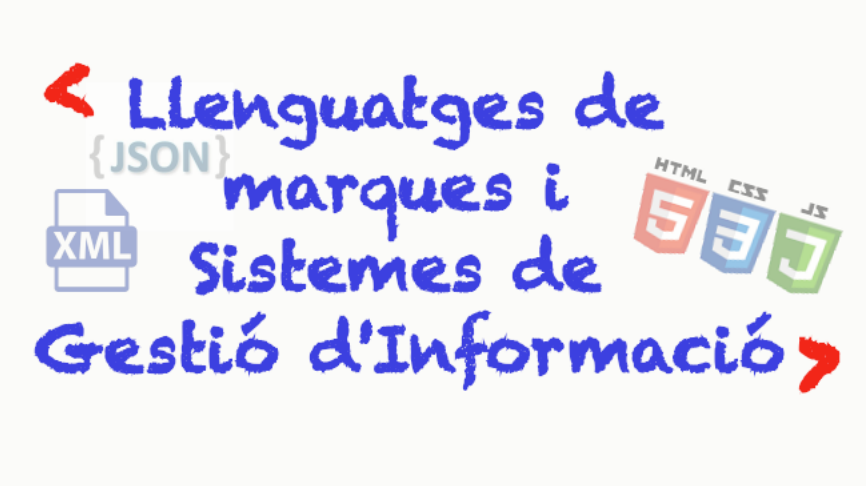
UNITAT 7. EMMAGATZEMAMENT D'INFORMACIÓ



IES Sant Vicent Ferrer
Algemesí

Contingut

1. Introducció

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

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

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

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

- Com que majoritàriament les organitzacions ja tenen sistemes d'emmagatzematge organitzats de dades basats en dades relacionals una possible solució podria ser **convertir els fitxers XML a dades relacionals**:
    - L'emmagatzematge de dades serà totalment organitzat i centralitzat en un punt on ja hi ha les altres dades de l'organització.
    - El seu llenguatge de consulta, SQL, és molt conegut i es fa servir molt, de manera que no caldrà que els usuaris aprenguin un nou llenguatge.
- Una altra solució consisteix a **crear un sistema d'emmagatzematge pensat només per als fitxers XML**. Són els sistemes coneguts com a **bases de dades natives XML**. Aquests:
    - Proporcionen un lloc per emmagatzemar ordenadament fitxers XML.
    - Tenen el seu llenguatge de consulta propi: **XQuery**.

2. Bases de dades relacionals

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

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

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

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

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

2.1.1. Convertir les dades XML en relacionals

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

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

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

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

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

2.1.2. Sistemes relacionals amb extensions XML

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

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

Com que SQL no tenia suport per a XML el 2003, es va modificar l'estàndard del llenguatge SQL per afegir l'extensió SQL/XML.

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

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

3. XQUERY

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

Es tracta d'un llenguatge funcional, de manera que en comptes de dir-li quines són les passes per fer una tasca el que es fa és avaluar les expressions contra el fitxer XML i generar un resultat.

A diferència dels llenguatges de programació habituals, en XQuery s'especifica què és el que es vol i no la manera com ho ha de fer per obtenir-ho.

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

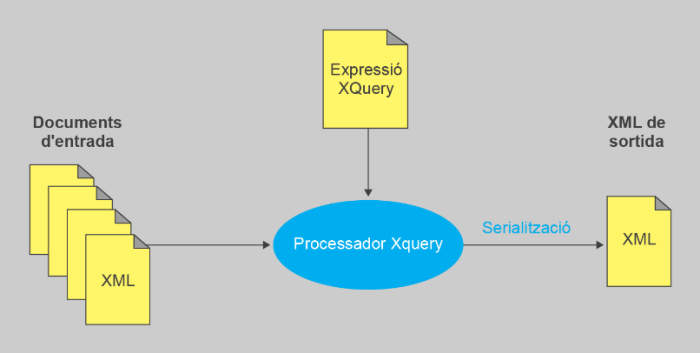
- Seleccionar la informació segons criteris. Ordenar, agrupar, afegir dades.
- Filtrar la informació que es vulgui del flux de dades.
- Cercar informació en un document o en un grup de documents.
- Unir dades de múltiples documents.
- Transformar i reestructurar XML.
- No estar limitat a la cerca, ja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

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

3.1. Processadors XQuery

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

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



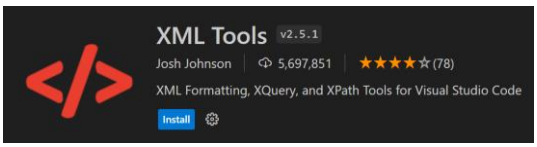
1 Funcionament dels processadors XQuery

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

3.1.1. XQUERY a Visual Studio Code

Podem utilitzar l'editor que hem fet servir en altres unitats d'aquest curs, Visual Studio Code, si instal·lem extensions que donen suport a Xquery

Una de les extensions més popular per dur a terme aquesta tasca és XML Tools, de l'autor Josh Johnson, que ja hem fet servir per avaluar expressiones Xpath.



2 XML Tools per a VS Code

3.2. Consultes XQuery

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

```
<Hola />
```

En executar aquesta ordre el resultat serà el següent:

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

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

```
<a>
```

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

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

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

El resultat d'executar-ho amb XQuery serà:

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

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

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

El return s'executaria més d'una volta:

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

3.3. Comentaris

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

```
(: Comentari :)
```

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

3.4. Càrrega de documents

Una de les coses que caldrà per interrogar un fitxer o fitxers XML és establir específicament a quins documents es vol fer la petició.

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

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

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

XQuery conté completament XPath, de manera que es pot fer servir qualsevol expressió XPath per filtrar resultats quan faça falta, i per tant es pot fer que en el mateix procés de càrrega s'hi apliqui un filtre. Per exemple una expressió XQuery que retorna quins són els noms dels alumnes del fitxer alumnes.xml pot ser la següent:

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

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

```
return doc("http://direccio.org/alumnes.xml")
```

3.5. Variables

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

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

```
let $x := 5

let $y := 4

return $x + $y
```

El resultat serà:

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

9
```

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

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

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

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

Es poden aplicar filtres XPath a les variables:

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

Els valors de les variables els podrem comparar amb els operadors de comparació habituals:

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

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

3.6. Expressions avaluables

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

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

```
<html>
<head><title>Llista de classe</title></head>
<body>
<h1>
{
doc("classe1.xml")//assignatura
}
</h1>
</body>
</html>
```

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

3.6.1. Creació d'elements i atributs

Fent servir les expressions avaluables és fàcil crear nous elements.

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

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

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

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

```
element modul {
}
```

Crearia un element <modul> buit:

```
<modul />
```

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

```
element modul {
    attribute nom {"Llenguatges de marques"},
    element alumnes { }
}
```

Generarà el següent:

```
<modul nom="Llenguatges de marques">
    <alumnes />
</modul>
```

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

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

3.7. Expressions FLWOR

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

Caràcter	Comando	Ús
f	for	Permet passar d'un element a un altre de la seqüència
l	let	Serveix per assignar valors a una variable
w	where	Permet filtrar els resultats segons condicions
o	order by	Usat per ordenar els resultats abans de mostrar-los
r	return	Retorna el resultat de tota l'expressió

for ... return

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

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

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

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

Generarà:

```
Hola Adéu
```

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

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

Que retorna:

```
1 4 9
```

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

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

El resultat serà:

```
<?xml version="1.0" encoding="UTF-8"?>
  <alumne>
     <cognom>Pi</cognom>
  </alumne>
  <alumne>
     <cognom>Garcia</cognom>
  </alumne>
```

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

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

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

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

for ... at

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

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

Mostrarà:

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

"for" amb múltiples variables

Fins ara només s'ha fet servir una variable en el for però es poden posar tantes com faça falta separant-les per comes:

```
<resultats>
{
for $tot in doc("classe.xml")/classe,
    $nom in $tot/alumnes,
    $modul in $tot/assignatura/text()
return
   <modul nom="{ $modul }"
          alumnes="{ count($nom/alumne)}">
         { $nom/alumne/nom }
   </modul>
}
</resultats>
```

Que donarà el resultat següent:

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

"for" niuat

Les ordres for es poden imbricar entre elles d'una manera similar a com ho fa SQL per poder aconseguir resultats més complexos que es basen en els resultats obtinguts anteriorment. Exemple:

```
for $selec in doc("classe.xml")//alumne
return
<persona>
    { $selec/nom }
    {
        for $selec2 in $selec
        return $selec2//cognom
    }
</persona>
```

let

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

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

En les expressions FLWOR pot haver tants let com calga.

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

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

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

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

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

Mostrarà:

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

I en canvi amb let:

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

Mostrarà:

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

Els resultats són bastant diferents, es pot veure clarament que let ha tractat tots els valors junts.

Condicions

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

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

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

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

En un **where** pot haver tantes condicions com calga simplement encadenant-les amb les operacions lògiques **and**, **or** o **not()**

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

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

A partir de dos fitxers amb les dades de dues classes diferents es poden obtenir només els alumnes que es repeteixen entre les dues classes amb:

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

Ordenació

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

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

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

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

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

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

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

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

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

Alternatives

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

```
let $doc := doc("classe1.xml")
let $aprovat := 5
for $b in $doc//alumne
return
if ($b/nota >= 5) then
<data>
{ $b/nom/text() } està aprovat
</data>
else
<data>
{ $b/nom/text() } no està aprovat
</data>
```

Això generarà el resultat següent:

```
<data>Frederic està aprovat</data>
<data>Filomeno no està aprovat</data>
<data>Manel està aprovat</data>
```

La instrucció **else** és habitual en llenguatges de programació, però a diferència del que passa normalment, en XQuery, encara que no es vulga fer res, s'ha d'especificar **else** obligatòriament. **Si no es vol que faça res es deixa en blanc.**

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

4. Bases de dades natives XML (NXD)

A diferència del que passa amb les bases de dades relacionals, que fa molts anys que funcionen i tenen darrere seu una base teòrica important, les NXD no tenen uns estàndards definits per fer les coses i la teoria que les suporta no està definida. Això fa que sovint cada base de dades faça les coses d'una manera que pot ser totalment diferent de com ho fa una altra.

Les bases de dades natives XML es fan servir sobretot per emmagatzemar dades que contenen:

- Contingut narratiu.
- Que són menys previsibles que les que s'emmagatzemen normalment en bases de dades relacionals.
- Que han de generar sortides per a entorns web.
- Que s'han de transportar d'un sistema a un altre.

Els sistemes NXD són molt heterogenis i sovint s'estan desenvolupant en direccions que no sempre coincideixen. Tot i la dispersió, sempre podem trobar punts en comú en la majoria de les bases de dades.

Llenguatge de consulta

Tots els sistemes NXD suporten com a mínim **XPath** com a llenguatge de consultes, tot i que la tendència general és **adoptar XQuery**; és lògic si es té en compte que XPath no va ser pensat per ser un llenguatge de consulta de base de dades, mentre que XQuery sí.

Tots els sistemes suporten algun tipus d'interfície que permet accedir-hi per mitjà d'algun llenguatge de programació, com ara la interfície XML:DB, o bé per mitjà d'algun protocol de xarxa com HTTP, WebDAV, SOAP…

Organització i emmagatzematge

L'organització més habitual de les dades és mitjançant col·leccions de documents XML que sovint intenten generar una estructura semblant a la que ofereixen els directoris dels sistemes operatius. En aquests sistemes una col·lecció és com una carpeta que conté un conjunt de documents XML o fins i tot altres carpetes

El tractament de les col·leccions sol ser una de les diferències que hi ha entre bases de dades XML. Algunes forcen que dins d'una col·lecció hi haja documents del mateix tipus, mentre que altres permeten que es pose qualsevol tipus de document.

L'emmagatzematge de les dades sol ser el que diferencia més les NXD, ja que la naturalesa dels fitxers XML no els fa ideals per ser emmagatzemats de manera eficient (tenen repeticions, ocupen molt d'espai…). Per aquest motiu totes les NXD intenten optimitzar d'alguna manera aquest emmagatzematge: convertint les dades en fitxers binaris (amb simples compressions de dades o codificant-les), creant estructures d'arbre, o simplement emmagatzemant els fitxers XML sense fer-hi res confiant que la reducció de costos de l'emmagatzematge farà que ocupar espai no sigui un problema.

Indexació

A l'hora de fer servir una base de dades el que importa realment a l'usuari és que faça la cerca en un temps acceptable. Els fitxers de text no són la millor manera d'emmagatzemar dades si l'objectiu és fer cerques, i per tant les bases de dades XML s'han esforçat a intentar crear algun sistema per fer aquestes cerques més eficients.

El més habitual sol ser:

- Organitzar de manera eficient les col·leccions
- Fer servir índexs

Seguretat

La seguretat és un altre aspecte per tenir en compte en els sistemes NXD. Generalment en un entorn empresarial no tothom pot accedir a les dades que vulga sinó que a la informació poden accedir només les persones autoritzades.

La identificació d'usuaris permet implementar els mecanismes bàsics per permetre controlar a quins documents pot accedir cada un dels usuaris de l'organització.

Actualitzacions de les dades XML

La falta d'un sistema d'actualitzacions dels fitxers XML en la primera versió **d'XQuery** ha fet que molt sovint els sistemes NXD, en comptes d'actualitzar les dades que han canviat, opten per substituir totalment el document cada vegada que hi ha canvis.

4.1. eXist-db

De les bases de dades XML de codi obert disponibles, probablement **eXist-db** és la que està més preparada per ser usada en un entorn professional. Es tracta d'una **base de dades nativa XML** que està desenvolupada en llenguatge Java, i que:

- Permet fer operacions amb els llenguatges XQuery 1.0, XPath 2.0 i XSLT 2.0.
- Pot funcionar tan com una biblioteca que s'incorpora als programes com ser executada per mitjà d'un servidor.
- Es pot accedir a la base de dades per mitjà de múltiples interfícies estàndards com REST, WebDAV, SOAP, XML-RPC o ATOM.
- Permet actualitzacions de dades per mitjà de l'API XMLDB, XUpdate i XQuery Update Facility.
- Porta incorporat un sistema de gestió d'usuaris de la base de dades i dels permisos d'accés a les col·leccions semblant al dels sistemes Unix

L'organització de les dades és per mitjà de col·leccions de fitxers XML que s'organitzen d'una manera pràcticament idèntica a com es fa en els fitxers d'un sistema operatiu. Igual que en els sistemes operatius, es poden fer col·leccions dins de col·leccions i s'hi pot navegar amb un sol clic.

Instal·lació

La instal·lació de l'última versió estable (actualment -abril 2024- 6.2.0) es pot fer descarregant el paquet d'instal·lació per al sistema operatiu desitjat des del repositori oficial de github (https://github.com/eXist-db/exist/releases/tag/eXist-6.2.0) o utilitzant el contenidor docker oficial (https://hub.docker.com/r/existdb/existdb/tags/)

En la instal·lació per defecte a més de l'eXist s'hi instal·la el servidor Jetty, que permetrà accedir a la gestió web del sistema amb un navegador en l'adreça http://localhost:8080/exist/.

Si es fa servir el servidor, l'accés a les consultes i l'administració del sistema es pot fer per mitjà d'un client Java o bé per mitjà de l'entorn web.

Tant el client com el client web disposen d'un entorn per poder elaborar consultes XQuery.

4.2. Altres bases de dades NXD

BaseX

BaseX es una base de dades XML nativa i una eina de processament de documentos XML. Aquest software, que és de codi obert i multiplataforma, proporciona un motor de consulta complet per a XPath, XQuery i XSLT.

Més informació: https://basex.org/

Fonto

Fonto es una eina online que permet realizar consultes XQuery 3.1 contra un documento XML.

Es molt senzilla d'utilitzar:

- En primer lloc, copiem el documento XML en la part esquerra (***XML input***). També permet importar un fitxer.
- Després, seleccionem l'opció XQuery 3.1 en la parte superior esquerra (desplegable, per defecte està configurada amb XPath 3.1).
- Por últim, indiquem la consulta XQuery en el quadre de text ***XPath/XQuery expression***.

Els resultats (***Query results***) se mostraran en la part dreta de forma instantània.

Més informació i accés: https://xpath.playground.fontoxml.com/

5. Enllaços d'interés

XQuery a **w3schools**: https://www.w3schools.com/xml/xquery_intro.asp

Especificació XQuery a w3: https://www.w3.org/TR/xquery-31/

6. Bibliografia

Sala, Xavier. (2023) Llenguatges de marques i sistemes de gestió d'informació.

Institut Obert de Catalunya