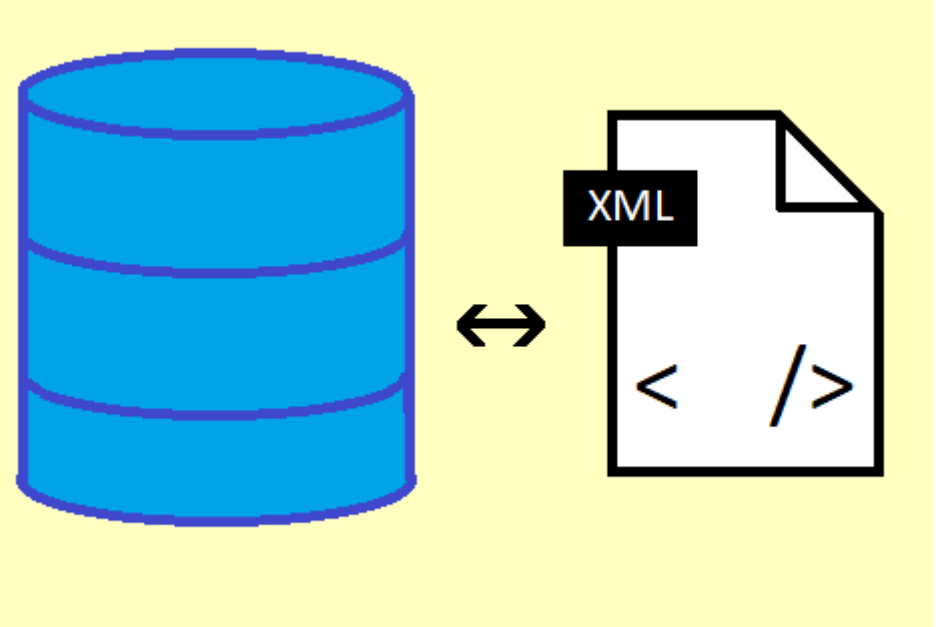
CFGS Desenvolupament d'Aplicacions Multiplataforma (DAM)
Mòdul 6 – Accés a Dades.

UF 3. Persistència en BD natives XML

Resum bases de dades XML natives



Índex de contingut

Què és una base de dades XML nativa

Una base de dades nativa és una base de dades dissenyada especialment per a l'emmagatzematge de documents en format XML i que compleix:

- El model lògic es basa en documents XML (a diferència, per exemple, de les relacionals, que es basen en dades).
- El mecanisme d'emmagatzematge subjacent roman transparent a qui utilitza la base de dades. El mecanisme d'emmagatzematge subjacent sol ser una combinació de mecanismes tradicionals.
- No hi ha limitació en la complexitat i el nombre de nivells a les dades.
- Permet l'ús de tecnologies de consulta i transformació pròpies d'XML: XPath, XQuery...
- Compleix les característiques ACID:
  - Atomicitat: una operació o es realitza tota o no es realitza (no pot realitzar-se parcialment).
  - Consistència: no es perd mai la integritat de les dades.
  - Aïllament: no hi ha interferències entre operacions sobre les mateixes dades.
  - Durabilitat: un cop executada una operació, aquesta no es pot desfer sota cap circumstància.

Conceptes bàsics

Un SGBD XML-natives ha de ser capaç de validar els documents XML a partir de documents de tipus XSD o DTD, que descriuen formalment les caracaterístiques que aquells han de tenir.

Idealment, en una base de dades XML-nativa els documents s'haurien d'agrupar en col·leccions. Aquestes col·leccions haurien de poder contenir, a més de documents, altres col·leccions. D'altra banda, un SGBD-XML natives hauria de poder contenir diferents bases de dades.

Molts SGBD-XML natives, però, tenen limitacions en alguna o algunes d'aquestes caracterísitiques, com s'indica a la taula «Quadre resum SGBD-XML natives del material».

Tipus de documents XML:

- Centrat en dades (data-centric): estructura regular; format ben definit. Solen usar-se per a intercanvi entre plataformes. El seu origen sol ser una base de dades tradicional. També solen emmagatzemar-se en una base de dades tradicional, encara que pot fer-se també en una base de dades XML-nativa o XML-habilitada.
- Centrat en el document (document-centric): estructura irregular; format poc estricte i poc definit. El seu origen sol ser un autor humà. El seu ús en bases de dades XML natives és més propi que al cas dels documents centrats en dades. També s'emmagatzemen en sistemes gestors de continguts.

SGBD-XML habilitats: SGBD no XML que ofereix a l'usuari característiques per a treballar amb documents XML. Són menys flexibles que els SGBD-XML natives, tot i que cada vegada les seves prestacions s'acosten més a les dels SGBD-XML natives.

Avantatges i inconvenients dels SGBD-XML natives en comparació amb els SGBD no XML natives:

- Avantatges:
  - No necessiten mapatge des del format XML a l'organització pròpia de les dades.
  - Solen incorporar un motor de cerca d'alt rendiment.
  - Proporcionen facilitats per a la gestió de les dades XML i dades heterogènies en general.
  - Conserven la integritat dels documents.

- Inconvenients:
  - Gran necessitat d'espai per emmagatzemar les etiquetes.
  - Només poden emmagatzemar i retornar dades en format XML.
  - Dificultat per generar noves estructures a partir de la informació existent (per exemple: relacionar dos tipus d'informació vinculades per una dada comuna).
  - Dificultats en la indexació.
  - Dificultats per fer canvis en un document XML sense haver de modificar tot el document.

Estratègies habituals d'emmagatzematge dels SGBD-XML natives:

- Estratègia basada en text: s'emmagatzema el document XML com un text i es mantenen índexs per optimitzar les consultes que rebin.
- Estratègia basada en el model de les dades de la base de dades subjacent: es realitza un mapatge des d'XML a models d'altres sistemes (per exemple, model relacional, model OO, etc.).
- Emmagatzematge desenvolupat específicament per a la gestió de documents XML.

Algunes estratègies d'emmagatzematge de documents XML tant en SGBD-XML natives com en SGBD-habilitades com, inclús, directament en el sistema de fitxers del disc, presenta algunes dificultats:

- Emmagatzemar els documents directament en el sistema d'arxius o en camps de tipus BLOB (Binary Large OBjects) d'un SGBD:
  - Cal gestionar el document com un tot.
  - Problemes de seguretat si s'utilitza directament el sistema d'arxius.
- Al cas d'utilitzar una base de dades tradicional, cal un mapatge XML-model de la base de dades. Aquest mapagte presenta diferents dificultats, depenent del tipus de gestor de base de dades:
  - SGBD jeràrquic: el model és jeràrquic, igual que el dels documents, però l'estructura de la base de dades no té la flexibilitat adient per emmagatzemar-hi documents.
  - SGBD relacional: el model és molt diferent, com es veu a la taula que segueix.

Model relacional	Model XML
Es basa en dades bidireccionals (taules[1])	Es basa en arbres jeràrquics (documents)
Ordre rellevant	Ordre no rellevant
En una taula no pot haver files repetides	En un document pot haver dades repetides
Relacions entre les dades clares	Relacions entre dades no sempre òbvies
Recuperació del document XML original complexa, a partir de les taules. És especialment complex determinar què era un atribut i què un element.	Treballa directament amb documents XML

1 Seria més purista parlar de relacions i no pas de taules; s'ha triat el terme 'taules' per evitar confusions entre les dades i les relacions entre aquestes (a través de claus foranes).

Es sol fer el mapatge XML-relacional seguint la següent correspondència (agafada de la taula 1.1. del material):

Esquema XML	Esquema relacional
Element	Taula
Atribut / Element imbricat	Columna
Atribut ID	Clau primària
IDREF / Element imbricat	Clau secundària
#REQUIRED, #IMPLIED	NOT NULL, NULL

- SGBD Orientat a Objecte: el fet que el model sigui més flexible fa més senzill el mapatge. No obstant, de manera semblant al cas anterior, tampoc hi ha ordre entre els objectes.

API estàndards de les bases de dades natives: són API (interfície de programació d'aplicacions) definides per un organisme o grup de treball no dependent de cap fabricant concret i que solen ser suportades per la majoria de fabricants.

Al cas de bases de dades natives, per al llenguatge Java n'hi ha dues:

- XML:DB: també anomenada XAPI. Va ser creada el 2000 pel grup XML:DB. Ha quedat obsoleta.
- XQuery API for Java: també anomenada XQJ. Pensada per utilitzar-se amb els llenguatges XQuery i XUpdate.

A part de les API estàndards, cada SGBD pot tenir la seva pròpia API -que anomenem API específica del SGBD- , amb característiques diferents de les API de la resta de SGBD.

Quadre resum SGBD-XML natives del material

Base de dades	API Java específica	API estàndard suportades		Llenguatges de consulta	Llenguatges d'actualització	Bases de dades	Col·leccions
		XML:DB	XQJ				
BaseX	Sí	No	Sí (Foster)	XPath XQuery	XQUF	Tracta totes dues coses com a equivalents. Admet diferents bases de dades, però 1 sol nivell.	
eXist-db	No	Sí	Sí (Foster)	XPath XQuery	Update XUpdate	Sí, però no-més admet una BD per cada SGBD	Sí. Poden tenir subcol·lec-cions.
Sedna	Sí	Sí (Foster)	Sí (Foster)	XPath XQuery	Update XUpdate	Sí. Admet diferents BD per cada SGBD	Sí, però només d'un nivell.

Important

- Les biblioteques que implementen l'API XQJ ha estat realitzada per Charles Foster a tots tres SGBD.

- El llenguatge d'actualització de dades XUpdate només està disponible quan s'utilitza l'API XML:DB. De fet, és el llenguatge propi d'aquesta API.

- El llenguatge d'actualització de dades XQUF és promogut pel W3C.

Introducció a XPath

XPath especifica un atribut o un element tot indicant el camí dins del document per trobar-lo, respectant l'estructura jeràrquica del document i separant els elements per una o dues barres (/). Aquest camí pot ser absolut o relatiu. Si és absolut, comença amb una barra. Si és relatiu, comença sense barra. Els camins relatius s'utilitzen per indicar algun atribut o element dins d'una consulta més gran i parteixen del punt que indica aquesta consulta que els engloba.

Per exemple, si tenim el document *grup.xml* amb la següent informació:

```
<?xml version="1.0" encoding="windows-1250"?>
<classe>
   <estudiant id = "393">
      <nom>Daniel</nom>
      <cognoms>Pi Gran</cognoms>
      <nota>85</nota>
          <aficions>
                <aficio>Lectura</aficio>
                <aficio>Cinema</aficio>
          </aficions>
          <naixement>
                <dia>3</dia>
                <mes>5</mes>
                <any>1990</any>
          </naixement>
   </estudiant>

   <estudiant id = "493">
      <nom>Vanessa</nom>
      <cognoms>Pou Salat</cognoms>
      <nota>95</nota>
          <aficions>
                <aficio>Lectura</aficio>
                <aficio>Esport</aficio>
                <aficio>Cinema</aficio>
          </aficions>
          <naixement>
                <dia>3</dia>
                <mes>1</mes>
                <any>1995</any>
```

```
        </naixement>
   </estudiant>

   <estudiant id = "593">
      <nom>Xavier</nom>
      <cognoms>Fuster Cantó</cognoms>
      <nota>90</nota>
          <aficions>
                <aficio>Esport</aficio>
                <aficio>Cinema</aficio>
          </aficions>
          <naixement>
                <dia>5</dia>
                <mes>5</mes>
                <any>1989</any>
          </naixement>
   </estudiant>
</classe>
```

i:

Volem recuperar:	Cal escriure	Observacions
Tota la classe	/**classe**	El resultat és **<classe>....</classe>**
Tots els estudiants	/**classe**/**estudiant**	El resultat és **<estudiant>...</estudiant>** tantes vegades com estudiants contenen les dades.
Els noms de tots els estudiants	/**classe**/**estudiant**/**nom**	El resultat és **<nom>...</nom>** tantes vegades com noms hi ha
Els identificadors dels estudiants	/**classe**/**estudiant**/**@id**	Els atributs es denoten amb @, mentre que els elements s'escriuen sense cap indicador especial. El resultat és **id**="..." tantes vegades com *id* hi ha.

De manera opcional es pot escriure al davant de cada consulta el document i la col·lecció. Si posem al davant el document i/o la col·lecció, la cerca es restringeix a aquests. En cas contrari, la cerca es fa per tota la base de dades.

El document i la col·lecció es posen al davant com a dos elements més del camí (separats de la resta, per tant, per una o dues barres). Els seus noms van tancats, respectivament, per les funcions *doc* i *collection*. L'ordre de *doc* i *collection* és intercanviable.

Així, per exemple, si el document es diu *grup.xml* i la col·lecció *alumnat*, la segona consulta podria escriure's així:

doc("grup.xml")/classe/estudiant	Si només volem especificar el document
collection("alumnat")/classe/estudiant	Si només volem especificar la

	col·lecció
doc("grup.xml")/collection("alumnat")/classe/estudiant	Si volem especificar document i col·lecció

A BaseX, els conceptes de base de dades i col·lecció coincideixen. A més, davant del nom del document cal indicar a quina col·lecció pertany. Per tant, els exemples anteriors quedarien així:

doc("alumnat/grup.xml")/classe/estudiant	Si només volem especificar el document
collection("alumnat")/classe/estudiant	Si només volem especificar la col·lecció
doc("alumnat/grup.xml")/collection("alumnat")/classe/estudiant	Si volem especificar document i col·lecció

A partir d'ara, per abreujar els exemples, s'ometrà el document i la col·lecció.
No obstant, si volguéssim fer les consultes exactes del tot, caldria posar-los.

Funcions

Conjuntament amb aquestes expressions poden utilitzar-se funcions. Exemples de funcions són:

- count(x): compta el nombre d'elements d'x; x és un camí XPath absolut o relatiu. En el segon cas, sol indicar un element o un atribut d'un node.
- string(): passen a cadena de caràcters l'element o atribut; es posa a continuació de l'últim element de l'expressió XPath.

Exemples d'utilització d'aquestes funcions:

count(/classe/estudiant)	A l'exemple, retorna 3, ja que només hi ha 3 estudiants.
/classe/estudiant/aficions/count(aficio)	A l'exemple retorna 2 3 2, que és el nombre d'aficions de cadascun dels estudiants.
/classe/estudiant/nom/string()	A l'exemple retorna Daniel Vanessa Xavier que són els noms de cadascun dels estudiants (sense /string() retornava el mateix, però cada nom anava delimitat per <nom>... </nom>
/classe/estudiant/@id/string()	A l'exemple retorna 393 493 593, que són els identificadors de cadascun dels estudiants (sense /string() retornava el mateix, però entre comets i precedit del text id=)

XPath té moltes més funcions. Podeu trobar-les aquí: http://www.w3.org/TR/xpath-functions-31/

Seleccionar

XPath també permet seleccionar nodes. Es fa posant la condició de selecció entre claudàtors, **[ ]** , i indicant dins dels claudàtors la condició que s'ha de complir. Per indicar els elements que

intervenen a la condició també s'utilitzen camins relatius (és a dir, que comencen al punt on posem la condició).

Exemples d'ús de la selecció són:

`/classe/estudiant[nom="Daniel"]`	Retorna totes les dades de l'estudiant de nom Daniel en format XML.
`/classe/estudiant[nom="Daniel"]/naixement`	Retorna la data de naixement de l'estudiant de nom Daniel en format XML.
`/classe/estudiant[@id<500]`	Retorna les dades en format XML de tots els estudiants amb id més petit que 500.
`/classe/estudiant[aficions/aficio="Lectura"]/nom`	Retorna en format XML els noms de tots els estudiants aficionats a la lectura.
`/classe/estudiant[aficions/aficio="Lectura"]/nom/string()`	Retorna els noms (sense el format XML) de tots els estudiants aficionats a la lectura.
`/classe/estudiant[nota>90 or @id<400]/nom/string()`	Retorna el nom (sense format XML) de tots els alumnes amb una nota major a 90 o un identificador inferior a 400.
`/classe/estudiant[nom="Daniel" or nota>90]/nom/string()`	Retorna el nom (sense format XML) de tots els alumnes anomenats Daniel o amb una nota superior a 90.
`/classe/estudiant[naixement/mes=5 and naixement/any<1990]/nom/string()`	Retorna el nom (sense format XML) de tots els alumnes nascuts el mes de maig i abans de 1990

Introducció a XQuery

XQuery pot veure's com una potent extensió d'XPath.

Concretament, hi afegeix les característiques FLWOR, que és un acrònim de:

- For: permet recórrer el resultat d'una expressió XPath amb un for i retornar un valor per a cada element d'aquest resultat.
- Let: permet utilitzar variables (en alguns llenguatges de programació, *let* s'utilitza per assignar valors a variables).
- Where: permet seleccionar o filtrar determinats nodes de l'expressió XPath amb una sintaxi similar al *WHERE* d'SQL. Els efectes són equivalents a l'ús de claudàtors en les expressions XPath.
- Order: permet ordenar els resultats.
- Return: permet retornar un valor per a cada element resultant d'avaluar l'expressió XPath.

Veiem variants dels exemples anteriors tot utilitzant aquestes noves característiques:

`for $i`
`  in`
`  /classe/estudiant[nom="Daniel"]/nom/string()`
`  return $i`
	Equival a:
	`/classe/estudiant[nom="Daniel"]/nom/string()`

`for $i in /classe/estudiant`
`  where $i/nom="Daniel"`
`  return $i/nom/string()`
	Equival a l'anterior, però la condició de selecció s'ha passat a la clàusula WHERE.
	Cal fixar-se en l'ús que es fa de la variable $i: $i fa referència a cadascun dels estudiants (un a cada volta del for); per això, per indicar el nom de cada estudiant cal escriure $i/nom

`for $i in /classe/estudiant`
`  where $i/naixement/any<="1990"`
`  order by $i/cognoms,$i/nom`
`  return $i/nom/string()`
	Retorna tots els noms dels estudiants nascuts al 1990 o abans i ordenats per cognoms i nom. Mostra directament els noms, sense l'embolcall XML.

Introducció a Update i XQUF

Update i XQUF són dues extensions d'XQuery que afegeixen a aquest llenguatge de consultes la possibilitat de realitzar també modificacions sobre les dades de la base de dades.

Update és anterior a XQUF. Va ser dissenyat per Patrick Lehti. XQUF (XQuery Update Facility) és suportada pel W3C (World Wide Web Consortium).

Com totes dues tenen una sintaxi molt semblant, pot desenvolupar-se l'essencial de manera simultània. A tots els casos, *element* és un fragment XML que conté dades i *ubicació* és una expressió XPath que serveix com a referència per realitzar l'actualització.

El llenguatge XUpdate va lligat a l'API XML:DB, que ha quedat obsoleta.

Inserció d'un element

Amb **Update:** **update insert** ***element*** **{ into | following | preceding }** ***ubicació***

La inserció es realitza una vegada per cadascun dels elements resultat d'aplicar la consulta XPath que defineix la ***ubicació***; per exemple, si estem insertant una afició nova i ***ubicació*** fa referència a tots els alumnes, a cada alumne se li afegiria aquesta afició.

Amb **XQUF** **insert node** ***element*** **{into | as first into | as last into | before | after}** ***ubicació***

ubicació ha de fer referència a un únic element.

Donen d'alta l'*element* a la ***ubicació*** indicada. Cal indicar un dels mots que van dins la clau. Aquests mots signifiquen el següent:

- **Update**
  - **into** dins de l'element especificat per la ubicació, després dels fills que conté actualment;
  - **following** a continuació de l'element o elements especificats per la ubicació;
  - **preceding** just abans de l'element o elements especificats per la ubicació;
- **XQUF**
  - **into** dins de l'element especificat per la ubicació, en una posició respecte als fills actuals que depèn de la implementació;
  - **as first into** / **as last into:** dins de l'element especificat per la ubicació, abans o després, respectivament, dels fills que conté actualment;
  - **before** / **after**: just abans o just després, respectivament, de l'element especificat per la ubicació.

Exemple: afegir l'afició "Gastronomia" a l'estudiant amb id = 393 al final de les seves aficions.

Amb **Update:**

```
update insert <aficio>Gastronomia</aficio> into /classe/estudiant[@id="393"]/aficions
```

Amb **XQUF:**

```
insert node <aficio>Gastronomia</aficio> as last into /classe/estudiant[@id="393"]/aficions
```

A tots dos, l'identificador pot posar-se amb o sense cometes.

Esborrat d'un element

Amb **Update:** update delete *ubicació*
Amb **XQUF:** delete node *ubicació*

Esborren l'element(s) indicat(s) per la ***ubicació***

Exemple: suprimir l'afició "Lectura" a l'estudiant amb id = 393

Amb **Update:**

```
update delete /classe/estudiant[@id="393"]/aficions/aficio[.="Lectura"]
```

Amb **XQUF:**

```
delete node /classe/estudiant[@id="393"]/aficions/aficio[.="Lectura"]
```

A tots dos, el punt fa referència a l'element actual, en aquesta cas, a una afició.

A tots dos, l'identificador pot posar-se amb o sense cometes.

Modificar un element

Amb **Update:** update replace *ubicació* with *element*
Amb **XQUF:** replace node *ubicació* with *element*

A tots dos llenguatges, *ubicació* ha de fer referència a un únic element;

Reemplacen l'element indicat per la ***ubicació*** per l'***element*** amb el qual s'acaba la instrucció.

Exemple: canviar el nom de l'estudiant amb id = 393 per Dani (es diu Daniel).

Amb **Update:**

```
update replace /classe/estudiant[@id="393"]/nom with <nom>Dani</nom>
```

Amb **XQUF:**

```
replace node /classe/estudiant[@id="393"]/nom with <nom>Dani</nom>
```

A tots dos, l'identificador pot posar-se amb o sense cometes.

Modificar un el valor d'un element

Amb **Update:** update value *ubicació* with *element*
Amb **XQUF:** replace value of node *ubicació* with *element*

A XQUF, ***ubicació*** ha de fer referència a un únic element;

Reemplacen l'element indicat per la *ubicació* per l'element amb què s'acaba la instrucció.

Exemple: canviar el nom de l'estudiant amb id = 393 per Dani (es diu Daniel).

Amb **Update:** `update value /classe/estudiant[@id="393"]/nom with "Dani"`

Amb **XQUF:** `replace value of node /classe/estudiant[@id="393"]/nom with "Dani"`

A tots dos, l'identificador pot posar-se amb o sense cometes.
A tots dos, el nou valor s'ha de posar entre cometes.

Tant Update com XQUF poden combinar-se amb la part FLWOR d'XQuery. La manera de fer-ho és escrivint la consulta FLWOR normalment, però substituint el que es retorna per una sentència Update o XQUF. En aquesta sentència, el paràmetre ***ubicació*** serà una expressió on pot intervenir la variable del ***for***. Aquesta sentència s'executarà una vegada per cada element trobat (és a dir, per cada ***return*** executat).

Un avantatge important de la utilització d'XQuery respecte a utilitzar directament XPath és que desapareix la limitació que hi havia en alguns casos que obligava a que la ***ubicació*** hagués de referenciar només un element. En fer-ho així, cada execució del ***return*** referencia un sol element, però, en executar tot el ***for***, s'executa l'actualització sobre un conjunt d'elements.

Alguns dels exemples anteriors podrien escriure's de la següent manera:

Esborrat d'un element

Exemple: suprimir l'afició "Lectura" a l'estudiant amb id = 393

Amb **Update:**
```
update delete /classe/estudiant[@id="393"]/aficions/aficio[.="Lectura"]
```

Quedaria així:
```
for $a in /classe/estudiant[@id="393"]/aficions/aficio[.="Lectura"] return update delete $a
```

Amb **XQUF:**
```
delete node /classe/estudiant[@id="393"]/aficions/aficio[.="Lectura"]
```

Quedaria així:
```
for $a in /classe/estudiant[@id="393"]/aficions/aficio[.="Lectura"] return delete node $a
```

Modificar un element

Exemple: canviar el nom de l'estudiant amb id = 393 per Dani (es diu Daniel).

Amb **Update:**
```
update replace /classe/estudiant[@id="393"]/nom with <nom>Dani</nom>
```

Quedaria així:
```
for $n in /classe/estudiant[@id="393"]/nom
                                    return update replace $n with <nom>Dani</nom>
```

Amb **XQUF:** `replace node /classe/estudiant[@id="393"]/nom with <nom>Dani</nom>`

Quedaria així:

```
for $n in /classe/estudiant[@id="393"]/nom return replace node $n with <nom>Dani</nom>
```

Un exemple diferent dels vistos abans d'utilitzar FLWOR podria ser el següent. S'ha inclòs perquè en ell l'ús de FLWOR evita alguna de les restriccions existents sense FLWOR.

Modificació massiva

Exemple: Substituir l'afició "Esport" per "Basquet" a tots els alumnes.

Podria fer-se reemplaçant el valor, però ho farem substituint tot el node perquè era l'operació que tenia la restricció que deia que el paràmetre ***ubicació*** només podia fer referència a un node.

Amb **Update**:

```
for $a in /classe/estudiant/aficions/aficio[.="Esport"]  return update replace $a  with
                                                                <aficio>Basquet</aficio>
```

Amb **XQUF**:

```
for $a in /classe/estudiant/aficions/aficio[.="Esport"]  return replace node $a  with
                                                                <aficio>Basquet</aficio>
```

Introducció a l'API XQJ (XQuery API for Java) i a l'API específica de BaseX

Recordatoris:

- L'API estàndard XML:DB ja és obsoleta.
- Les bases de dades estudiades suporten l'API XQJ gràcies a les biblioteques de Charles Foster.

API XQJ

Element	Codi Java
Declaració de la connexió amb la base de dades.	`XQConnection conn;`

Obrir la connexió:
• *nomServidor* és una cadena de caràcters amb l'adreça IP o el nom DNS del servidor (també pot ser "localhost").
• *port* és un enter amb el número del port on escolta el servidor; la instal·lació per defecte treballa amb el port 1984.
• *usuari* i *contrasenya* són dos String que contenen, respectivament, l'usuari i la contrasenya de la base de dades; en la instal·lació per defecte de BaseX existeix l'usuari "admin" amb contrasenya "admin".
	`try {`
	`  XQDataSource xqs = new BaseXXQDataSource();`
	`  xqs.setProperty("serverName",nomServidor);`
	`  xqs.setProperty("port",port);`
	`  conn = xqs.getConnection(usuari, contrasenya);`
	`} catch (XQException ex) {`
	`  // tractament de l'excepció`
	`}`

Element	Codi Java

Tancar la connexió.	`try {`
	`  conn.close();`
	`} catch (XQException ex) {`
	`  // tractament de l'excepció`
	`}`

Executar una instrucció d'actualització.
 	
L'expressió *actualitzacio* és una cadena de caràcters que conté la sentència d'actualització en Update o XQUF.
	`try {`
	`  XQExpression query = conn.createExpression();`
	`  query.executeQuery(actualitzacio);`
	`} catch (XQException ex) {`
	`  // tractament de l'excepció`
	`}`

Execució d'una consulta i recorregut del resultat.
• *consulta* és una cadena de caràcters que conté la consulta a realitzar en XQuery o XPath.
• *getItemAsString(null)* retorna l'element com un string en format XML
• si només esperem un element, pot substituir-se el while per un if.
	`try {`
	`  XQExpression query=conn.createExpression();`
	`  resultatQuery=query.executeQuery(consulta);`
	
	`  while(resultatQuery.next()){`
	`  tractament(resultatQuery.getItemAsString(null));`
	`  // ↑ aquí tractem cada element`
	`  }`
	`} catch (XQException ex) {`
	`  // tractament de l'excepció`
	`}`

API específica de BaseX

Element	Codi Java
Declaració de la sessió sobre la base de dades.	`ClientSession session;`

Obrir la sessió:
• nomServidor és una cadena de caràcters amb l'adreça IP o el nom DNS del servidor (també pot ser "localhost").
• port és un enter amb el número del port on escolta el servidor; la instal·lació per defecte treballa amb el port 1984.
• usuari i contrasenya són dos String que contenen, respectivament, l'usuari i la contrasenya de la base de dades; en la instal·lació per defecte existeix l'usuari "admin" amb contrasenya "admin".
	`try {`
	`  session = new ClientSession(nomServidor, port,`
	`  usuari, contrasenya);`
	`} catch (IOException ex) {`
	`  // tractament de l'excepció`
	`}`

Tancar la sessió.	`try {`
	`  session.close();`
	`} catch (IOException ex) {`
	`  // tractament de l'excepció`
	`}`

Element	Codi Java

Executar una instrucció d'actualització.
 	
L'expressió *actualitzacio* és una cadena de caràcters que conté la sentència d'actualització en XQUF.
	`try {`
	`  ClientQuery query=session.query(actualitzacio);`
	`  query.execute();`
	`} catch (IOException ex1) {`
	`  // tractament de l'excepció`
	`}`

Execució d'una consulta i recorregut del resultat.
• *consulta* és una cadena de caràcters que conté la consulta a realitzar en XQuery o XPath.
• *element* obté un element com un string en format XML.
	`try {`
	`  query=session.query(consulta);`
	`  for(String element=query.next();element!=null;`
	`  element=query.next()){`
	`  tractament(element)); //tractem cada element`
	`  }`
	`} catch (IOException ex) {`
	`  // tractament de l'excepció`
	`}`

Execució d'una consulta de resultat únic i tractament d'aquest resultat.
• *consulta* és una cadena de caràcters que conté la consulta a realitzar en XQuery o XPath.
• *element* obté el resultat com un string en format XML.
	`try {`
	`  query=session.query(consulta);`
	`  String element=query.next();`
	`  if(element==null){`
	`  // tractament no hi ha cap resultat`
	`  }else{`
	`  tractament(element); //tractem l'element`
	`  }`
	`} catch (IOException ex) {`
	`  throw new GestorException(ex.getMessage());`
	`}`