Títol

BD:XML natives per a documents MPEG-7

Con formato

Estudiant

Ginés Fernàndez Alacid

Con formato

Titulació

Enginyeria tècnica en Informàtica de Sistemes

Consultor

Òscar Celma Herrada

Data

Juny de 2004

ÍNDEX

Taula de figures.

Resum Inicial

El treball de fi de carrera que proposem a continuació, és un intent de apropar-nos a la tecnologia XML en la seva vessant de contenidor de dades estructurades, per posteriorment, poder recuperar-les, modificar-les o realitzar altres tractaments oportuns.
És per això que ens plantegem la necessitat d'estudiar a fons la tecnologia XML, amb totes les seves virtuts i defectes(esperem que pocs), per entendre com s'estructuren aquestes dades en els documents i que, posteriorment, voldrem recuperar.
Entre d'altres facetes del metallenguatge XML, trobarem les regles bàsiques que tot document d'aquest tipus ha de complir, les diferents alternatives d'estructuració de les dades, com són els DDT i els Esquemes, i, en concret, dins d'aquest últim ens decantarem per l'estudi del esquema per a documents MPEG-7, per continguts Multimèdia.

També voldrem apropar-nos als diferents gestors de bases de dades contingudes en documents XML i n'estudiarem en profunditat dos d'ells, eXist i Xindice, per posteriorment, intentar comprendre el funcionament d'aquest programes utilitzant els llenguatges de consulta XPath (versions 1 i 2.0) i XQuery, que està basat en els anteriors.

Realitzarem casos pràctics i analitzarem els resultats per veure si eren els esperats, per, finalitzar amb les conclusions típiques d'aquests tipus de treballs.

Moltes Gràcies a tots els que m'han ajudat.

1. Descripció i objectius del projecte

1.1. Descripció

El projecte es basa en avaluar diferents BD:XML natives per a desar i recuperar documents XML basats en l'estàndard MPEG-7. L'estàndard proposa un llenguatge per a descriure el contingut (metadades) de documents Multimèdia; és a dir àudio i vídeo. El format de representació de l'estàndard MPEG-7 és XML i es basa en un esquema ja predefinit (XMLSchema d'MPEG-7).
L'objectiu del projecte, doncs, és avaluar diferents BD:XML natives (com per exemple *eXist* i *XIndice*) que permetin desar i recuperar metadades de documents Multimèdia descrits amb l'estàndard MPEG-7. Caldrà investigar els avantatges i inconvenients dels diferents sistemes gestors de BD així l'ús dels diferents llenguatges de consulta XML actuals: XPath i XQuery.

1.2. Objectius

- Conèixer l'estructura i organització de les BD:XML natives.
- Avaluar l'adequació d'utilització de BD:XML per a desar i recuperar descripcions basades en MPEG-7.
- Realització de casos pràctics d'aquestes tecnologies, implementant un sistema que permeti fer cerques sobre documents MPEG-7.

2. La recerca

La primera tasca a realitzar consisteix en fer una recerca de les diferents tecnologies involucrades en el projecte: XML, Schema MPEG-7, eXist, Xindice, XPath, XQuery…..

S'ha mirat de trobar la major part de la informació en Català i Castellà, però, en algunes cerques els resultats han estat molt pobres i s'ha hagut d'ampliar la cerca a pàgines d'altres llengües, com Anglès. Aquest fet s'ha vist especialment accentuat pel cas de la documentació de bases de dades XML (eXist i Xindice).

2.1. Recerca d'XML

La recerca d'XML ha estat la més fructífera de totes. Hi ha innumerables pàgines a Internet i prou llibres com per obtenir, de sobres, material amb el qual poder començar a treballar el projecte.

D'altra banda, també he constatat que, si bé hi ha molta informació d'XML, la major part d'aquesta informació no fa referència directa a la vessant d'XML per bases de dades, encara que, això no ha suposat cap problema per trobar material suficient.

De tota aquesta informació s'ha recopilat la major part de la que feia referència a Bases de dades Natives per XML i ens ha confirmat la presència de molta documentació.

2.2. Recerca de "Schema Mpeg-7"

Ha estat una de les cerques més complicades de totes. No es va poder trobar cap referència en castellà o català sobre aquest *Schema* concret, encara que sí es va poder trobar alguna cosa sobre els *Schemes XML* en general. De totes maneres, ampliant la recerca es va poder recopilar suficient informació en Anglès per poder treballar en el projecte.

2.3. Recerca de bases de dades natives eXist i Xindice

Per una banda es va poder trobar bastant informació sobre aquestes dues bases de dades natives XML amb la que poder desenvolupar el projecte i, per altra banda, s'ha pogut descarregar i instal.lar les darreres versions d'aquests programes en versió windows.

2.4. Recerca de llenguatges de consulta per bases de dades natives XML (XQuery, XPath..)

També ha estat possible trobar suficient informació sobre diversos llenguatges de consulta per bases de dades natives com XQuery i XPath

2.5. Conclusions sobre les recerques de dades

Cal dir que totes les recerques efectuades (la major part a través d'Internet), s'han fet sense seguir cap criteri específic i, ha estat, posteriorment, en una segona fase d'avaluació i tria de les cerques on s'ha procedit a descartar material recopilat per ser, o bé redundant, o bé per no tenir cap interès pel nostre projecte.

No és descartable, en cap fase del projecte, tornar a efectuar cerques de nou material que ens pugui ser útil per la realització d'alguna tasca.

3. Instal·lació de sistemes

Després d'analitzar les bases de dades referides en l'apartat de recerca s'ha decidit instal·lar els gestors eXist i Xindice com a bases de dades natives per l'estudi.

3.1. eXist

S'ha optat per instal·lar la versió **eXist-1.0b1**, que es tracta d'una versió autoinstalable en format **.jar** per Java.
El programa ha estat descarregat de la pàgina principal de eXist[1]

Fins el moment no s'ha detectat cap problema ni a la instal·lació ni a les primeres proves que s'hi han efectuat.

3.2. Xindice

En aquest cas, s'ha escollit la versió **xml-Xindice 1.1b3 per windows**, que encara que es tracta d'una versió beta, s'ha mostrat molt estable, tant en la instal·lació com en el posterior funcionament de les primeres proves efectuades.

També s'ha obtingut l'arxiu corresponent a la versió **xml-Xindice 1.0** que es tracta de la versió anterior, però més estable, per si l'anterior versió mostrés signes d'inestabilitat. De moment no ha calgut instal·lar-la.

La descàrrega s'ha efectuat des de la pàgina oficial d'Apache[2]

Tampoc s'ha detectat cap mena de problema, ni a la instal·lació ni en les primeres preses de contacte amb el gestor.

4. XML

4.1. Què és XML?

XML vol dir **Llenguatge de Marques Extensible (Extensible Markup Language).** Va ser creat pel Consorci de la World Wide Web (W3C)[3] per tal de superar les limitacions d'HTML, el llenguatge de Marques d'Hipertext que és la base de les pàgines Web.

HTML és el llenguatge de marques amb més èxit del moment, lògicament, degut al seu ús per a la web. Pot interpretar les etiquetes més simples en quasi qualsevol dispositiu, des de dispositius palmtops fins a els mainframes, i fins i tot, hi ha eines que poden convertir HTML en veu sintètica o en altres formats digitals. Llavors, per què la W3C[3] va crear XML?. Per respondre aquesta pregunta, veurem un exemple:

```
<p>
  <b>
    Joan Carlet Saura Gomis
  </b>
  <br>
  carrer de les Oliveres 332
  <br>
  Barcelona, 08777
</p>
```

Figura 1. Exemple de part d'un document en HTML

El problema d'HTML és que va ser dissenyat per la forma de pensar del humans. Fins i tot sense veure l'exemple anterior en un navegador podem imaginar que es tracta d'una adreça de correu i encara que algú no estigui familiaritzat amb les adreces de correu postal, podria imaginar que és una adreça de Barcelona.

Els humans tenim la intel·ligència per comprendre el significat i la intenció de molts documents, però una màquina, desafortunadament no ho pot fer. Les etiquetes d'aquest document li diuen al navegador com ha de mostrar la informació, però no l'informen de "què significa" aquesta informació. Nosaltres sabem que és una adreça postal però la màquina no.

Veiem ara un document XML d'exemple. Amb XML, es pot assignar un significat a les etiquetes i donar una estructura al document i, per una màquina, és mes fàcil processar la informació.
Es pot extreure el carrer d'un document localitzant el contingut envoltat per les etiquetes **<carrer>** i **</carrer>,** tècnicament conegut com l'element **<carrer>.**

```
<adreça>
        <nom-complet>
                <nom> Joan Carlet  </nom>
                <cognoms> Saura Gomis </cognoms>
        </nom-complet >
        <carrer>
                carrer de les Oliveres 332
        </carrer>
        <ciutat>
                Barcelona
        </ciutat>
        <codi-postal bcn="08777" />
</adreça>
```

Figura 2. Exemple de document en XML

4.2. Etiquetes, elements i atributs

Hi ha tres elements que descriuen un document XML: les etiquetes, els elements i els atributs. Tornem a l'exemple:

- Una etiqueta és un text entre el símbol menor que (**<**) i el símbol major que (**>**). Hi ha etiquetes d'inici **(<nom>)** i etiquetes finalitzadores **(</nom>).**
- Un element està format per una etiqueta d'inici, una etiqueta finalitzadora i tot allò que hi hagi entre elles. A l'exemple anterior, l'element **<nom-complet>** conté dos elements : **<nom>** i **<cognoms>.**
- Un atribut és una tupla nom-valor dins d'una etiqueta d'inici d'un element. En el nostre exemple bcn és un atribut de l'element **<codi-postal>** i té un valor, que és 08777.

4.3. Diferències bàsiques entre XML i HTML

Ara que hem introduït la base del XML caldria fer una recapitulació tot comparant-lo amb l'estàndar HTML:

- HTML només formateja dades i les etiquetes s'utilitzen exclusivament per això, és a dir, les etiquetes informen al navegador com ha de mostrar la informació. XML estructura la informació que pretén emmagatzemar. L'estructura la marca la lògica pròpia de la informació.

- El desenvolupament de HTML va estar marcat per la competència entre els diferents navegadors del mercat. Cada navegador proposava etiquetes noves que, posteriorment, entraven a formar part del estàndard del W3C[3], com per exemple el cas de l'etiqueta <FRAME>. En canvi, el desenvolupament de XML s'està portant a terme amb rigor, ajustant-se a l'estàndard del W3C[3], i amb més diligència que les empreses amb interessos particulars.

- Processar la informació en HTML és inviable, per estar barrejada amb estils i les etiquetes que formatejen la informació. El problema és que el disseny (o format) i el contingut estan clarament solapats. XML pot processar la informació amb molta facilitat perquè tot està endreçat d'una manera lògica i el formateig de la informació per a què sigui entenible per l'usuari és viable a través de fulls d'estil o similars.

4.4. Avantatges pròpies d'XML

- XML simplifica l'intercanvi de dades. Degut a que diferents organitzacions (fins i tot entre diferents òrgans d'una mateixa organització) rarament utilitzen un únic conjunt d'eines i formats, pot suposar una gran quantitat de feina per comunicar aplicacions. Utilitzant XML ens estalviem aquesta feinada només transformat els formats natius de les aplicacions en XML i al inrevés, per tant XML ajuda clarament a la interoperabilitat entre aplicacions..
- XML permet un codi més lleuger. Els documents XML poden ser estructurats per identificar cada part important d'informació (i també les relacions entre aquestes parts.
- XML permet cerques ràpides i eficients. Encara que els motors de recerca han millorat molt els últims anys, encara, avui dia, ens trobem infinitat de resultats erronis en una cerca. Si estem cercant informació sobre algú que es diu "*Lluís Granada*" en documents HTML ens podem trobar informació de la ciutat de Granada, de la fruita Granada i altra informació irrellevant pels nostres interessos. Si la recerca s'hagués efectuat sobre documents XML en elements <nom> que continguin el text Granada probablement s'obtindrien millors resultats.

4.5. Regles dels documents XML

Molts analitzadors (browsers) HTML accepten marcats sense regles ni control i fan conjectures al voltant del text llegit. Per evitar el garbuix d'estructures trobades en documents HTML, els creadors d'XML van decidir reforçar l'estructura del document de bon començament

4.6. Documents invàlids, vàlids i ben formats

Tenim tres tipus de documents XML:

4.6.1. Documents invàlids

Són aquells que no segueixen les regles de sintaxi definides per la especificació XML. Si un desenvolupador té unes regles definides del què aquest document pot contenir (ja sigui con a DTD o com Esquema, que més endavant definirem i analitzarem), i aquest document no les segueix, llavors és invàlid.

4.6.2. Documents ben formats

Són els documents que segueixen les regles de sintaxi XML però no corresponen a cap DTD o Esquema determinat.

4.6.3. Documents vàlids

Són aquells documents que segueixen, tant les regles de sintaxi XML com les regles definides en el seu propi DTD o Esquema.

4.7. Regles: l'element arrel

Un document XML ha d'estar contingut en un element únic, l'element arrel, que contindrà el text i qualsevol altre element del document. En l'exemple següent observem que el document està contingut en un element anomenat **<principal>**.

```
<principal>
        <salutació>
                Hola a tots!!!!
        </salutació>
</principal>
```

Figura 3. Exemple d'element arrel en document XML

i aquest altre document com exemple de document invàlid per falta d'un element arrel:

```
<principal>
        <salutació>
                Hola a tots!!!!
        </salutació>
</principal>
<un altre element>
        Hola, una altra vegada!!!
</un altre element>
```

Figura 4. Exemple de document invàlid (sense element arrel)

4.8. Regles: Els elements no es poden sol·lapar

Aquí tenim un exemple il·legal:

```
<principal>
        <salutació>
                Hola a tots!!!!
</principal>
        </salutació>
```

Figura 5. Exemple de document invàlid (sol·lapament d'etiquetes)

Abans de tancar l'element **<principal>** hem de tancar **<salutació>.**

4.9. Regles: Les etiquetes finalitzadores són obligatòries

Al contrari dels documents HTML, les etiquetes finalitzadores en XML són estrictament obligatòries.

4.10. Regles: Els elements són "Case Sensitive" (sensibles a majúscules / minúscules)

Així com en HTML, **<hola>** i **<Hola>** són el mateix, en XML no.

Si intentem finalitzar un element **<hola>** con una etiqueta **</Hola>**, obtindrem un error.

4.11. Regles: Els atributs han de tenir valors i han d'estar entre cometes

Tot atribut ha de complir aquesta regla doble:

- Els atributs han de tenir valors
- Els valors han de estar entre cometes

Comparem aquest dos exemples (un que està bé i un altre que està malament):

CODI VÀLID
```
<principal>
        <salutació =" Hola a tots!!!! " />
</principal>
```

CODI INVÀLID
```
<principal>
        <salutació =Hola a tots!!!! />
</principal>
```

Figura 6. Exemples de documents vàlid i invàlid (valors entre cometes)

4.12. Altres coses que podem trobar en documents XML

4.12.1.– Comentaris

Els comentaris poden aparèixer en qualsevol lloc d'un document XML, fins i tot abans o després de l'element arrel.
Un comentari comença amb **<!-** i acaba amb **-->**.
Un comentari no pot contenir un guió doble (**--**) excepte com a final del mateix comentari, i fora d'aquesta excepció, pot contenir qualsevol cosa que serà sempre ignorada pels analitzadors XML

```
<!-  Això és un comentari per XML -->
<!-  i això un altre     ☺ ** #####&&& -->

<principal>
        <salutació =" Hola a tots!!!! " />
</principal>
```

Figura 7. Exemple de documents vàlid i invàlid (valors entre cometes)

4.12.2. Instruccions de processament

Una instrucció de processament és una marca que s'interpretarà com un segment de codi. Concretament les marques **<?** i **?>**
Veiem un exemple que indica al "parser" quin full d'estil ha de fer servir:

```
<? xsl-stylesheet href = "estil.css" ?>
```

Figura 8. Exemple de documents XML amb instrucció de processament

4.12.3. Entitats

L'especificació XML defineix entitats que s'utilitzen en lloc dels caràcters especials, alhora d'escriure el contingut entre les etiquetes d'inici i final:
< pel símbol menor que (<)
> pel símbol major que (>)
" per les cometes dobles (")
' per la cometa simple (')
& pel símbol (&).
&dw; es reemplaça amb la cadena "developerWorks"

4.13. Definició del contingut d'un document XML

Per definir els elements que s'utilitzaran per representar les dades d'un document XML tenim dues alternatives:

- Un mètode és utilitzar els **Document Type Definition** o **DTD**. Un DTD defineix els elements que poden aparèixer en un document XML, l'ordre d'aquests elements i altres detalls bàsics de l'estructura del document. Els DTD formen part de la especificació original de XML i són molt similars als DTD de SGML.
- l'altra alternativa consisteix en utilitzar un XML Schema o Esquema XML. Un esquema pot definir totes les estructures de document que es poden definir en un DTD i, a més, pot definir tipus de dades i regles complexes. El W3C[3] va desenvolupar l'especificació d'esquemes més tard que les especificacions originals XML

4.14. Esquemes XML

Degut a que el projecte en centra en l'estudi de diverses bases de dades natives per documents elaborats segons les regles de l'esquema MPEG-7, ens centrarem exactament en això, els esquemes XML i, en concret, en el MPEG-7.

Amb els esquemes XML tenim un major flexibilitat per definir els documents XML i presenten més avantatges que els DTD:

- Els esquemes utilitzen estrictament la sintaxi XML. Això vol dir que es pot processar un esquema XML de la mateixa manera que qualsevol altre document XML**.**
- Els esquemes XML suporten tipus de dades, tant els suportats pels DTD (Id, referències..) com d'altres (enters, nombres en coma flotant, dates, hores, cadenes de text, URL i altres tipus de dades útils pel processat i validació de dades).
- Els esquemes XML són extensibles i poden crear tipus de dades nous i derivar altres a partir dels ja creats.

Aquí podem observar un esquema XML d'exemple

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<xs:element name="alumnes">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name = "alumne"  type = "tipusAlumne"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
        <xs:complexType name = "tipusAlumne">
                <xs:sequence>
                        <xs:element name="nom" type="xs:string"/>
                        <xs:element name="cognom" type="xs:string"/>
                        <xs:element name="dataNaixement" type="xs:gYear"/>
                </xs:sequence>
                        <xs:attribute name="dni" type="xs:string"/>
        </xs:complexType>
</xs:schema>
```

Figura 9. Exemple d'esquema XML.

El llenguatge d'esquemes XML està definit en tres parts:

- **Un primari**, localitzat a http://www.*w3.org/TR/xmlschema-0*, que proporciona una introducció als documents d'esquema XML i per a què estan dissenyats

- Un estàndard pels **documents d'estructura**, localitzats a http://www.*w3.org/TR/xmlschema-1,* que defineix l'estructura dels documents XML.
- Un estàndard pels **tipus de dades**, localitzat en http://www.*w3.org/TR/xmlschema-2,* que defineix alguns tipus de dades comuns i les regles per crear d'altres.

4.14.1. Esquemes XML: Vocabulari

El vocabulari XML-Schema ens servirà per veure quin tipus d'elements es fan servir per descriure els esquemes dels documents:

- **Schema**: Serveix com element arrel dels documents XML-Schema
- **datatype**: Descriu els tipus de dades en els elements i els atributs
- **ElementType**: Descriu el tipus d'element
- **element**: Identifica un element que pot produir-se en altre tipusd'element
- **group**: Organitza els elements en grups
- **AttributeType**: Descriu el tipus d'atribut
- **attribute**: Identifica un atribut que pot produir-se en un tipus d'element
- **description**: Proporciona documentació per a un element o un atribut

4.14.2.Esquemes XML: Atributs dels elements

Els elements poden incloure atributs:

- **Schema**:
  - ◊ **name**: nom de l'esquema.
  - ◊ **xmlns: espai de noms de l'esquema.**

- **datatype**:
  - ◊ **dt:type:** especifica el tipus d'element o atribut (entity, entities, enumeration, boolean, char, date, etc.).

- **ElementType**:
  - ◊ **name:** nom de l'element**.**
  - ◊ **model:** si el model de contingut està obert o tancat.

- ◊ **content:** tipus de contingut que hi ha a l'element (empty, textOnly, eltOnly, mixed).
- ◊ **order:** ordre dels elements secundaris i grups que hi ha en l'element (one,seq, many).
- ◊ **dt:type:** tipus de l'element**.**

- **element:**
  - ◊ **type:** tipus de l'element**.**
  - ◊ **minOccurs:** mínim nombre de vegades que l'element ha de produir-se.
  - ◊ **maxOccurs:** màxim nombre de vegades que l'element ha de produir-se.

- **group:**
  - ◊ **order:** ordre dels elements secundaris que hi ha al grup (one, seq, many).
  - ◊ **minOccurs:** mínim nombre de vegades que l'element ha de produir-se.
  - ◊ **maxOccurs:** màxim nombre de vegades que l'element ha de produir-se.

- **AttributeType:**
  - ◊ **name:** nom del tipus de l'atribut
  - ◊ **dt:type:** tipus de dades del tipus d'atribut
  - ◊ **dt:values:** llista de valors possible d'un atribut enumerat(només s'aplica quan dt:type està establert a enumeration).
  - ◊ **default:** valor predeterminat d'un atribut.
  - ◊ **required:** indicador que mostra si ha de proporcionar-se l'atribut en l'element.

- **attribute:**
  - ◊ **type:** tipus de l'atribut
  - ◊ **default:** valor predeterminat de l' atribut**.**
  - ◊ **required:** indicador que mostra si ha de proporcionar-se l'atribut en l'element.
  - ◊ **description:** element de només text dissenyat amb finalitat de documentació.

5. MPEG-7

MPEG-7 és probablement l'estàndard més ambiciós de definició de metadades per a material Multimèdia (bàsicament, àudio i vídeo).

Representa l'esforç realitzat pel comitè MPEG de ISO per afegir descripció de continguts a la seva llista d'estàndards desenvolupats pel processat i representació d'informació Multimèdia, que va començar amb MPEG-1, MPEG-2 i MPEG-4.

L'objectiu s'aparta d'aquests estàndards, que s'ocupen, principalment, de la representació del contingut (codificació), i afegeix una capa semàntica per sobre, per la qual cosa depèn d'ells per proporcionar accés als continguts mateixos.

MPEG-7 és un estàndard recent: la data oficial de publicació com Estàndard Internacional (IS) és de l'octubre de 2001 i és per això que aquest estàndard incorpora molts avenços recents en el tractament de la informació Multimèdia.

MPEG-7 s'estructura al voltant del concepte de *descriptor*, una representació d'una propietat determinada de material audiovisual que caracteritza al contingut d'alguna manera. Els descriptors s'organitzen en *esquemes de descripció*, que són un conjunt ordenat de components (descriptors o altres esquemes de descripció) tot especificant les relacions entre ells. Un grup d'esquemes de descripció aplicat a un contingut audiovisual forma la *descripció* d'aquest contingut. Les descripcions s'expressen en un llenguatge denominat *Llenguatge de Definició de Descriptors* (DDL), definit per MPEG, i prenent com a base el llenguatge *XML Schema* del consorci W3C[3]. El llenguatge DDL permet també extensibilitat de l'estàndard (a nivell sintàctic) permetent especificar descriptors i esquemes de descripció no definits prèviament.

Com tots els estàndards produïts per MPEG, l'especificació de MPEG-7 està orientada a un model de decodificador. És a dir, MPEG-7 detalla la sintaxi i semàntica de les descripcions, de tal forma que qualsevol decodificador que compleixi la norma sigui capaç d'interpretar una descripció MPEG-7. El que no està especificat són el filtratge, la cerca, la navegació o l'accés i el mètode utilitzat en el codificador per generar-la, deixant espai per desenvolupaments o millores i la possibilitat d'oferir valor afegit per part d'empreses o institucions que produeixin aplicacions basades en MPEG-7.

A més, durant el desenvolupament de l'estàndard s'ha anant construint un model de programari complet, que inclou codificador i descodificador, anomenat "Model d'experimentació (XM)". El XM permet realitzar proves de funcionament de les parts desenvolupades i orientar la construcció de sistemes MPEG-7.

MPEG-7 ha estat dividit en seccions, que tracten diferents aspectes de la descripció del material Multimèdia. Les més importants són:

- **Sistemes**: que avarca aspectes globals d'accés a la informació, als mecanismes de transmissió, a la sincronització de la informació, als formats de fitxer, al multiplexat de descripcions i a la qualitat de servei.
- **DDL**. Defineix el llenguatge de definició de descripcions.
- **Visual**. Defineix la descripció d'elements visuals, per les seves propietats espacials (color, textura o forma), per les seves temporals (moviment o activitat), per la seva localització espai-temporal (posició o trajectòria) o per les seves característiques específiques (reconeixement de cares…).
- **Audio**. Especifica els mètodes de descripció de continguts d'àudio basats en les seves característiques (timbre, ritme, distribució espaial, temporal i espectral o font de só), així com la transcripció a text (per veu o lletres de cançons…).
- **Esquemes de descripció (MDS).** Secció que especifica les eines de descripció genèriques (o no aplicables només a vídeo o àudio) i l'organització dels elements individuals, per poder aplicar aplicar a la creació de descripcions completes de l'estructura física i del contingut semàntic del material. Dintre de MDS es troben els esquemes de descripció pels usuaris dels serveis audiovisuals, que permeten especificar les preferències personals i els patrons d'ús per oferir serveis de personalització.

Els següents són extractes d'un complet exemple disponible a la web de la Universitat Pompeu Fabra[4]:

Video description

```
<Mpeg7 type="complete">
        <ContentDescription xsi:type="ContentEntityType">
                <MultimediaContent xsi:type="VideoType">
                        <TextAnnotation>
                                <StructuredAnnotation>
                                        <Who><Name> Suecia </Name></Who>
                                        <WhatObject><Name>Soccer field</Name></WhatObject>
                                        <WhatAction><Name>Soccer game</Name></WhatAction>
                                        <Where><Name> Spain </Name></Where>
                                </StructuredAnnotation>
                        </TextAnnotation>
                </MultimediaContent>
        </ContentDescription>
</Mpeg7>
```

Figura 10. Exemple d'esquema XML MPEG-7.

Point of view description

```
<TemporalDecomposition>
  <Header xsi:type="DescriptionMetadataType">
        <Confidence>1.0</Confidence>
        <LastUpdate>2001-02-20T10:00:00+00:00</LastUpdate>
        <Comment>
              <FreeTextAnnotation>Description of PointOfView of soccer game video.</FreeTextAnnotation>
        </Comment>
        <Creator>
              <Role href="creatorCS">
                    <Name>Creator</Name>
              </Role>
              <Agent xsi:type="PersonType">
                    <Name>
                        <GivenName>Koichi Emura</GivenName>
                    </Name>
              </Agent>
        </Creator>
        <CreationTime>2001-01-08T10:00</CreationTime>
        <Instrument>
              <Tool>
                  <Name>Manual</Name>
              </Tool>
        </Instrument>
  </Header>
  <VideoSegment id="Seg0">
        <PointOfView viewpoint="CAMERA ZOOM">
              <Importance>      <Value>0.2</Value>          </Importance>
        </PointOfView>
        <PointOfView viewpoint="CAMERA SPEED">
              <Importance>      <Value>0.6</Value>          </Importance>
        </PointOfView>
        <PointOfView viewpoint="exciting">
              <Importance>      <Value>0.6</Value>          </Importance>
        </PointOfView>
        <MediaTime>
              <MediaRelTimePoint mediaTimeBase="../../MediaLocator[1]">PT10S</MediaRelTimePoint>
              <MediaDuration>PT1N10F</MediaDuration>
        </MediaTime>
  </VideoSegment>
  <VideoSegment id="Seg1">
        <PointOfView viewpoint="CAMERA ZOOM">
              <Importance>      <Value>1.0</Value>          </Importance>
         </PointOfView>
        <PointOfView viewpoint="CAMERA SPEED">
              <Importance>      <Value>0.3</Value>          </Importance>
        </PointOfView>
        <PointOfView viewpoint="exciting">
              <Importance>      <Value>0.7</Value>          </Importance>
        </PointOfView>
        <MediaTime>
              <MediaRelTimePoint mediaTimeBase="../../MediaLocator[1]">PT20S</MediaRelTimePoint>
              <MediaDuration>PT1N10F</MediaDuration>
        </MediaTime>
  </VideoSegment>
</TemporalDecomposition>
```

Figura 11. Exemple(2) d'esquema XML MPEG-7.

6. Bases de dades natives XML

Una de les solucions més eficients per guardar dades en format XML és la de les bases de dades natives XML. Aquestes bases de dades proporcionen la funcionalitat necessària per emmagatzemar dades en format XML i realitzar consultes en la informació gestionada mantenint el model XML associat intacte.

Una base de dades nativa XML defineix un model lògic per cada document XML i emmagatzema i recupera informació ajustada a aquests model. Com a mínim, aquests models lògics han d'incloure elements, atributs, seccions PCDATA i han de contemplar l'ordenació de documents.

A més, les bases de dades natives XML tenen un document XML com a unitat bàsica d'emmagatzematge, de la mateixa manera que una base de dades relacional té un registre que pertany a una taula determinada.

Un altre aspecte a considerar és que no es requereix que tingui una estructura associada per l'emmagatzematge físic de les dades. Pot ser construïda sobre una base de dades relacional o orientada a objectes, o bé utilitzar formats d'emmagatzematge propietaris com fitxers indexats o comprimits.

De totes maneres, les bases de dades natives XML no volen representar un nou model de base de dades de baix nivell ni reemplaçar als sistemes d'emmagatzematge actuals. S'han plantejat com una eina de suport al desenvolupament que permetin als desenvolupadors disposar d'una solució robusta per l'emmagatzematge i manipulació de dades en format XML.

6.1. Emmagatzematge de XML natiu

Les bases de dades natives XML emmagatzemen els documents creant models lògics. Aquests models es basen directament en XML o en alguna de les tecnologies relacionades com DOM o Infoset. Aquests models inclouen diferents nivells d'agregació i complexitat, i un suport complet per la gestió de dades semiestructurades.

Els models són automàticament mapejats per les bases de dades XML natives al mecanisme d'emmagatzematge corresponent. El mapeig utilitzat per aquestes bases de dades garantitzarà la correcta utilització del model associat al document original.

Una vegada s'ha emmagatzemat la informació, i per poder aprofitar al màxim la capacitat d'aquests sistemes, és necessari continuar utilitzant eines XML que permetin realitzar altres tipus d'operacions. No té sentit, per exemple, emmagatzemar dades XML sobre una base de dades de tipus relacional i després executar consultes amb SQL, ja que, en realitat el que es guarda en el sistema d'emmagatzematge és el model del document XML en lloc de les entitats de negoci que representa cada dada. El model de negoci només existeix en el nivell de domini del document XML, no al nivell del sistema intern d'emmagatzematge. Llavors, per treballar amb les dades és necessari fer-ho amb XML.

El principal avantatge de les bases de dades natives XML és l'abstracció. El programador no té per què conèixer els detalls de com s'emmagatzema la informació, i pot construir les seves aplicacions amb total llibertat.

6.2. Dependència de l'esquema

Les bases de dades natives XML gestionen els documents com col·leccions de dades, permetent realitzar consultes i manipulacions sobre aquests documents en forma de conjunts d'informació. És un concepte similar al de les taules en les bases de dades relacionals. La diferència amb els sistemes relacionals és que no totes les bases de dades natives XML necessiten un esquema per generar una col·lecció de documents. Això vol dir que es pot emmagatzemar qualsevol document XML en la col·lecció independentment de la seva estructura, permetent la realització de consultes en tot el conjunt de dades amb una gran flexibilitat.

Les bases de dades natives XML que suporten aquesta característica es denominen *independents de l 'esquema*. Disposar de col·leccions de documents independents de l'esquema proporciona a la base de dades una alta flexibilitat i facilita el desenvolupament d'aplicacions. Per altra banda, aquesta característica dificulta la tasca dels administradors de bases de dades degut a possibles problemes d'integritat de dades. De vegades és preferible disposar de menys flexibilitat, assegurar una correcta integritat referencial i optar per solucions dependents de l'esquema o altres tipus d'aplicacions d'emmagatzematge de documents XML.

Alguns dels productes disponibles suporten validació de documents a través de DTDs i alguns disposen de llenguatges de definició d'esquemes propietaris.

6.3. eXist

eXist és una base de dades nativa XML que proveeix una solució d'emmagatzematge pròpia i, a més, permet connectar aplicacions de tipus relacional per emmagatzematge auxiliar.

Es distribueix en forma de codi lliure, i es pot recuperar la darrera versió des de la pàgina web[1]

La solució d'emmagatzematge pròpia està totalment escrita en Java i utilitza les seves pròpies estructures de dades internes per guardar i indexar documents XML al disc.

També pot integrar-se amb altres aplicacions de tipus relacional per l'emmagatzematge indexat dels documents.

Suporta MySQL, PostgreSQL i Oracle i, independentment del sistema utilitzat (solució nativa o relacional), el processador XPath integrat en el producte permet realitzar consultes amb total abstracció de la capa d'emmagatzematge inferior.

El motor de cerques ha estat dissenyat per l'execució de "queries" XPath i XQuery de forma eficient, utilitzant índexs.

El servidor és accessible a través d'interfícies HTTP o XML/RPC i suporta l'API de programació Java XML:DB.

6.4. Xindice

Xindice és una base de dades XML desenvolupada en Java que s'ha dissenyat per l'emmagatzematge de grans quantitats de petits documents XML.

Es distribueix en forma de codi obert lliure des de la pàgina web de la *Apache software foundation*[2]

Pot indexar valors estructurals i contingut, emmagatzemant els documents XML en format comprimit per estalviar espai al disc.

Els documents s'estructuren en una jerarquia de col·leccions i poden ser consultats utilitzant XPath. Com llenguatge d'actualització es pot utilitzar XUpdate, i per la gestió d'enllaç entre document el llenguatge Xlink.

Suporta tres tipus d'interfícies de programació: XML:DB, un API per CORBA i un plugin XML-RPC que suporta accessos des de PHP, Perl i Applescript.

La funcionalitat del servidor es extensible utilitzant la tecnologia XMLObjects.

7. El llenguatge de consultes XPath

XPath és un llenguatge sofisticat i complex, però diferent dels llenguatges procedurals com són C, C++, Basic, Java...

XPath és, també, la base sobre la qual s'han especificat noves eines per tractar amb documents XML, com, per exemple, **XPointer**, **XLink** i **XQuery** (el llenguatge que fa servir els documents XML com si fos una base de dades).

XPath sirveix per decidir com s'ha de processar un full d'estil o el contingut d'una pàgina XML, però també per poder posar enllaços o carregar en un navegador zones determinades d'una pàgina XML, en lloc de tota la pàgina.

7.1. El model de dades de XPath

7.1.1. Construcció de l'arbre de nodes

Un document XML és processat per un analitzador (parser) construint un **arbre de nodes**. Aquest arbre comença amb un element arrel, que es diversifica al llarg dels elements que pengen d'ell i finalitza en nodes fulla, que contenen només text, comentaris, instruccions de processat o, fins i tot, que poden estar buits i només tenen atributs.

La forma en què XPath selecciona parts del document XML es basa precisament en la representació en forma d'arbre que es genera del document. De fet, els "operadors" de què consta aquest llenguatge ens recorden la terminologia genealògica: fill, pare, descendent...

Un cas especial de node són els nodes **atribut**. Un node pot tenir tants atributs com desitgi, i per cada un es crea un node atribut. No obstant, aquest nodes atribut no es consideren fills seus, sinó més aviat com etiquetes afegides al node element.

A continuació es mostra un exemple de com convertim en arbre un document XML.

Document XML

```
<llibre>
        <titol>Dos por tres calles</titol>
        <autor>Josefa Santos</autor>
        <capitol num="1"> La primera calle
                <paragraf> Era una sombría noche del mes de agosto...
                </paragraf>
                 <paragraf destacar="si"> Ella, inocente cual que surca el cielo en busca de libaciones... </paragraf>
        </capitol>
        <capitol num="2" public="si"> La segunda calle
                <paragraf>Era una obscura noche del mes de septiembre...
                </paragraf>
                <paragraf> Ella, inocente cual que surca el viento en busca del néctar de las flores... </paragraf>
        </capitol>
        <apendix num="a" public="si"> La tercera calle
                <paragraf> Era una densa noche del mes de diciembre...
                </paragraf>
                <paragraf> Ella, cándida cual que surca el espacio en busca de bichejos paracomer...
                </paragraf>
        </apendix>
</llibre>
```

Figura 12. Exemple de document XML.

Arbre associat

```
/
|
+---llibre
    |
    +---titol
    |   |
    |   +---(text)Dos por tres calles
    |
    +---autor
    |   |
    |   +---(text)Josefa Santos
    |
    +---capitol [num=1]
    |   |
    |   +---(text)La primera calle
    |   |
    |   +---paragraf
    |   |   |
    |   |   +---(text)Era una sombr&iacute;a noche ...
    |   +---paragraf
    |       |
    |       +---(text)Ella, cual inocente mariposa...
    |
    +---capitol [num=2]
        |
        +---(text)La segunda calle
        |
        +---paragraf
        |   |
        |   +---(text)Era una obscura noche ...
        +---paragraf
            |
            +---(text)Ella, cual inocente abeja...
```

Figura 13. Exemple d'arbre associat a un document XML.

7.1.2. Camins de localització

Una "instrucció" en llenguatge XPath es denomina **expressió**, les cometes de la paraula instrucció és per què XPath és un llenguatge declaratiu i les instruccions no són exactament el que estem acostumats a veure en llenguatges procedurals o imperatius.

Aquestes expressions poden incloure certa varietat d'operacions sobre diferents tipus d'operands. Hi ha, bàsicament, dos tipus d'operands: *crides a funcions* i location paths (camins de localització).
Els location path són la més important de les expressions que es poden especificar en notació XPath. La sintaxi d'un location path és simliar a la utilitzada a l'hora de descriure els directoris que formen una unitat de disc en Unix o Linux (i similar a la dels sistemes basats en MS-DOS i Windows, amb l'excepció feta per les que utilitzem amb les unitats de disc (C:, A:) i que les barres utilitzades són "/" en lloc de les típiques "\" d'aquests sistemes operatius. La resta de sintaxi és molt similar a la del sistema d'arxius citat.

Però el significat de les expressions és totalment diferent. Per exemple, el següent path en Unix: **/usr/home/user/docs** fa referència a un únic directori: **docs** que "penja" del conjunt de directoris **/usr/home/user**.

En canvi, la següent expressió en XPath: **/llibre/capítol/paràgraf** fa referència a tots els elements **paràgraf** que pengin directament de qualsevol element **capítol** que "pengi" de qualsevol element **llibre** que, finalment, "pengin" del node arrel, /.

Hem de tenir en compte que una expressió en XPath no retorna els elements que compleixin amb el patró que representa aquesta expressió, si no que retorna una **referència** a aquests elements; és a dir, una expressió XPath retorna una llista d'apuntadors als elements que encaixen en el patró. Aquesta llista pot estar buida o contenir un o més nodes.

7.1.3. Node context

Un location path sempre té un punt de partida anomenat **node context**. El **node context** és com el directori actual si ens referim a un sistema d'arxius. Així, si estant en Unix, donem una ordre **ls** obtindrem els arxius que hi ha en el directori actual, mentre que si diem **ls /usr/bin** obtindrem el llistat dels arxius existents en el directori **/usr/bin** amb independència del directori en què ens trobem en aquest moment.

En els location path passa el mateix. A menys que s'indiqui un camí explícit, s'entendrà que el location path parteix del node què en cada moment s'estigui processant.

El concepte de "node context" és imprescindible per comprendre com es duu a terme l'elecció dels nodes que s'ajusten al patró indicat en el location path. Per explicar això, podem observar com actuaria un motor d'avaluació d'expressions XPath al llegir la següent expressió aplicada al document XML de l'exemple anterior: **/llibre/capitol/paràgraf**

1. En primer lloc llegeix /, la qual cosa li fa seleccionar el node arrel, independentment del node context que en aquell moment existeixi. En el moment en que l'avaluador de XPath localitza el node arrel, aquest passa a ser el node context de l'expressió.

2. L'analitzador llegeix ara **llibre**, i selecciona tots els que "pengen" del node context (que ara mateix és l'arrel) que s'anomenin **llibre**. A més, ara, el node context passa a ser **/llibre**.

3. Si continuem avançant l'analitzador llegirà **capitol**, i selecciona tots els elements **capitol** del node. En un disc seria impossible que haguessin dos directoris amb el mateix nom "penjant" d'un mateix directori pare. En canvi, en el nostre document és perfectament possible i vàlid. Per tant, en aquest moment hi ha dos elements que encaixen amb el patró **/llibre/capitol.**

4. L'analitzador continua legint l'expressió XPath que l'hem donat i arriba a **paràgraf**. Llavors selecciona tots els elements **paràgraf** que "pengen" del node context...¡¡però no hi ha un node context, si no dos!!. Però, no hi ha problema ja que l'avaluador d'expresions els recorrerà tots, un per un avaluant cada node i tractant-lo com el node context d'aquell determinat moment.

5. El resultat final és un nou conjunt de nodes (o per ser més precisos, conjunt de punters a node) que encaixen amb el patró cercat.

7.1.4. Predicats

XPath és un gran mecanisme per seleccionar molts nodes a la vegada. Però, què passa si només volem seleccionar un node que compleix unes característiques determinades?. O més d'un node que compleixi amb un patró però no tots els que ho compleixen si no només aquells amb un atribut determinat?
Doncs, amb els **predicats**.

Els predicats s'inclouen dins d'un location path utilitzant corxets, per exemple:
/llibre/capitol[@num="1"]/paragraf

Mitjançant l'anterior location path estem indicant que s'escollissin tots els elements **paragraf** de tots els elements **capitol** que tinguin un atribut anomenat **num** i que tingui assignat el valor "1" (recordem que en XML tots els atributs tenen valors de tipus String).

En el nostre exemple, només hi ha un **capítol** que compleixi aquestes condicions, de manera que només els elements **paràgraf** que conté seran seleccionats

7.1.5. Eixos

Un **eix** inclòs en un location path realitza una selecció de nodes dins de l'arbre (o millor dit, dins del subarbre que "penja" del node o conjunt de nodes context) d'acord amb algun patró.

Per exemple, cada vegada que utilitzàvem la barra / (excepte per denominar el node arrel) estàvem utilitzant un **eix**.

Vegem els diferents **eixos** que podem utilitzar per recórrer l'arbre.

7.1.5.1. Fill

És l'eix utilitzat per defecte. Es correspon amb la barra / (malgrat que té una forma més llarg que és: **/child::**).

Selecciona tots els **titol** d'un **llibre**:

/llibre/titol

Figura 14. Exemple d'accés a eix fill.

7.1.5.2. Atribut

Es correspon amb el signe "arroba", @ (o la forma llarga: **attribute::**).

Mitjançant aquest operador podem seleccionar aquells nodes atribut que vulguem, indicant el nom. Per seleccionar els nodes element que contenen els atributs, hem d'utilitzar un predicat (tal i com hem fet anteriorment)

Selecciona l'atribut **num** que tinguin els elements **capitol**

/llibre/capitol/@num

Selecciona tots els elements fill dels capitols que tinguin l'atribut **public**

/llibre/capitol[@public]/*

Selecciona tots els elements fill de **paragraf** amb atribut **destacar** igual a "si".

/llibre/titol/paragraf[@destacar="si"]

Figura 15. Exemple d'accés a atributs.

7.1.5.3. Descendant

S'especifica posant una doble barra: // (la seva forma llarga és: **descendant::**).

Serveix per seleccionar tots els nodes que "pengen" del conjunt de nodes context. Es a dir, no només els fills dels nodes context, si no també tots els descendents.

Selecciona tots els **paragraf** d'un **llibre**:

/llibre//paragraf

Selecciona tots els descendents de **paragraf** que tenen un atribut **href**.

//paragraf//*[@href]

Per veure el valor de l'atribut **href** del cas anterior:.

//paragraf//*[@href]/@href

Selecciona tots els elements descendents de **capitol**

/llibre/titol//*

Figura 16. Exemple d'accés a eix *descendant*.

7.1.5.4. Self

S'especifica mitjançant el punt (.)

És molt útil ja que selecciona el node context. Per exemple, suposem que volem seleccionar tots els **paragraf** descendents del node context. No podem escriure **//paragraf**, ja que seleccionaria tots els descendents del node arrel. Llavors, la forma correcta és: **.//paragraf**

7.1.5.5. Parent

S'especifica de manera similar als sistemes d'arxius, amb els dos puntets (..)

El comportament d'aquest eix realitza una passa enrrera en l'arbre de nodes

Selecciona tots els nodes que tenen fills **paragraf**:

//paragraf/..

Selecciona tots els nodes **capitol** que tenen fills **paragraf**:

//paragraf/../../capitol

O bé:

//capitol/paragraf/..

Figura 17. Exemple d'accés a eix *parent*.

7.1.5.6. Ancestor

De tots els eixos que podem utilitzar, aquest és l'únic que no té cap forma abreujada, si no que s'ha d'escriure: **ancestor::**

Ancestor és a **parent** el mateix que **descendant** es a **child (fill).** És a dir, retorna tots els elements dels quals el node context és descendent.

Selecciona tots els elements que tenen entre els seus descendents, **paragraf**

//ancestor::paragraf/ancestor::*

Selecciona tots els **capitol** que tenen entre els seus descendents cap amb l'atribut **href**

//*[@href]/ancestor::capitol

Figura 18. Exemple d'accés a eix *ancestor*.

7.1.5.7. Nodes Test

Els **nodes test** són una mena de funcions que restringeixen els resultats retornats per expressions **XPath**.

N'hi ha vàries:

- **node():** El node test nod() retorna tots els nodes de tots els tipus.
- **text():** retorna qualsevol node de tipus text. Només aplicable a eixos de contingut
- **comment():** retorna qualsevol node de tipus comentari**.**
- **processing-instruction():** retorna qualsevol node de tipus d'instrucció de procés, independentment de la seva destinació
- **processing-instruction(destinació):** retorna qualsevol node de tipus d'instrucció de procés relatiu a la destinació especificada**.**
- **processing-instruction(cursor):** retorna qualsevol node de tipus d'instrucció de procés relatiu a la destinació cursor

7.1.5.8. Altres funcions

També tenim certes funcions que ens serviran per restringir el conjunt de nodes retornats en una expressió XPath basant-se en la posició de l'element retornat.

Aquestes funcions són:

- **position( )**

Selecciona el segon **capitol**:

//capitol[position()=2]

o bé.

//capitol[2]

Figura 19. Exemple de funció *position ( ).*

- **id( )**

Seleccionar els **paragraf** fill de l'element amb id="capitol_1":

id("capitol_1")/paragraf

Figura 20. Exemple de funció *id ( ).*

- **last( )**

Selecciona l'últim **capitol**:

//capitol[last()]

Selecciona tots els **capitol** menys l'últim:

//capitol[not(position()=last())]

Selecciona el penúltim **capitol**:

//capitol[last()-1]

Figura 21. Exemple de funció *last ( ).*

7.2. XPath 2.0

XPath 2.0 és una nova especificació del consorci W3C[3] que intenta ampliar les funcionalitats i, a la vegada, corregir certs aspectes de XPath 1.0, mantenint les característiques del llenguatge i la compatibilitat, de manera que s'aconsegueixi un canvi gradual i gens traumàtic.

Entre les diferències més significatives en el nou XPath 2.0 hi trobem:

7.2.1. Generalització del concepte de seqüència

En XPath 1.0 el tipus bàsic de dades és el *node-set*, en el que l'ordre dels elements és el mateix que en el document i no s'admeten elements repetits. En canvi, en XPath 2.0, el concepte de *node-set* ha estat substituït por altre més obert (la seqüència), on, a diferència del seu predecessor, es permet l'aparició d'elements repetits i es poden definir altres criteris d'ordenació diferents al de l'aparició en el document.

7.2.2. Expresions condicionals i d'iteració (if i for)

XPath 2.0 incorpora novetats en el camp del control de flux per l'avaluació d'expressions XPath, cosa que no s'observava en les versions anteriors de l'estàndard.

Nous constructors d'expressions condicionals del tipus **if-then-else** i repetitives del tipus **for-in** han estat incorporades, gràcies a la influència de XQuery, per la definició d'aquests tipus d'expressions.

Però, la definició d'aquests tipus d'expressions no és la mateixa que ens trobem en un llenguatge imperatiu, ja que es tracta de constructors d'expressions, no de sentències. És a dir, una expressió **for** de **XPath**, no constitueix un conjunt de sentències executades seqüencialment, si no que construeix una expressió que té com a valor la seqüència formada pels elements construïts en cada una de les iteracions.

7.2.3. Expresions quantificades

Les expresions quantificades serveixen per comprovar que es compleix una determinada propietat en tots els elements d'un conjunt o, si més no, en algun d'ells. Les eines que utilitza són dos quantificadors, **some** i **every**, que, aplicats a

una propietat o valor i a una llista d'elements, determinen si aquesta propietat es certa per, al menys un, o per a tots els elements de la seqüència.

7.2.4. Operacions de conjunts

Una Altra novetat incorporada a la nova versió 2.0 de **XPath** són els operadors definits per construir noves seqüències a partir d'altres seqüències ja existents. D'aquesta manera, hi ha operadors para crear unions (**union**), interseccions (**intersect**) o diferència (**except**) de seqüències com si fossin conjunts eliminant els elements repetits i no considerant l'existència d'un ordre entre els seus elements.

Per exemple, la seqüència formada per la unió o per la intersecció de dues seqüències A i B es la formada pels elements que pertanyen a A i/o pertanyen a B (respectivament).

8. El llenguatge de consultes XQuery

El component principal de XQuery que més familiar troben els usuaris de XML és XPath. La raó es troba en que XQuery està basat, entre altres, en el llenguatge de consultes XPtah. En concret, XQuery està basat en els següents llenguatges:

- XPath i XQL: Del quals adopta la sintaxi per navegar pels documents jeràrquics.
- SQL: (llenguatge relacional). D'on extreu les clàusules FLWR i operadors com el **join**.
- XML-QL: D'aquest llenguatge adopta la idea d'assignar variables per crear noves estructures.

8.1. Origens de XQuery

XQuery va ser creat, originàriament com a vehicle de proves per la sintaxi de nivell d'usuari i estava basat en un altre llenguatge que es diu Quilt. Quilt, per la seva banda estava basat en els esforços de col·laboració d'un gran grup de treball dins del W3C[3], per definir requisits, casos d'us i un model de dades i àlgebra subjacents.

Quilt va ser un projecte encapçalat per Jonathan Robie, Don Chamberlin i Daniela Florescu. Robie és, probablement el més conegut por encapçalar els esforços per estandarditzar i promoure XQL, el predecessor de XQuery i el més proper actualment a un llenguatge estàndard real en el món dels llenguatges de consultes XML, amb una dotzena d'implementacions en us. Chamberlin és el co-disenyador de SQL i és membre de l'equip d'investigació d'IBM que va desenvolupar la major part de les tecnologies de bases de dades relacionals avui dia. L'aportació de Florescu no és menys destacable. És autora i co-autora de més de 50 documents d'investigació sobre l'optimització dels llenguatges de consulta i de l'arquitectura iniciada fa ja quasi 10 anys.

Davant les diferents perspectives dels dos grups (els del punt de vista de documents i els del punt de vista de dades), i davant la solidesa de les bases exposades, no és estrany que l'especificació hagi trigat tant de temps en estar a disposició del públic.

Els procediments interns dels grups de treball de W3C[3] són confidencials, i la major part dels esforços del grup **Query Language Working Group** realitzats abans de mitjans de febrer de 2001 van tenir lloc a porta tancada.

En un principi es va publicar un document de requisits i un document esborrany de treball del model de dades, però el volum de publicació del grup de treball va començar realment a ser una realitat en febrer, quan va aparèixer un volum més gran de documentació.

Després de dues actualitzacions importants en juny i desembre de 2001, ara ja la major part de la documentació ja es troba publicada.

Entre aquesta documentació cal destacar:

- XML Query Requirements que es tracta d'un document de planificació pel grup de treball. Una llista de *desiderátum* de XQuery.

- XML Query Use Cases: Diferents casos reals i fragments de XQuery que resolen problemes específics.

- XQuery 1.0: An XML Query Language: Document central que constitueix una introducció al llenguatge i una descripció general de la major part de conceptes.

- XQuery 1.0 and XPath 2.0 Data Model: Ampliació de la informació sobre XML. En aquest document es descriuen elements de dades que han de quedar clars en una implementació d'una consulta, i els conceptes bàsics de la semàntica formal

- XQuery 1.0 Formal Semantics: Àlgebra subjacent que defineix formalment el llenguatge.

- XML Syntax for XQuery 1.0 (XQueryX): Sintaxi alternativa per aquelles persones que prefereixen XML. És la sintaxi preferida per la major part de màquinas.

- XQuery 1.0 and XPath 2.0 Functions and Operators Version 1.0: Funcions i operadors bàsics de tipus de dades d'esquema i nodes i seqüències de nodes XQuery.

- XML Path Language (XPath) 2.0: Documentació de XPath.

Per començar amb aquest gran volum de documentació és recomanable fer-ho des del document XQuery 1.0: An XML Query Language, que conté una visió general introductòria molt bona, o bé pel document esborrany de treball Use Cases, on es descriuen casos reals d'aplicació de XQuery sobre dominis d'aplicació específics i amb enumeració de possibles consultes XQuery.

Els documents esborrany de treball Data Model i Formal Semantics proporcionen una base teòrica precisa per XQuery. Els dos documents detallen l'*àlgebra* de consultes, un conjunt de descripcions precises que defineixen en termes formals les entitats principals en les què podem aplicar una consulta XQuery, i formulacions que els diferents operadors poden realitzar amb els operands.

8.2. XQuery germà de XPath 2.0

XQuery comparteix un model de dades comú amb XPath 2.0. El model de dades només descriu la informació bàsica en un document XML que és interessant per un processador XPath i de la mateixa manera que XPath 2.0, les seqüències de nodes estan ordenades, i permeten l'existència de duplicats.

8.3. XQueryX

XQueryX és la especificació d'una sintaxi alternativa basada en XML pel llenguatge superficial.

Un dels requisits per XQuery indica que ha de ser possible utilitzar diverses sintaxis i, per això, una de les sintaxis hauria de ser adient per a què els usuaris poguessin llegir i escriure, mentre que la resta s'haurien de poder expressar en llenguatge XML.

Tenir una representació de consulta basada en XML té tots els avantatges coneguts de XML: facilita a les eines estàndards analitzar, generar i interrogar el contingut d'una consulta. Això pot ser útil, per exemple, si es realitza una optimització o transformació a nivell de font, que pot dependre, a la vegada, de la capacitat d'analitzar amb facilitat una consulta per detectar una estructura gramatical determinada. Sabem que XML funciona molt bé en aquests casos.

Aquest exemple ens mostra una consulta simple en sintaxi XQuery:

```
LET $authors := /book/author
        RETURN
        <AUTHORS>
        {  $authors }
         </AUTHORS>
```

Figura 22. Exemple de sintaxi XQuery.

i aquest altre, el codi equivalent en XQueryX:

```
<q:query xmlns:q="http://www.w3.org/2001/06/XQueryx">;
      <q:flwr>;
         <q:letAssignment variable="$authors">;
            <q:step axis="CHILD">;
               <q:identifier/>;
             <q:step axis="CHILD">;
                  <q:identifier>;book</q:identifier>;
                  <q:identifier>;author</q:identifier>;
             </q:step>;
            </q:step>;
         </q:letAssignment>;
         <q:return>;
            <q:elementConstructor>;
               <q:tagName>;
                  <q:identifier>;AUTHORS</q:identifier>;
               </q:tagName>;
               <q:variable>;$authors</q:variable>;
            </q:elementConstructor>;
         </q:return>;
      </q:flwr>;
   </q:query>;
```

Figura 23. Exemple de sintaxi XQueryX.

L'excés d' informació de procés és irrellevant per la màquina que processa una consulta (a menys que es tracti de consultes realment grans), però si que suposaria un problema per una persona encarregada de dur a terme una depuració de la operació.

És per això que XQueryX no és gaire utilitzat i els motors de cerca actuals, normalment, no el suporten.

8.4. Sintaxi de XQuery.

De la mateixa manera que hem fet anteriorment per examinar i analitzar la sintaxi de XPath, prenem com a base de treball un document extret del document Use Cases citat anteriorment.

Aquest és el document que utilitzarem pel nostre anàlisi:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bib>
	<book year="1994">
		<title>TCP/IP Illustrated</title>
		<author>
			<last>Stevens</last>
			<first>W.</first>
		</author>
		<publisher>Addison-Wesley</publisher>
		<price>65.95</price>
	</book>
	<book year="1992">
		<title>Advanced Programming in the Unix environment</title>
		<author>
			<last>Stevens</last>
			<first>W.</first>
		</author>
		<publisher>Addison-Wesley</publisher>
		<price>65.95</price>
	</book>
	<book year="2000">
		<title>Data on the Web</title>
		<author>
			<last>Abiteboul</last>
			<first>Serge</first>
		</author>
		<author>
			<last>Buneman</last>
			<first>Peter</first>
		</author>
		<author>
			<last>Suciu</last>
			<first>Dan</first>
		</author>
		<publisher>Morgan Kaufmann Publishers</publisher>
		<price>39.95</price>
	</book>
	<book year="1999">
		<title>The Technology and Content for Digital TV</title>
		<editor>
			<last>Gerbarg</last>
			<first>Darcy</first>
			<affiliation>CITI</affiliation>
		</editor>
		<publisher>Kluwer Academic Publishers</publisher>
		<price>129.95</price>
	</book>
</bib>
```

Figura 24. Document XML per consultes XQuery.

8.4.1. Extreure nodes amb funcions

Amb aquest propers exemples d'utilització de XQuery, podrem observar com s'utilitza la mateixa sintaxi que XPath 2.0 en les expressions ..

La següent expressió XQuery:

```
doc("books.xml")
```

Figura 25. Expressió XQuery (1).

Extreu el document complet de XML de l'arxiu "books.xml".

i aquesta altra:

```
doc("books.xml")/bib/book/title
o bé
doc("books.xml")//title
```

Figura 26. Consulta XQuery.

ens dóna el següent resultat:

```
<title>TCP/IP Illustrated</title>
<title>Advanced Programming in the Unix environment</title>
<title>Data on the Web</title>
<title>The Technology and Content for Digital TV</title>
```

Figura 27. Resultat de la consulta XQuery.

També podem fer servir expressions:

```
doc("books.xml")/bib/book[price<50]
```

Figura 28. Consulta XQuery (2).

de la que obtindrem:

```
<book year="2000">
  <title>Data on the Web</title>
  <author><last>Abiteboul</last><first>Serge</first></author>
  <author><last>Buneman</last><first>Peter</first></author>
  <author><last>Suciu</last><first>Dan</first></author>
  <publisher>Morgan Kaufmann Publishers</publisher>
  <price>39.95</price>
</book>
```

Figura 29. Resultat de la consulta XQuery(2).

8.4.2. Clàusules FLWOR = For, Let, Where, Order by, Return

Mirem aquest exemple:

```
for $x in doc("books.xml")/bib/book
where $x/price>50
order by $x/title
return $x/title
```

Figura 30. Consulta XQuery (3).

resultat:

```
<title>Advanced Programming in the Unix environment</title>
<title>TCP/IP Illustrated</title>
<title>The Technology and Content for Digital TV</title>
```

Figura 31. Resultat de la consulta XQuery(3).

- La clàusula **for** selecciona tots els "**book** nodes" (**book** elements que "pengen" de l'element **bib**) i els posa a la variable anomenada **$x**.
- La clàusula **where** selecciona de **$x** "**book** nodes" aquells nodes que tenen un element **price** més gran que 50.
- La clàusula **order by** ordena aquest últim resultat pel element **title** "**title** nodes".
- La clàusula **return** ens facilita el resultat definitiu.

8.5. XQuery i Schema

XQuery també està especialment indicat per treballar amb Schemes XML

Veiem uns exemples:

Schema d'exemple:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="characters" type="Characters"/>

  <xsd:complexType name="Character">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="gender" type="Gender"/>
      <xsd:element name="species" type="xsd:string"/>
      <xsd:element name="vocation" type="xsd:string"/>
      <xsd:element name="level" type="xsd:nonNegativeInteger"/>
      <xsd:element name="health" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:simpleType name="Gender">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Female"/>
      <xsd:enumeration value="Male"/>
      <xsd:enumeration value="Other"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="Characters">
    <xsd:sequence>
      <xsd:element name="character" type="Character" minOccurs="0"
                                                    maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

Figura 32. Exemple de XML Schema.

i com a document associat a aquest Schema el document Characters.xsd localitzat a la pàgina web de Kurt Cagle[5].

```
<characters>
	<character>
	<name>Aleria</name>
	<gender>Female</gender>
	<species>Heroleim</species>
	<vocation>Bard</vocation>
	<level>5</level>
	<health>25</health>
</character>
<character>
	<name>Shar</name>
	<gender>Male</gender>
	<species>Human</species>
	<vocation>Merchant</vocation>
	<level>6</level>
	<health>28</health>
</character>
<character>
	<name>Gite</name>
	<gender>Female</gender>
	<species>Aelvar</species>
	<vocation>Mage</vocation>
	<level>7</level>
	<health>18</health>
</character>
<character>
	<name>Horukkan</name>
	<gender>Male</gender>
	<species>Udrecht</species>
	<vocation>Warrior</vocation>
	<level>5</level>
	<health>32</health>
</character>
<character>
	<name>Gounna</name>
	<gender>Female</gender>
	<species>Noleim</species>
	<vocation>Mage</vocation>
	<level>8</level>
	 <health>31</health>
</character>
<character>
	<name>Sheira</name>
	<gender>Female</gender>
	<species>Human</species>
	<vocation>Cleric</vocation>
	<level>4</level>
	 <health>9</health>
</character>
<character>
	<name>Drue</name>
	<gender>Female</gender>
	<species>Voleim</species>
	<vocation>Warrior</vocation>
	<level>6</level>
	<health>32</health>
</character>
<character>
	<name>Paccu</name>
	<gender>Male</gender>
	<species>Human</species>
	<vocation>Merchant</vocation>
	<level>5</level>
	 <health>24</health>
</character>
</characters>
```

Figura 33. Document Characters.xsd associat a l'esquema anterior

Provem amb una “query” simple:

```
for $character in
document('characters.xml')//character
order by health
return
   <healthReport>
      {$character/name}
      {$character/health}
</healthReport>
```

Figura 34. Exemple de consulta XQuery (1)

per obtenir la següent col·lecció:

```
<healthReport>
   <name>Sheira</name>
   <health>9</health>
</healthReport>
<healthReport>
   <name>Gite</name>
   <health>18</health>
</healthReport>
<healthReport>
   <name>Paccu</name>
   <health>24</health>
</healthReport>
<healthReport>
   <name>Aleria</name>
   <health>25</health>
</healthReport>
<healthReport>
   <name>Shar</name>
   <health>28</health>
</healthReport>
<healthReport>
   <name>Gounna</name>
   <health>31</health>
</healthReport>
<healthReport>
   <name>Horukkan</name>
   <health>32</health>
</healthReport>
<healthReport>
   <name>Drue</name>
   <health>32</health>
</healthReport>
```

Figura 35. Resultat de la consulta XQuery (1)

Una altra consulta:

```
let $characters := document('characters.xml')//character
for $level in distinct-values($characters/level)
  order by $level
return
  <level value="{$level}">{
    for $character in $characters
    where $character/level = $level
      order by $character/name
    return $character/name
    }
  </level>
```

Figura 36. Exemple de consulta XQuery (2)

i obtenim:

```
<level value="4">
  <name>Sheira</name>
</level>
<level value="5">
  <name>Aleria</name>
  <name>Horukkan</name>
  <name>Paccu</name>
</level>
<level value="6">
  <name>Shar</name>
  <name>Drue</name>
</level>
<level value="7">
  <name>Gite</name>
</level>
<level value="8">
  <name>Gounna</name>
</level>
```

Figura 37. Resultat de la consulta XQuery (2)

9. Anàlisi dels gestors BD:XML per Schema MPEG-7

Per poder avaluar la idoneïtat dels gestors de bases de dades XML natius per tractar documents basats en l'estàndard MPEG-7, farem un estudi de les capacitats d'aquests gestors utilitzant el llenguatge de consultes XQuery sobre uns documents d'exemple proporcionats pel consultor de l'assignatura, Òscar Celma[6], que són:


opera_aida.xml


opera_don_giovanni.xml

Figura 38. Documents XML Schema MPEG-7 d'exemple (**incrustats en el document)**

Les consultes que farem a continuació s'han realitzat amb el gestors eXist i Xindice i els resultats es mostren just al costat de l'expressió de la consulta en forma d'arxiu vinculat, per, posteriorment analitzar els resultats.

9.1. Consultes XQuery

La metodologia que s'utilitzarà per procedir a les consultes serà la següent:

1. Ens formulem una pregunta del tipus: "Quants cantants hi ha a la òpera Aida?" o bé "Quants tenors hi ha a la base de dades?"….

2. Es munta la Query i s'executa.

3. S'analitza si el resultat és l'esperat.

- Consulta 1.

Obtenir el nom de tots els cantants de l'opera Aida ordenats alfabèticament:

```
for $a in doc
("opera_aida.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Creator
where $a/Role[@href="OpenDramaSinger%"]
order by $a/Agent/Name/FamilyName
return
for $nom_complet in ($a/Agent/Name)
let $nom := $nom_complet/GivenName/text()
let $cognom := $nom_complet/FamilyName/text()
return ($nom, " ", $cognom)
```

Figura 39. XQuery n.1

Adina Aaron
Aldrich Kate
Garra Giuseppe
Enrico Guiseppe Iori
Micaela Patriarca
Paolo Pecchioli
Scott Piper
Stefano Pisani

Figura 40. Resultat de XQuery n.1

- Consulta 2.

Obtenir el nom de tots els cantants de l'opera don Giovanni ordenats alfabèticament amb una Query diferent de l'anterior:

```
for $a in doc
("opera_don_giovanni.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/
Creator
where $a//Role [@href="OpenDramaSinger%"]
order by $a//Agent/Name/FamilyName
return
for $b in ($a//Agent)
let $c := $b//Name
where ($c//GivenName != 'Unknown'
and $c//GivenName != 'Fondazione')
return (string($c//GivenName), string($c//FamilyName))
```

Figura 41. XQuery n.2

Antonio Baglioni
Bassi Luigi
Caterina Bondoni
Giuseppe Lolli
Caterina Micelli
Felice Ponziani
Saporiti Teresa

Figura 42. Resultat de XQuery n.2

- Consulta 3.

Ara, per comprovar la capacitat de recuperar dades de més d'un document, recuperem els cantants de totes dues òperes a la vegada, amb un format de sortida diferent

```
for $a in /Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Creator
where $a/Role[@href="OpenDramaSinger%"]
order by $a/Agent/Name/FamilyName
return
for $nom_complet in ($a/Agent/Name)
let $nom := $nom_complet/GivenName/text()
let $cognom := $nom_complet/FamilyName/text()
return <cantant>{($nom, " ", $cognom)}</cantant>
```

Figura 43. XQuery n.3

```
<cantant>Adina Aaron</cantant>
<cantant>Kate Aldrich</cantant>
<cantant>Antonio Baglioni</cantant>
<cantant>Luigi Bassi</cantant>
<cantant>Caterina Bondoni</cantant>
<cantant>Giuseppe Garra</cantant>
<cantant>Enrico Guiseppe Iori</cantant>
<cantant>Giuseppe Lolli</cantant>
<cantant>Caterina Micelli</cantant>
<cantant>Micaela Patriarca</cantant>
<cantant>Paolo Pecchioli</cantant>
<cantant>Scott Piper</cantant>
<cantant>Stefano Pisani</cantant>
<cantant>Felice Ponziani</cantant>
<cantant>Teresa Saporiti</cantant>
```

Figura 44. Resultat de XQuery n.3

- Consulta 4.

Obtenir les URL (ordenades alfabèticament) de totes les referències a Internet incloses en el document de l'òpera Aida

```
for $a in
doc("opera_aida.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation//RelatedMaterial
order by $a/MediaLocator/MediaUri
return <enllaç>{
$a/MediaLocator/MediaUri
}</enllaç>
```

Figura 45. XQuery n.4

```
<enllaç>
   <MediaUri>http://www.giuseppeverdi.org</MediaUri>
</enllaç>
<enllaç>
   <MediaUri>http://www.grandi-tenori.com</MediaUri>
</enllaç>
<enllaç>
   <MediaUri>http://www.ludwigvanweb.com</MediaUri>
</enllaç>
<enllaç>
   <MediaUri>http://www.musicappreciation.com/opera.htm</MediaUri>
</enllaç>
<enllaç>
    <MediaUri>http://www.opera.it/Operaweb/en/home.html</MediaUri>
</enllaç>
```

Figura 46. Resultat de XQuery n.4

A partir d'aquesta consulta s'ha optat per realitzar-les únicament amb el gestor eXist, ja que els resultats són els mateixos i s'ha decidit que eXist té una interfície més amigable i una major celeritat en la capacitat de resposta a les consultes XQuery que Xindice.

- Consulta 5.

Obtenir cada part de l'òpera Aida amb el temps de duració

```
for $a in
doc("opera_aida.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/TemporalDecomposition/AudioVisualSegment
return ($a/@id ,
$a/MediaTime/MediaDuration/fn:get-minutes-from-dayTimeDuration( text() ), string("minuts"),
$a/MediaTime/MediaDuration/fn:get-seconds-from-dayTimeDuration( text() ), string("segons"))
```

Figura 47. XQuery n.5

```
AID_01_opening
2 minuts 9.0 segons
AID_01_preludio
3 minuts 46.0 segons
AID_01_act1
40 minuts 55.0 segons
AID_01_act2
32 minuts 9.0 segons
AID_01_act3
34 minuts 20.0 segons
AID_01_act4
33 minuts 20.0 segons
```

Figura 48. Resultat de XQuery n.5

- Consulta 6.

Obtenir les Àries de l'òpera Aida amb els cantants que les interpreten

```
for $a in
doc("opera_aida.xml")/Mpeg7/Description/MultimediaContent/AudioVisual/TemporalDecomposition/AudioVisual
Segment/TemporalDecomposition/AudioVisualSegment/TemporalDecomposition/AudioVisualSegment
let $b := $a/CreationInformation/Creation/Title
let $c := $a/CreationInformation/Creation/Creator/Agent/Name
where $a//Name = "Aria"
return (string("titol de l'Aria"),$b/text() , $c)
```

Figura 49. XQuery n.6

```
titol de l'Aria: Se quel guerrier io fossi!
<Name>
   <GivenName>Scott</GivenName>
   <FamilyName>Piper</FamilyName>
</Name>
titol de l'Aria: Ritorna vincitor!
<Name>
   <GivenName>Adina</GivenName>
   <FamilyName>Aaron</FamilyName>
</Name>
titol de l'Aria: Qui Radames verra...
<Name>
   <GivenName>Adina</GivenName>
   <FamilyName>Aaron</FamilyName>
</Name>
titol de l'Aria: Gia i Sacerdoti adunansi
<Name>
   <GivenName>Kate</GivenName>
   <FamilyName>Aldrich</FamilyName>
</Name>
<Name>
   <GivenName>Scott</GivenName>
   <FamilyName>Piper</FamilyName>
</Name>
titol de l'Aria: O terra, addio
<Name>
   <GivenName>Scott</GivenName>
   <FamilyName>Piper</FamilyName>
</Name>
<Name>
   <GivenName>Adina</GivenName>
   <FamilyName>Aaron</FamilyName>
</Name>
<Name>
   <GivenName>Kate</GivenName>
   <FamilyName>Aldrich</FamilyName>
</Name>
```

Figura 50. Resultat de XQuery n.6

- Consulta 7.

Quines són les soprano de les dues òperes?

```
for $a in /Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Creator
where $a/Role[@href="OpenDramaSinger%"]
and $a/Role/Name/@xml:lang = "en"
and $a/Role/Name = "Soprano"
return <soprano>
{$a//Name}
</soprano>
```

Figura 51. XQuery n.7

```
<soprano>
   <Name xml:lang="en" preferred="true">Soprano</Name>
   <Name>
      <GivenName>Adina</GivenName>
      <FamilyName>Aaron</FamilyName>
   </Name>
</soprano>
<soprano>
   <Name xml:lang="en" preferred="true">Soprano</Name>
   <Name>
      <GivenName>Micaela</GivenName>
      <FamilyName>Patriarca</FamilyName>
   </Name>
</soprano>
<soprano>
   <Name preferred="true" xml:lang="en">Soprano</Name>
   <Name xml:lang="it">
      <GivenName xml:lang="it">Teresa</GivenName>
      <FamilyName xml:lang="it">Saporiti</FamilyName>
   </Name>
</soprano>
<soprano>
   <Name preferred="true" xml:lang="en">Soprano</Name>
   <Name xml:lang="it">
      <GivenName xml:lang="it">Caterina</GivenName>
      <FamilyName xml:lang="it">Micelli</FamilyName>
   </Name>
</soprano>
```

Figura 52. Resultat de XQuery n.7

- Consulta 8.

Obtenir els títols de les òperes i els seus compositors

```
for $a in
/Mpeg7/Description/MultimediaContent/AudioVisual/CreationInformation/Creation/Creator
let $b := $a
where $a/Role/@href="OpenDramaRoleCS:authors:composer"
return
for $b in $a/..
return
($a/Agent/Name, $b/Title/text())
```

Figura 53. XQuery n.8

```
<Name xml:lang="it">
   <GivenName xml:lang="it">Guissepe</GivenName>
   <FamilyName xml:lang="it">Verdi</FamilyName>
</Name>
AIDA
<Name xml:lang="de">
   <GivenName xml:lang="de">Wolfgang</GivenName>
   <FamilyName xml:lang="de">Amadeus Mozart</FamilyName>
</Name>
Don Giovanni
Don Juan
```

Figura 54. Resultat de XQuery n.8

- Consulta 9.

Obtenir tots els "tracks" de l'òpera Aida, amb els temps de duració i les localitzacions web on els podem trobar

```
for $a in doc("opera_aida.xml")//AudioVisualSegment
where $a/@id="AID_01_track%"
return
for $b in $a/MediaTime/MediaDuration
return
( $b/../../@id,
$b/fn:get-hours-from-dayTimeDuration( text() ),string("hores"),
$b/fn:get-minutes-from-dayTimeDuration( text() ),string("minuts"),
$b/fn:get-seconds-from-dayTimeDuration( text() ),string("segons"),
string("Localitzacions..."),$b/..//MediaUri/text(),  string ("--------------------------"))
```

Figura 55. XQuery n.9

AID_01_track1.1.1-cd1.3
0 hores 1 minuts 46.0 segons
Localitzacions...
https://mtg127.upf.es/opendrama/media/aida/audio/act1/scene1.1/track1.1.1-cd1.3.mp3
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.1.1-cd1.3.mov
---
AID_01_track1.1.2-cd1.4
0 hores 4 minuts 36.0 segons
Localitzacions...
https://mtg127.upf.es/opendrama/media/aida/audio/act1/scene1.1/track1.1.2-cd1.4.mp3
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.1.2-cd1.4.mov
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.1.2-cd1.4-preview.mov
---
AID_01_track1.1.3-cd1.5
0 hores 2 minuts 47.0 segons
Localitzacions...
https://mtg127.upf.es/opendrama/media/aida/audio/act1/scene1.1/track1.1.3-cd1.5.mp3
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.1.3-cd1.5.mov
---
AID_01_track1.1.4-cd1.6
0 hores 2 minuts 38.0 segons
Localitzacions...
https://mtg127.upf.es/opendrama/media/aida/audio/act1/scene1.1/track1.1.4-cd1.6.mp3
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.1.4-cd1.6.mov
---
AID_01_track1.1.5-cd1.7
0 hores 3 minuts 13.0 segons
Localitzacions...
https://mtg127.upf.es/opendrama/media/aida/audio/act1/scene1.1/track1.1.5-cd1.7.mp3
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.1.5-cd1.7.mov
---
AID_01_track1.1.6-cd1.8
0 hores 2 minuts 49.0 segons
Localitzacions...
https://mtg127.upf.es/opendrama/media/aida/audio/act1/scene1.1/track1.1.6-cd1.8.mp3
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.1.6-cd1.8.mov
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.1.6-cd1.8-preview.mov
---
AID_01_track1.1.7-cd1.9
0 hores 6 minuts 55.0 segons
Localitzacions...
https://mtg127.upf.es/opendrama/media/aida/audio/act1/scene1.1/track1.1.7-cd1.9.mp3
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.1.7-cd1.9.mov
---
AID_01_track1.2.1-cd1.10
0 hores 3 minuts 41.0 segons
Localitzacions...
https://mtg127.upf.es/opendrama/media/aida/audio/act1/scene1.2/track2.1-cd.1.10.mp3
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.2.1-cd1.10.mov
---
AID_01_track1.2.2-cd1.11
0 hores 2 minuts 55.0 segons
Localitzacions...
https://mtg127.upf.es/opendrama/media/aida/audio/act1/scene1.2/track1.2.2-cd1.11.mp3
https://mtg127.upf.es/opendrama/media/aida/video/act1/scene1.1/track1.2.2-cd1.11.mov

…….continua. **Per veure el resultat complet clikeu aquí →** consulta9.txt

Figura 56. Resultat de XQuery n.9

Conclusions

Un cop finalitzada l'exposició del projecte i analitzades les diferents parts del mateix s'ha arribat a les següents conclusions:

Podem dir que XML ja s'ha convertit en un estàndard sòlid i amb un gran futur per davant. XML és, avui dia, el recurs amb més futur a Internet.

XML no és un llenguatge de programació. És un llenguatge estàndard que estableix un format per a la codificació de dades i informació. Les més destacades característiques són el seu conjunt de marques obertes i ampliables, la seva distinció entre la estructura i la presentació de documents, la seva gestió avançada de hipervíncles o la seva modularitat.

També es destacable la capacitat que té XML per contenir informació, de manera estructurada i fàcilment localitzable, i, és per això, que van sorgir eines especialitzades en el tractament d'aquesta informació basada en XML: els gestor de bases de dades XML natius.

Aquest gestors basen tota la seva potència en el recolzament que fan sobre tota la solidesa de XML. Essent aquest metallenguatge molt sòlid i robust, els gestors natius XML només han de dedicar esforços a gestionar i tractar la informació sense la preocupació que suposa comprovar si el document és correcte sintàcticament.

Els esquemes específics d'XML (com **MPEG-7**) són igualment tractables pels gestors de bases de dades natius XML ja que, en essència, un document amb un esquema determinat compleix les regles estrictes d'un document XML.

En concret, els gestor analitzats en el present treball, **eXist** i **Xindice**, són una bona mostra del funcionament d'aquests programes.

Els llenguatges analitzats per realitzar les cerques en els documents XML (**XPath**, **XPath 2.0** i **XQuery**, basat en aquest últim), són llenguatges de consultes relativament joves i, penso que encara han d'evolucionar fins un estadi en el que no sigui tant complex realitzar cerques. Ara per ara, he trobat dificultats per utilitzar XQuery i poder realitzar consultes relativament complexes. Malgrat tot, es tracta d'eines potents i versàtils que amb un profund coneixement poden ajudar-nos en la recerca d'informació a través de documents XML.

Bibliografia i referències

- **Charles F. Goldbarg**, **Paul Prescod**. *The XML handbook*. Ed. Prentice Hall PTR, 2000
- **Elliotte Rusty Harold**. *XML Bible*. Ed. Hungry Minds, Inc, 1999
- http://geneura.ugr.es/~victor/cursillos/xml/XPath/
- http://www.brics.dk/~amoeller/XML/
- http://www.ipedo.com/html/XQuery/XQuery_tutorial/
- http://www.kurtcagle.net/schemas/
- http://www.mulberrytech.com/quickref/,
- http://www.samspublishing.com/articles/
- http://www.w3.org/TR/XPath
- http://www.w3.org/TR/XQuery/
- http://www.xml.com.ve/que_es_XQuery.htm
- http://www.xmlsoftware.com/XPath/
- http://www.zvon.org/HTMLonly/XPathTutorial/General/examples.html
- http://www-106.ibm.com/developerworks/es_es/xml/library/x-XQuery.html

Referències

[1]. http://www.exist-db.org
[2]. http://xml.apache.org/xindice/
[3]. http://www.w3.org/
[4]. http://www.iua.upf.es/mtg/opendrama/mpeg7/docs/descriptionExample000.xml
[5]. http://www.kurtcagle.net/schemas/
[6]. ocelma@uoc.edu),