Tema 6: XQuery

6.1. Introducció

6.1.1. Base de dades

Una base de dades (BBDD) és una eina que permet organitzar les dades que tenen algun tipus de relació entre sí, típicament emmagatzemada a un ordinador. Una base de dades és controla normalment mitjantçant un sistema gestor de base de dades (SGDB) o DBMS en anglés.

6.1.2. Tipus de BBDD

Hi ha molts tipus de BBDD segons el model lògic on es descriuen les dades.

BBDD jeràrquica

Utilitza una estructura en forma d'arbre (arrel, nodes pares/fills, nodes fulla). Són eficients per a la cerca de dades, però la seva estructura és més dificil de modificar. Permet la redundància de les dades

BBDD relacional

En aquest model no existeix jerarquia, les dades s'emmagatzemen en tuples (una fila) que formen taules (conjunt de tuples). Cada taula representa un objecte del mon real que es vol representar en un ordinador (contenen dades amb la mateixa informació semàntica). Les taules es poden relacionar entre sí amb taules intermèdies (claus primàries i claus foràanees), solucionant el problema de la redundància. Fan servir el llenguatge SQL per a fer consultes.

Alguns exemples de BBDD relacionals a nivell comercial són Oracle (primer SBDG comercial) o Microsoft SQL Server. En l'àmbit Open Source tenim postgreSQL o mariaDB/mySQL.

BBDD transaccional

Poc comuns. Es fan servir a les entitats bancàries per a la transferència de diners. Permeten enviar informació a gran velocitat.

BBDD noSQL

Són bases de dades no relacionals. Algunes característiques:

- No fan servir el lleguatge SQL, les dades s'emmagatzemen en parells clau-valor. Els registres s'emmagatzemen com a documents (típicament JSON). Un exemple seria el sistema mongoDB.

6.1.2. SGDB

L'ús dels SGBD és molt comú i té alguns avantatges respecte a l'ús de fitxers:

- Evita la redundància de dades.
- Evita errors de robustesa (dos programes volen modificar la mateixa dada).
- Permet la independència del Hardware i del software que gestiona la BBDD.

També té alguns inconvenients:

- No són fàcils d'administrar, es requereixen coneixements tècnics, personal especialitzat en administració de BBDD i en llenguatges de consultes.
- Si no necessitem accés concurrent a les dades (més d'un usuari vol modificar la mateixa dada), podem fer servir fitxers.
- Cost elevat en HW i SW, requereix d'un servidor.

XML al B2B

La majoria de BBDD actuals estan basades en el model de dades E-R i totes les empreses disponen d'alguna BBDD on emmagatzemen informació del seu negoci.

Anomenem B2B (Bussiness to Bussiness) a les transaccions comercials entre empreses (proveïdors, distribuidors), que implica la compartició d'informació entre elles. Aquesta compartició de dades suposa un probleme doncs cada empresa utilitza SGBD diferents. XML permet definir una representació d'informació que les empreses desitjen compartir. El SGDB haurà d'incorporar la possibilitat d'exportar o importar dades amb XML.

Per exemple, les empreses amb ERPs com SAP fan servir el document IDoc (Intermediate Document) que permet tranferir informació entre bases de dades. Un IDoc pot representar una factura, una comanda de vendes, una entrada al magatzem, una entrega, etc.

6.2. Bases de dades XML

Un document XML és pot considerar una base de dades (BBDD), doncs la informació està guardada de manera estructurada, és a dir organitzada o ordenada dintre del document. Podem dir que una base de dades XML (document) permet emmagatzemar dades en format XML (format estructurat).

Les operacions que normalment es poden fer amb una BBDD com inserir, modificar o esborrar -operacions CRUD- es realitzen amb els llenguatges XPath i XQuery.

Les bases de dades XML són una bona solució si la informació està estructurada de forma jeràrquica i necessitem un accés lleuger a la BBDD. Per actualitzar una columna a XML hem d'actualitzar tot el document XML. Per tant, si les dades són actualitzades molt freqüentment i els documents XML tenen moltes files (o registres) pot ser més eficients guardar les dades en una BBDD relacional.

Per altra banda, si actualitzem documents petits i pocs documents al mateix temps, XML també pot ser eficient.

6.2.1. Sistemes d'emmagatzematge de la informació mitjançant XML

Existeixen tres formes d'emmagatzemament de la informació mitjançant XML:

- Fitxers de text
- Bases de dades no natives XML (XML Enabled)
- Bases de dades natives XML

Fitxers

Es fan servir fitxers XML, els que fem servir al nostre SO. No és la millor opció, doncs no podem garantir la concurrència (dues persones no poden modificar el mateix arxiu a la vegada), la seguretat és menor i altres avantatges que ofereixen els SGBD.

Bases de dades no natives XML (XML-Enabled)

Les dades es guarden en una base de dades relacional. La BBDD relacional converteix el document XML en un esquema relacional per importar després aquestes dades. Aquestes BBDD poden generar un document XML com a sortida.

Aquestes BBDD presenten alguns inconvenients:

- No es poden restaurar el documents XML originals (la BBDD és relacional i necessiten conversió).
- L'estructura de les dades deixa de ser jeràrquica per convertir-se en relacional.

Aquestes BBDD són més adients per sistemes on la majoria de les dades són no-XML. Els següents SGDB suporten el tipus de dades ISO XML:

- IBM DB2
- Microsoft SQL Server
- Oracle Database
- PostgreSQL

Bases de dades natives XML

Aquestes BBDD estan especialment adaptades per treballar amb dades XML. El seu model intern es basa en XML i les seves unitats d'emmagatzematge són els nodes XML i el document, mentre que en les BBDD relacionals són els camps i els registres.

En aquestes BBDD no s'utilitza el llenguatge de consulta SQL, sinó que es fan servir els llenguatges XPath i XQuery. Alguns exemples de BBDD natives XML són: MarkLogic, BaseX, eXist-db o sedna.

XPath serveix per seleccionar els nodes del document XML i XQuery permet fer les consultes al document XML.

Comparativa de BBDD XML natives:

Nom	Llicència	Llenguatge nadiu	XQuery 3.1/3.0/1.0	XQuery Update	XSLT 2.0	XForms
BaseX	BSD	Java	Si/Si/Si	Si	Si	Si
eXist	GNU LGPL	Java	Parcial/Parcial/Si	Parcial/Si	Si	Si
MarkLogic	Comercial	C++	No/Parcial/Si	Propietari	Si	Si
Sedna	Apache 2.0	C++	No/?/Si	Si	No	?

6.3. SQL i XQuery

SQL (Structured Query Language) és un llenguatge declaratiu. A diferència dels llenguatges imperatius, li indiquem les dades que volem, no com obtenir-les físicament del disc. SQL serveix per accedir i manipular BBDD relacionals:

Concretament, les operacions CRUD són:

- Creació i eliminació de taules
- Inserció, modificació i eliminació de registres.
- Execució de cerques mitjançant consultes (SELECT).

Les sentències SQL es classifiquen en diferents tipus segons el seu propòsit:

- **DDL (Data Definition Language)**: Permeten crear les estructures que emmagatzemaran les dades, com CREATE, ALTER o DROP.
- **DML (Data Manipulation Language)**: Permeten introduir dades i manipularles: SELECT, INSERT, UPDATE o DELETE.
- **DCL (Data Control Language)**: Permeten als administradors controlar l'accés als objectes de la BBDD: GRANT o REVOKE.

La versió de SQL més utilitzada és la del 1993. Respecte a l'us amb XML:

- Estàndar 2003, suport inicial a documents XML.
- Estàndar 2006, major integració amb XML (importació i exportació de dades XML).

XQuery (XML Query) és a XML el que SQL a les BBDD relacionals. XQuery és compatible amb les següents tecnologies: XML, Namespaces, XSLT, XPath i XML Schema.

XQuery 1.0 va ser desenvolupat pel grup de treball W3C i XPath 2.0 és un subconjunt de XQuery 1.0.

XQuery és molt semblant a SQL, afegint-li estructures dels llenguatges de programació com for, return, etc. Aquestes estructures s'anomenen sentències **FLOWR**, és a dir **For, Let, Where, Order by i Return**.

Quan s'analitza el document XML es crea un arbre de nodes on tenim:

- Node arrel o "/": El primer node del document XML.
- Node element: Qualsevol element del document XML, si te fills és un node pare, sino és un node fulla.
- Node text: Node que conté només texte
- Node atribut: Atributs dels elements XML que complementen la informació del node element.

Semblances i diferències entre XQuery i SQL:

- Els dos permeten fer consultes a documents XML i BBDD relacionals, respectivament.
- Comparteixen algunes característiques de les sentències FLWOR.

Per altra banda:

- XQuery recorre dades jeràrquiques i SQL dades relacionals.
- SQL permet eliminar i modificar dades, XQuery té dificultat a l'hora d'esborrar o editar dades.

6.4. XQuery

Algunes consideracions inicials:

- És sensible a majúscules i minúscules com XML.
- Els elements, atributs i variables han de tenir noms vàlids a XML.
- Podem realitzar les consultes carregant un arxiu .xq a al gestor de BBDD XML.
- Podem realitzar les consultes directament a l'interfície del SGDB-Jeràrquic com baseX, eXistDB, etc..
- Per processar els arbres dels nodes d'un document XQuery utilitza XPath.

Extracció de dades:

- Per extreure totes les dades d'un document:

Local:

```
doc("ruta_a_l'arxiu/nom_arxiu.xml")
```

URL:

```
doc("http://adreça_del_document")
```

Exemple: pel document "catalog.xml" emplaçat a la mateixa arrel:

```
doc("catalog.xml")
```

- Per accedir a un o un conjunt de nodes:

```
doc("ruta_a_l'arxiu/nom_arxiu.xml")Xpath
```

Exemple: per accedir a TOTS els cd's del document catalog.xml:

```
doc("catalog.xml")/catalog/cd
```

Comparació

Mitjançant operadors de comparació en els elements i els arguments d'un arbre. Utilitzem XPath.

```
doc("document.xml")//element_de_cerca[element_filtre="argument"]
```

Exemple:

```
doc("catalog.xml")//catalog/cd [price>=10]
```

Comparació de strings

```
doc("catalog.xml") //catalog/cd[./artist eq "Bob Dylan"]
```

Operadors XPath (w3schools)

6.5. Expressions FLWOR

FLWOR és un acronim de "For, Let, Where, Order by, Return". És una més potent que les vistes amb XPath per seleccionar nodes.

- **For** - selecciona una seqüència de nodes, guardant-los en una variable.
- **Let** - Uneix una seqüència a una variable, és opcional.
- **Where** - Filtra els nodes guardats a la variable del *for* o del *let*
- **Order by** - Ordena els nodes
- **Return** - Que retorna (s'evalua una vegada per cada node)

Utilitza variables amb el simbol dòlar ($).

Aquestes dues expressions XPath i FLWOR són equivalents:

XPath

```
doc("catalog.xml")//catalog/cd[price>=10]/artist
```

FLWOR

```
for $cd in doc("catalog.xml")//catalog/cd
  where $cd/price<=10
  return $cd/artist
```

Comparació SQL i FLWOR

```
SELECT ...columnes...
  FROM ...taules...                ===>
  WHERE ...condició_columnes...
```

```
SELECT ...nodes...
  FROM ...catalog.xml...           ===>
  WHERE ...condició_nodes...
```

```
for $var
  in (...catalog.xml...)...nodes...
  where ...condició_nodes (amb $var)...
```

FOR

En aquest cas es mostra els artistes que tenen un cd amb un preu superior a 10 i més petit a 15.

```
for $cd in doc("catalog.xml")//catalog/cd
  where $cd[price>10] and $cd[price<15]"
  return $cd/artist
```

LET

Podem assignar una nova variable, per mostrar els resultats.

```
for $cd in doc("catalog.xml")//catalog/cd
let $n:=$cd/artist
where $cd/price<10
return $n
```

RETURN Si volem tornar més d'un valor podem fer:

```
return $cd/(artist/name, Title)
```

o

```
let $Titulo:=$cd/Titile
let $nomArtis:=$cd/artist/name
let $codigo:=$cd/artist/cod_Artist
return concat($Titulo, " ", $nomArtis," ",$codigo)
```

ORDER

En aquest cas ordenem els artistes per nom.

```
order by $variable [ascending, descending]
```

```
for $a in doc("catalog.xml")//catalog/cd/artist
  order by $a descending
  return $a
```

De FLWOR a HTML

Podem realitzar consultes i guardar els resultats en format HTML. Ho podem fer tant amb expressions XQuery com clàusules FLWOR.

Indicarem la part del codi FLWOR entre claus "{" "}".

Si volem afegir, per exemple, codi CSS utilitzarem {{ i }}, en la declaració CSS, en compte d'una sola { i }.

```
<ul>
{
for $x in doc("catalog.xml")//catalog/cd/artist
  order by $x
  return <li>{$x}</li>
}
</ul>
```

Exemple complert:

XQuery

```
<html>
  <head>
    <title>Prova XQuery</title>
  </head>
  <body>
    <ul>
      {
      for $x in doc("catalog.xml")//catalog/cd/artist
        return <li>{$x}</li>
      }
    </ul>
  </body>
</html>
```

HTML

```
<html>
  <head>
    <title>Prova XQuery</title>
  </head>
  <body>
    <ul>
      <li><artist>Bob Dylan</artist></li>
      <li><artist>Bonnie Tyler</artist></li>
      <li><artist>Dolly Parton</artist></li>
    </ul>
  </body>
</html>
```

Per imprimir només el nom de l'element, enlloc d'escriure

```
return <li>{$x}</li>
```

posem

```
return <li>{data($x)}</li>
```

6.6. Actualització de dades amb XQuery

Encara que XQuery permet seleccionar dades, la versió XQuery 1.0 no té funcions per insertar, esborrar o actualitzar dades d'un XML. Patrick Lehti va desenvolupar un sistema per possibilitar aquestes funcions, encara que el consorci W3C, posteriorment, va proposar l'extensió XQUF (XQuery Update Facilites) que ja està implementada a partir de la versió 7.1 de BaseX. La versió que farem servir de BaseX és la 9.0.1 (2022), així que farem servir la implementació XQUF.

En els exemples que mostrarem a continuació **element** és un fragment XML i **ubicació** és una expressió XPath.

6.1 Insercio d'un element

La sintaxi general amb XQUF és:

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

S'ha d'especificar un dels elements que van dintre de les claus:

- into: l'element s'insereix dins dins l'element especificat, al final dels fills actuals (depen de l'implementació).
- as first into / as last into: l'element s'insereix dins de l'element especificat, al principi o al final respectivament dels fills actuals.
- before / after: l'element s'insereix abans o després de l'element especificat per la ubicació.

Exemple (veure XML exemple de l'apartat 5.9):

Afegir l'afició "gastronomia" a l'estudiant amb id = 393 al final de les seves aficions

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

L'atribut id pot anar o sense cometes

6.2 Esborrat d'un element

La sintaxi amb XQUF és:

delete node ubicacio

Exemple:

Suprimir l'afició lectura de l'estudiant amb id = 393

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

El punt fa referència al node actual, és a dir a una afició.

L'atribut id pot anar o no sense cometes

La versió amb llenguatge FLWOR seria:

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

6.3 Modificar un element

En el cas de modificació, podem modificar l'element sencer o el contingut de l'element especificat.

La sintaxi amb XQUF per modificar l'element:

replace node ubicació **with** element

Modificar un element:

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

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

Per altra banda, la sintaxi per modificar el valor de l'element:

replace value of node ubicació **with** element

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

La versió replace node amb FLWOR seria:

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

6.4 Restriccions per a la modificació massiva.

Als apartats anteriors, el paràmetre ubicacio no sempre pot fer referència a més d'un element. A continuació es mostren les restriccions per a cada sentencia.

sentencia	restricció
insert node	un unic element
delete node	un o varis elements
replace node	un unic element
replace value of node	un unic element

Amb FLWOR podem fer moficacions massives, cosa que amb XQUF sense FLWOR no podem. Per modificar massivament nodes (replace node) fariem el següent:

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

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

6.5 Enllaços d'interés

XQuery Update - BaseX Documentation