molt bo en totes les proves que s'han realitzat.

L'aplicació en concret que s'ha creat no és molt completa degut al poc temps del que s'ha disposat per la seva creació, però crec que amb tot i això dona una idea bastant clara de com totes les tecnologies esmentades fins ara poden treballar juntes.

El sistema consta de dos parts:

- Un ordinador que fa de servidor (AMD 550 Mhz) en el que tinc instal.lada una distribució Linux, concretament la "Suse 9.0 Professional" (s'ha de dir que el seu funcionament ha estat molt bo per totes les proves que he realitzat). En aquest ordinador, que és el que realment ha suportat tot el treball de programació, s'han instal.lat totes les eines necessàries per fer aquest treball, que han estat:
  - El interpret de PHP que acompanya aquesta distribució.
  - El servidor web Apache que acompanya aquesta distribució.
  - Diversos editors que acompanyen aquesta distribució utilitzats per crear les pàgines web necessàries.
  - El contenidor de servlets Tomcat que acompanya aquesta distribució.
  - La BD eXist en la seva versió de treball com a servlet servida per Tomcat, concretament la versió "eXist-1.0b2-build-1107.war".
- Un altre ordinador, amb sistema operatiu Windows XP, utilitzat com a client des del que es feien les consultes via web i amb el que s'ha escrit aquesta memòria.

Els documents en format MPEG-7 han estat subministrats pel consultor de l'assignatura per tal de disposar de material sobre el que poder treballar.

La part de les consultes a la BD s'ha fet mitjançant serveis web i SOAP.

8.1 Components de l'aplicació

Aquesta aplicació consta de diverses parts que estan definides per les diferents pantalles pel treball interactiu amb la base de dades i els documents que conté. Les diferents parts de que es composa són:

- Una pantalla d'inici en la que es mostra tant el títol de l'aplicació, el meu nom, el nom del consultor encarregat d'aquest TFC i els estudis als quals correspon.
- Una pantalla que fa les funcions d'índex i que ens dona accés a la resta de parts de l'aplicació.
- Una pantalla de consulta amb un quadre de text des de la que podem introduir les nostres consultes a mà.
- Una pantalla de consulta per localitzar informació sobre els cantants i els personatges que participen en les òperes.
- Una pantalla des de la que es poden fer consultes sobre les òperes senceres o sobre els actes que les composen.
- Una pàgina amb enllaços d'interès sobre tot lo tractat en aquest treball.

A continuació passem a mostrar captures de pantalla a on es poden veure les finestres de treball que composen l'aplicació.

8.1.1 Pantalla d'inici

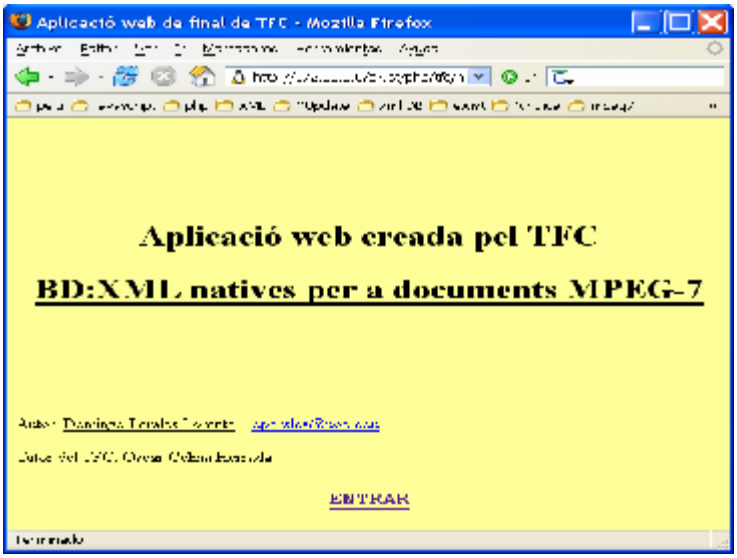

Figura 10. Pantalla d'inici de l'aplicació web

Aquesta pantalla ha estat creada utilitzant un document ".xml" ("menu_tfc.xml") associat a una fulla d'estil ".xsl" ("menu.xsl"), de manera que la primera conté la informació que es pot modificar i la segona l'estil que s'ha d'aplicar als diferents elements del document.

El fet d'haver creat aquesta pàgina com un document XML associat a una fulla d'estil i no com directament com una pàgina web ha estat per que es pugui comprovar, a mena d'exemple, el resultat obtingut.

8.1.2 Pantalla d'índex

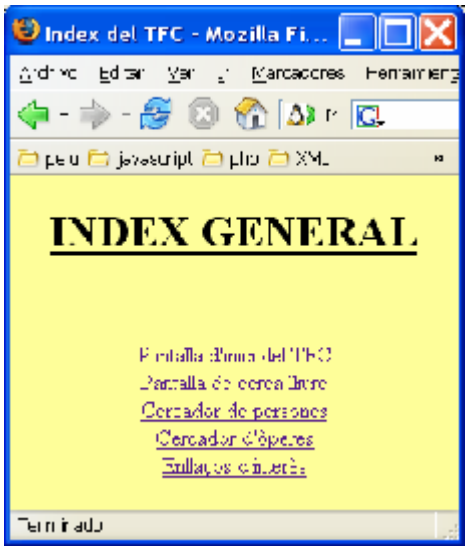

Figura 11. Pantalla d'índex general de l'aplicació

Aquesta pantalla d'índex. al igual que la pàgina anterior també està composta per una pàgina ".xml" ("index.xml"), i una fulla d'estil ".xsl" ("index.xsl") associada.

8.1.3 Pantalla de cerca lliure

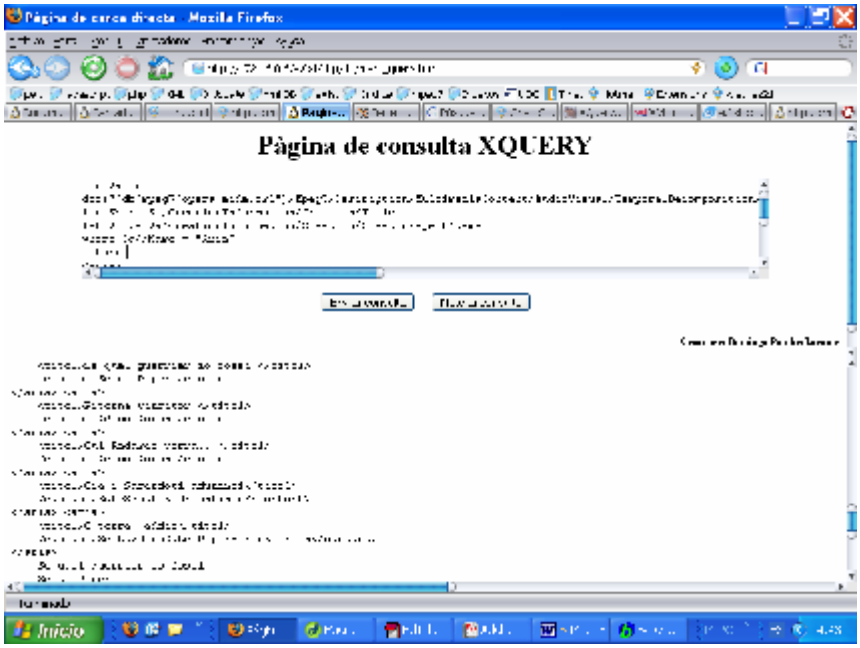

Figura 12. Pantalla de cerca lliure

Aquesta pantalla de treball realment està composta de dues parts.

- La part superior que conté un formulari de consulta amb un quadre de text en el que podem escriure les consultes per la BD eXist que vulguem. Dos botons, un per enviar les dades del formulari i un altre per esborrar el contingut del quadre de text.
- La part inferior és on apareixen els resultats obtinguts de les consultes fetes a la base de dades. Cada vegada que es fa una nova consulta els nous resultats esborren el resultat anterior de manera que només ens queden els últims que hem obtingut.

De fet aquesta és la pantalla de consulta que més he utilitzat a l'hora de crear consultes i fer proves a la BD.

8.1.4 Pantalla de cerca de persones i personatges

Figura 13. Pantalla de cerca de persones i personatges

Aquesta pantalla de consultes de persones i personatges de les diferents òperes ens permet fer consultes diverses a través de la introducció del nom de la persona cercada o personatge i la seva ocupació i fer ordenacions diferents en funció de l'ordre amb el que vulguem obtenir els resultats. A més també podem seleccionar diferents idiomes per aconseguir diferents resultats sempre i quan el document disposi d'informació en el llenguatge seleccionat.

En aquesta pantalla haurem de seleccionar si la consulta que volem fer va dirigida cap a un cantant, un personatge o a la resta del personal que ha treballat, podent especificar la seva ocupació dintre del conjunt de l'obra.

Les cerques es poden fer per cantants, personatges o resta de personal. En funció de l'opció que seleccionem es tindran en compte les seleccions de la fila corresponent de la taula (separades en diferents colors), sense tenir en compte les dades que hi hagin a les altres files.

8.1.5 Pantalla de cerca d'òperes i d'actes

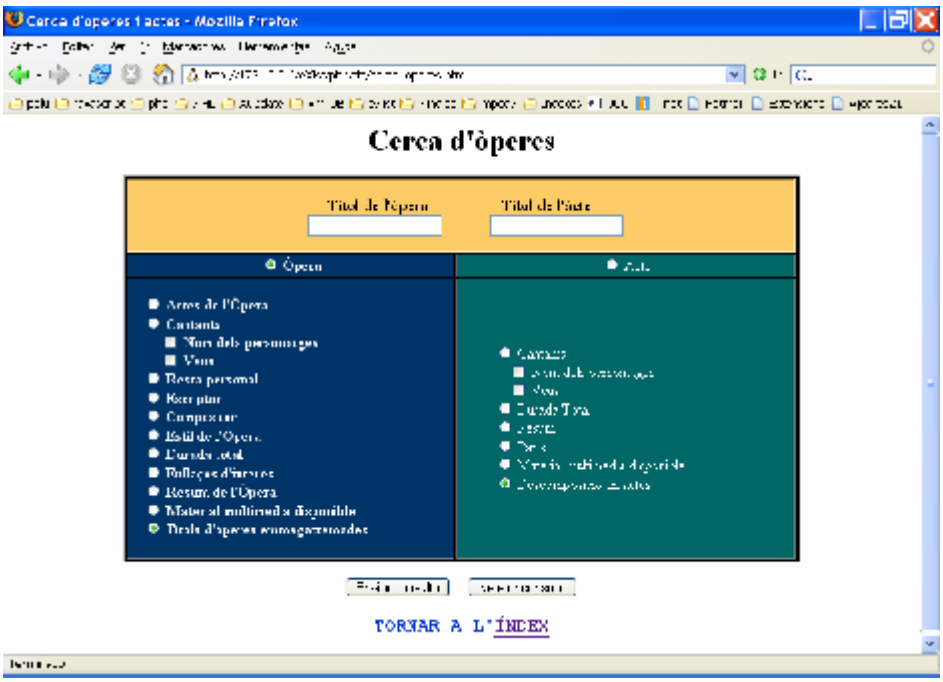

Figura 14. Pantalla de cerca d'òperes i actes

Aquesta captura mostra el formulari de consulta per fer una sèrie de cerques que ja venen definides.

Es divideix en dues parts, que correspondrien a consultes que es poden fer sobre el text general dels documents MPEG-7 i que fan referència a les òperes emmagatzemades. Com es pot veure cadascuna de les dues parts conté una sèrie de cerques que l'únic que necessiten és que se li introdueixi o el nom d'una òpera o el nom d'un acte per accedir a la seva informació.

Tant en aquesta pantalla com les altres, quan trobem un camp de text el seu funcionament és el següent:

- Si l'omplim les cerques es faran en funció de les dades que li introduïm.
- Si no l'omplin els camps de text, les cerques que es fan consideren aquesta dada genèrica, de manera que si el que volem cercar són cantants i no indiquem cap òpera, la cerca es farà entre tots els documents MPEG7 que continguin cantants.

Quan seleccionem l'apartat d'òperes es tindrà en compte el nom de l'òpera a l'hora de cercar en un document específic, en cas contrari la cerca es farà per totes les òperes. Quan seleccionem l'apartat d'actes també tindrà en compte el nom de l'acte sobre el que volem que es faci la cerca.

En l'apartat dels actes tenim opcions per accedir a consultes sobre els diferents actes que composen una òpera en concret. Com que aquestes opcions són prou clares no donarem més explicacions sobre elles.

8.1.6 Pantalla amb adreces d'interès

Figura 15. Pantalla amb adreces d'interès

Aquesta pantalla de treball segueix els mateixos passos que les altres pàgines que utilitzen un arxiu amb dades ".xml" (links.xml") i una fulla d'estil associada ".xsl" ("links.xsl").

El seu contingut són adreces d'Internet que m'han estat molt útils a l'hora de recopilar dades e informació necessàries per poder portar a terme aquest treball i el seu programari associat.

Per tal de poder comprovar el funcionament de l'aplicació, aquesta estarà funcionant a la següent adreça web: "http://80.24.18.102/eXist/php/tfc/index.xml".

9 Conclusions

Com a conclusió per a aquest treball voldria dir que ho he trobat molt interessant degut al fet que he pogut fer una investigació i recopilació d'informació que fa tan sols uns anys haurien estat molt més complicades, ja que mitjançant l'ús de les noves tecnologies com són Internet i tot allò que l'envolta, aquestes cerques d'informació es tornen molt més senzilles que si les hagués hagut de fer anant a biblioteques o cercant llibres que continguéssin la informació que he necessitat.

Si que hi ha hagut moments que un punt concret m'ha costat una mica més de trobar degut a la novetat d'algunes d'aquestes noves tecnologies, com per exemple informació correcta d'instal.lació de les bases de dades tractades, però en general puc dir que la informació necessària per aquest treball està accessible a Internet, esperant que algú la demani i la sàpiga utilitzar.

Una de les dificultats que he trobat sovint ha estat que la major part de la informació està en anglès, cosa que si en principi no és un punt decisiu, si que fa que es trobi a faltar molta més documentació en català o en castellà, sobretot si tenim en compte que el castellà, o l'espanyol com li solen dir, està tant estès arreu del planeta.

També vull recalcar la bona impressió que m'han causat totes les noves tecnologies estudiades, ja que partint d'una base accessible per tothom com és el format ".txt", es poden construir documents tan complexos i que siguin acceptats per tants tipus d'aplicacions, i sobretot pel fet que aquesta informació, pel fet d'estar en format text, sempre pot ser accedida i fins i tot llegida per conèixer el seu contingut directament sense necessitat de cap tipus de programari específic. Aquest és un punt al meu entendre molt important, ja que facilita molt l'intercanvi d'informació entre aplicacions diferents i fins i tot entre sistemes operatius diferents, cosa que acosta molt més al mon en quant a intercanvi de dades i ens allibera una mica de les grans companyies de creació de software, ja que al ser un format no propietari sempre dona molt més joc per tothom que vulgui crear o llegir documents, seguint les pautes marcades per aquest estàndard XML.

A part d'això, utilitzant el conjunt d'eines que formen aquestes noves tecnologies es poden crear aplicacions realment potents que es poden equiparar a d'altres molt costoses, de manera que iguala una mica la balança a l'hora de crear software, fent que el software lliure tingui una mica més de camí lliure per endavant.

Un altre punt que m'ha semblat molt interessant i que no coneixia era l'estàndard MPEG-7 i el seu intent de crear descripcions de material multimedia. Aquestes noves tecnologies crec que són un pas lògic degut a les revolucions constants per part de les empreses que creen maquinari per enregistrar material audiovisual, ja que la velocitat en què es genera aquest material és tant gran que estem corrent el risc de perdre gran part d'aquest pel simple fet de no poder tenir una correcta documentació i informació de tot el material enregistrat.

Per tant, si a l'estàndard MPEG-7 se li dona una aplicació correcta podrem tenir a l'abast de la mà material que ara mateix se ens comença a escapar, i permetrà que se li donin una sèrie d'usos que potser ara mateix no arribem a imaginar, o permetent que a partir d'ell puguin néixer d'altres que portin l'emmagatzematge de dades relacionades amb el material multimèdia i audiovisual, en formats molt diferents a punts d'accessibilitat que avui en dia serien impossibles d'imaginar.

En definitiva, voldria dir que aquest treball ha estat molt instructiu i que m'ha fet aprendre, encara més, la manera de cercar informació i com donar-li un ús útil a l'hora de crear una aplicació i instruir-me sobre un tema concret.

10 Enllaços d'interès:

- World Wide Web Consortium, "W3C": http://www.w3.org/
- Xquery 1.0 and XPath 2.0 Functions and Operators: http://www.w3.org/tr/xquery-operators/
- XUpdate i el seu DTD associat: http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html
- XML:DB XUpdate Use Cases: http://www.xmldatabases.org/projects/XUpdate-UseCases/
- BD eXist: http://exist.sourceforge.net/
- DB Xindice: http://xml.apache.org/Xindice/
- MPEG-7 Consortium: http://mpeg7.nist.gov/mp7c/meeting_2.html
- Tutotrial sobre XSLT: http://geneura.ugr.es/~jmerelo/XSLT/
- Opendrama project: http://www.iua.upf.es/mtg/opendrama/
- MPEG-7 Description Definition Language (DDL): http://archive.dstc.edu.au/mpeg7-ddl/
- PerfectXML: http://www.perfectxml.com/articles.asp
- PHP en castellà: http://es2.php.net/manual/es/
- XML-ES: tutorials sobre XML: http://www.it.uc3m.es/~xml/indice.html
- Tota la informació i més...: http://www.google.es/advanced_search?hl=es