

## TEMA 5 BASES DE DADES XML

### 5.1 BASES DE DADES NATIVES XML

A diferència de les bases de dades relacionals (centrades en les dades). Les bases de dades natives *XML*, no posseeixen camps, ni emmagatzemem dades, el que emmagatzemen són documents *XML*, són bases de dades centrades en documents i la unitat mínima d'emmagatzematge és el document *XML*.

Es pot definir una base de dades nativa *XML* (o *XML* nativa), com un sistema de gestió d'informació que compleix amb els següents punts:

- Defineix un model lògic per a un document *XML* (i no per a les dades que contenen el document), i emmagatzema i recupera els documents segons aquest model.
- Tenen una relació transparent amb el mecanisme d'emmagatzematge, que ha d'incloure els característiques *ACID* de qualsevol SGBD (*Atomicity*, *Consistency*, *Isolation* and *Durability*: Atomicitat, Consistència, Aïllament i Durabilitat).
- Inclou un nombre arbitrari de nivells de dades i complexitat.
- Permet les tecnologies de consulta i transformació pròpies de *XML*, *Xquery*, *Xpath*, *XSLT*, etc., com a vehicle principal d'accés i tractament.

#### Avantatges de les bases de dades *XML*:

- Ofereix un accés i emmagatzematge d'informació ja en format *XML*, sense necessitat d'inserir codi addicional.
- La majoria incorpora un motor de cerca d'alt rendiment.
- És molt senzill afegir nous documents *XML* al repositori.
- Es pot emmagatzemar dades heterogènies.

#### Desavantatges de les bases de dades *XML*:

- Pot ser difícil indexar documents per a fer cerques.
- No sol oferir funcions per a l'agregació (crucial per al processament de transaccions en línia *OLTP – Online Transaction Processing*) en molts casos s'ha de reintroduir tot el document per a modificar només una línia.
- S'acostuma a emmagatzemar la informació en *XML* com a document o com un conjunt de nodes, per al que la seva síntesi per fer noves estructures sobre la marxa pot resultar complicat i lenta.

En l'actualitat la majoria dels SGBD incorpora mecanismes per a extreure i emmagatzemar dades en format *XML*. Tot seguit es mostra exemples de suport de dades *XML* en *Oracle* i en *MySQL*:

- El següent SELECT d'*Oracle* torna les files de la taula **EMPLEATS** en format **XML**:

```
SELECT XMLEMENT («EMP_ROW»,
    XMLFOREST (EMP_NO, COGNOMS, OFICI, DIR, DATA_ALTA,
    SALARI, COMISSIO, DEPT_NO) FILA_EN_XML
FROM EMPLEATS;
```

- **Creació d'una taula que emmagatzema dades del tipus *XML*** (el tipus de dades **XMLTYPE** permet emmagatzemar i consultar dades *XMLS*):

```
CREATE TABLE TAULA_XML_PROVA (COD NUMBER, DADES XMLTYPE);
```

- **Inserir files en format *XML*:**

```
INSERT INTO TAULA_XML_PROVA VALUES (1,
XMLTYPE ('<FILA_EMP><EMP_NO>123</EMP_NO><COGNOMS>Ferre
Garcia</COGNOMS><OFICI>Professor</OFICI><SALARI>1500</SALARI><
/FILA_EMP>'));
```

```
INSERT INTO TAULA_XML_PROVA VALUES (1,
XMLTYPE ('<FILA_EMP><EMP_NO>124</EMP_NO><COGNOMS>Plana
Ramirez</COGNOMS><OFICI>Dentista</OFICI><SALARI>3500</SALARI><
/FILA_EMP>'));
```

- **Extracció de dades de la taula**, es fa servir expressions *Xpath*:

Mostra els cognoms dels empleats:

```
SELECT EXTRACTVALUE (DADES, '/FILA_EMP/COGNOM') FROM
TAULA_XML_PROVA;
```

Mostra el nom de l'empleat amb número d'empleat = 123:

```
SELECT EXTRACTVALUE (DADES, '/FILA_EMP/COGNOM') FROM
TAULA_XML_PROVA WHERE EXISTSNODE (DADES,
'/FILA_EMP[EMP_NO=123]') = 1;
```

- Al següent exemple es mostra com **MySQL torna les dades d'una taula en format *XML***. Per exemple, si es disposa de la base de dades MySQL EXEMPLE i es vol obtenir les dades de la taula Departaments en format *XML*. L'usuari de la base de dades i la seva contrasenya és també exemple.

Des de la línia d'ordres i a la carpeta on es troba instal·lat MySQL s'escriu la ordre que es pot veure a **5.1 UNIX** per a obtenir les dades de la taula Departaments. On:

- ◆ **- -xml**, es fa servir per a produir una sortida *XML*.
- ◆ **-u**, tot seguit es posa el nom d'usuari de la base de dades EXEMPLE.
- ◆ **-p**, quan es pren la tecla [Intro] per a executar l'ordre demana la contrasenya d'usuari (*exemple*).
- ◆ **-e**, executa el comando i surt de MySQL.

```
X□ - user@user-desktop:~  
~$ mysql --xml -u exemple -p -e "SELECT * FROM exemple.departamentos;"  
Enter password:  
<?xml version="1.0"?>  
  
<resultset statement="SELECT * FROM exemple.departamentos"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
  <row>  
    <field name="dept_no">10</field>  
    <field name="dnombre">CONTABILIDAD</field>  
    <field name="loc">SEVILLA</field>  
  </row>  
  . . .  
~$ |
```

## 5.1 UNIX

El resultat sencer de **5.1 UNIX** és:

```
<?xml version="1.0"?>  
  
<resultset statement="SELECT * FROM exemple.departamentos"  
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
  <row>  
    <field name="dept_no">10</field>  
    <field name="dnombre">CONTABILIDAD</field>  
    <field name="loc">SEVILLA</field>  
  </row>  
  
  <row>  
    <field name="dept_no">20</field>  
    <field name="dnombre">INVESTIGACIÓN</field>  
    <field name="loc">MADRID</field>  
  </row>  
  
  <row>  
    <field name="dept_no">30</field>  
    <field name="dnombre">VENTAS</field>  
    <field name="loc">BARCELONA</field>  
  </row>  
  
  <row>  
    <field name="dept_no">40</field>  
    <field name="dnombre">PRODUCCIÓN</field>  
    <field name="loc">BILBAO</field>  
  </row>  
  
  <row>  
    <field name="dept_no">60</field>  
    <field name="dnombre">MARKETING</field>  
    <field name="loc">REUS</field>  
  </row>  
  
  <row>  
    <field name="dept_no">70</field>  
    <field name="dnombre">INFORMÀTICA</field>  
    <field name="loc">Reus</field>
```

```

</row>

<row>
  <field name="dept_no">77</field>
  <field name="dnombre">PRODUCCIÓ</field>
  <field name="loc">TORTOSA</field>
</row>
<row>
  <field name="dept_no">80</field>
  <field name="dnombre">PRODUCCIÓ</field>
  <field name="loc">TARRAGONA</field>
</row>
</resultset>

```

Si es vol redirigir la sortida a un fitxer **departaments.xml** es pot fer tal com es veu a **5.2 UNIX**.

```
X - user@user-desktop:~ 
~$ mysql --xml -u exemple -p -e "SELECT * FROM exemple.departamentos" > /home/user/departaments.xml
Enter password:
~$ |
```

5.2 UNIX

## 5.2 BASE DE DADES EXIST

**eXist** és un SGBD lliure de codi obert que emmagatzema dades XML d'acord amb un model de dades XML. El motor de base de dades està completament escrit en Java, suporta els estàndards de consulta *Xpath*, *Xquery* i *XSLT*, a més d'indexació de documents i suport per a actualitzar les dades i per a multitud de protocols com *SOAP*, *XML-RPC*, *WebDav* i *REST*. Amb el SGBD s'acompanya de diverses aplicacions que permeten executar consultes de manera directa sobre la base de dades.

Els documents XML s'emmagatzemen en col·leccions, les quals poden ser niades; des de un punt de vista pràctic el magatzem de dades funciona com a un sistema de fitxers. Cada document es troba a una col·lecció. No és necessari que els documents tinguin una *DTD* o un *XML Schema* associat, i dins d'una col·lecció pot emmagatzemar documents de qualsevol tipus.

A la carpeta **eXist\weapp\WEB-INF\data** és on es guarden els fitxers més importants de la base de dades:

- **dom.dbx**: magatzem central nadiu de dades; és un fitxer paginat on s'emmagatzema tots els nodes del document d'acord amb el model *DOM* del W3C.
- **collections.dbx**: s'emmagatzema la jerarquia de col·leccions i relaciona aquesta amb els documents que contenen; s'assigna un identificador únic a cada document de la col·lecció que és emmagatzemat també juntament amb l'índex.
- **elements.dbx**: és on es guarda l'índex d'element i atributs. El gestor de manera automàtica indexa tots els documents fent servir índexs numèrics per a identificar

els nodes del mateix (element, atributs, text i comentaris). Durant la indexació s'assigna un identificador únic a cada document de la col·lecció, que és emmagatzemat també junt amb l'index.

- **words.dbx**: per defecte eXist indexa tots els nodes de text i valors d'atributs dividint el text en paraules. A aquest fitxer s'emmagatzema aquesta informació. Cada entrada de l'index està formada per un parell *<id col·lecció, paraula>*; i després una llista al node que conté aquesta paraula.

### 5.2.1 Instal·lació d'eXist

eXist pot funcionar de diverses modes:

- Funcionant com a servidor autònom (ofert serveis de crida remota a procediments que funcionen sobre Internet com XML-RPC, WebDAV i REST).
- Inserint dins d'una aplicació Java.
- A un servidor J2EE, ofert serveis XML-RPC, SOAP i WebDAV.

A l'enllaç <http://exist-db.org/exist/apps/homepage/index.html#downloads> [link5.1] es pot descarregar l'última versió de la base de dades. En aquest cas la versió és **eXist-db-setup-2.2.jar**. S'instal·la fent servir el fitxer JAR que es descarrega de la web oficial. A la línia d'ordres cal invocar Java per fer la instal·lació tal com es pot veure a **5.3 UNIX** o fent doble clic sobre la icona del fitxer descarregat (se suposa a l'exemple que el fitxer JAR s'ha descarregat a /home/user/download).

```
X □ — user@user-desktop:~  
~$ cd /home/user/download  
~/home/user/download$ java -jar eXist-db-setup-2.2.jar  
~/home/user/download$ |
```

### 5.3 UNIX

**ULL!** Amb això arrenca el procés d'instal·lació, cal fixar-se durant aquest procés (Figura 5.1):

- En un dels passos de la instal·lació demanarà la **contrasenya** per a l'usuari **admin** (Figura 5.2).
- Cal indicar la **ruta on s'ha d'instal·lar eXist-db** (per defecte /usr/local/eXist-db veure Figura 5.3).
- Et pot demanar la ruta on es troba instal·lat el Java JDK.

A l'enllaç <http://exist-db.org/exist/apps/doc/quickstart.xml> [link5.2] és trobar més informació pel procés d'instal·lació.

Per arrencar el servidor d'eXist cal anar a la carpeta d'instal·lació (en aquest cas se suposa que és /home/user/eXist-db) i executar l'arxiu start.jar. Per fer això es pot fer des

de la línia d'ordres tal com es pot veure a **5.4 UNIX**.

Una vegada arrencat eXist no es pot aturar mentre es treballi amb el sistema de gestió de base de dades (SGBD). Si es fa, deixa de funcionar el sistema.



Figura 5.1: Instal·lació d'eXist.

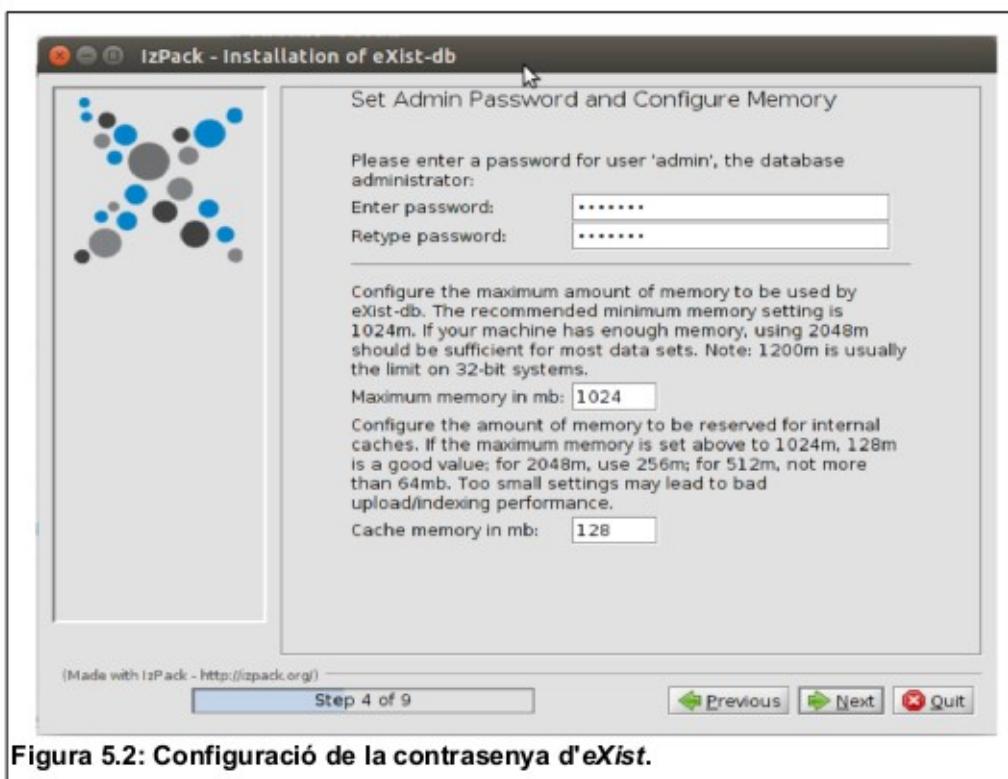


Figura 5.2: Configuració de la contrasenya d'eXist.

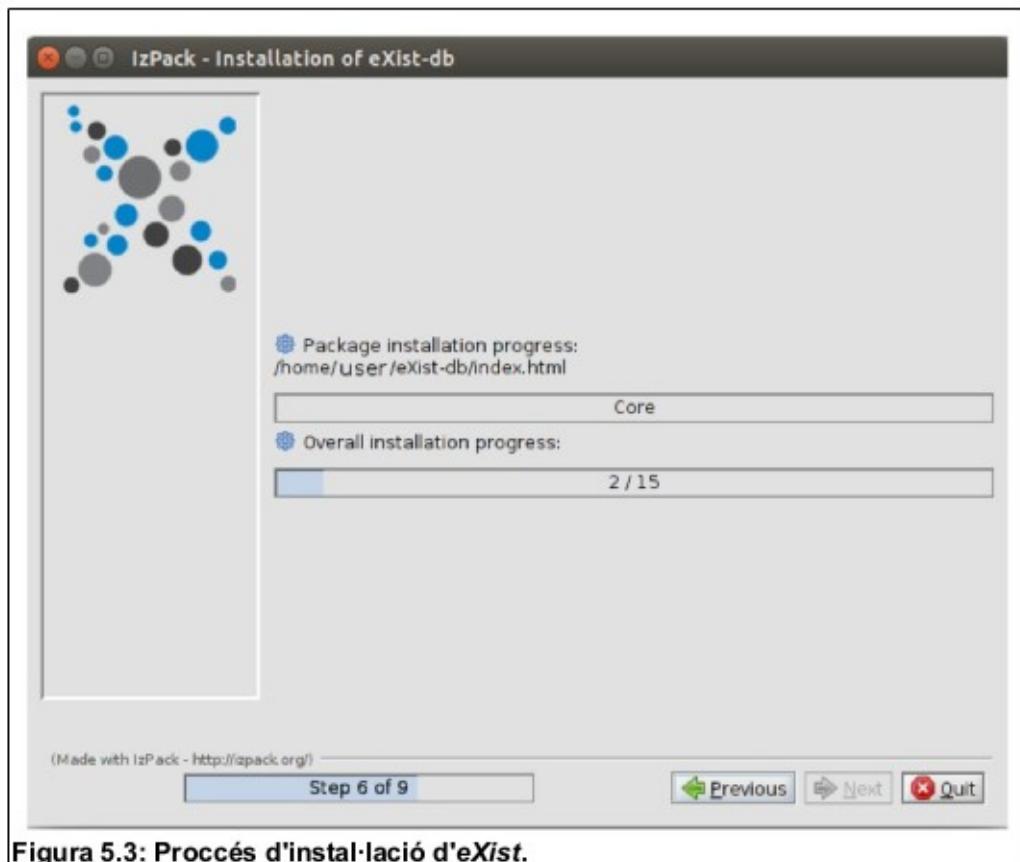


Figura 5.3: Proccés d'instal·lació d'eXist.

```
X□ - user@user-desktop:~  
~$ cd /home/user/eXist-db  
~/home/user/eXist-db$ java -jar start.jar  
Java opts: -Dfile.encoding=UTF-8 -Xms64m -Xmx1024m -Dexist.home="/home/  
user/eXist-db" -Djava.endorsed.dirs= . . .  
. . .  
. . .  
not part of the command.  
spawned process java.lang.UNIXProcess@8560f3  
~/home/user/eXist-db$ |
```

#### 5.4 UNIX

Es pot accedir al dashboard de diferents maneres; per exemple:

- Escrivint en un **navegador** web: **<http://localhost:8080/exist>**
- O donant-li clic dret a la **icona** de la barra d'eines i triar l'opció **Open dashboard**.

També és important tenir en compte que és possible que el port de connexió de la base de dades (per defecte el 8080) es trobi bloquejat per alguna altra aplicació (per exemple, pel firewall de Windows). S'ha d'assegurar que el port romanqui obert o habilitat.

Per **apagar** el servidor procedim de la mateixa forma i triem **Stop server**.

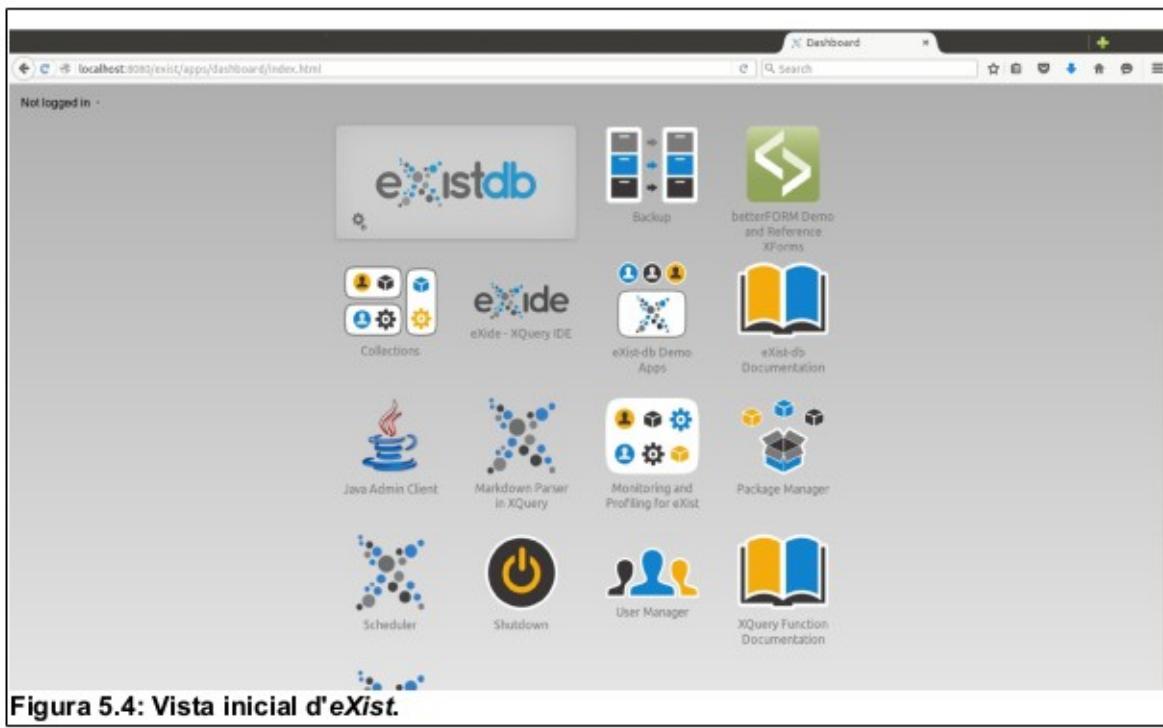


Figura 5.4: Vista inicial d'eXist.

### 5.2.2 Primers passos amb eXist

El primer que es vol fer és carregar un fitxer XML que servirà com a base de dades per a fer consultes fent servir XQuery i Xpath, així que serà necessari crear una col·lecció i després emmagatzemar el document XML a la col·lecció.

The screenshot shows the eXist login page at localhost:8080/exist/apps/dashboard/index.html. At the top left, it says 'Not logged in'. Below that is a form with the following fields: 'User:' followed by a text input field containing 'admin', 'Password:' followed by a text input field containing '\*\*\*\*\*', and a 'Remember me' checkbox which is unchecked. At the bottom of the form is a 'Login' button.

Figura 5.5: Panell d'administració d'eXist.

Al menú de la esquerra de la pantalla hi ha un enllaç que obre una finestreta per poder logar-se com a administrador (recorda que l'usuari és admin i la contrasenya configurada durant la instal·lació). Veure Figura 5.5.

Des de la finestra d'administració es pot veure tots els accessos a les diferents eines que proporciona l'aplicació.

Es vol crear una col·lecció de documents XML i pujar els documents *empleats.xml* i *departaments.xml* (aquests fitxers es troben als recursos del llibre base <https://sites.google.com/site/libroaccesoadatos/home/recursos-tema-5> [link5.3]) per a provar consultes. Es fa clic a **Collection Browser**, es crea la col·lecció **Proves**, fent clic a les icones d'eines de la barra superior **New collection**. Ja a la col·lecció creada es puja els documents *empleats.xml* i *departamentos.xml*, veure Figura 5.6. Cal recordar que els documents XML han d'estar ben formats per a no tenir cap problema al pujar aquests a el gestor de base de dades.

The screenshot shows the 'Collection Browser' window with the title 'Collection Browser'. Below the title is a toolbar with several icons. The main area displays a table titled '/db/Proves' with four columns: 'Name', 'Permissions', 'Owner', and 'Group'. There are three rows in the table:

Name	Permissions	Owner	Group
..	crwxr-xr-X	SYSTEM	dba
departamentos.xml	-rw-r--r--	admin	dba
empleados.xml	-rw-r--r--	admin	dba

**Figura 5.6: Col·lecció Proves.**

Amb la col·lecció ja creada cal obrir el gestor de consultes de eXist, des del panell d'administració (*dashboard*) es pot accedir a **eXide – Xquery IDE**. Des d'aquest gestor de consultes es pot fer consultes als documents XML de la col·lecció.

### 5.2.3 El eXide – Xquery IDE

eXide valida constantment codi mentre s'edita una XQuery. Combina un analitzador XQuery del costat del client (xqlint) amb els errors reportats pel servidor eXist-db.

La validació al servidor troba errors en tots els mòduls relacionats, mentre que l'anàlisi del costat del client ofereix l'arbre de sintaxi necessària per a la refactorització sofisticat, sensible al context codi de finalització, etc.

Característiques:

- Sintaxi-conscious: Validació mentre escriu
- Intellicode: Contextual finalització de codi
- Fragments de codi: Plantilles de codi per estalviar alguns tecleig
- Refactoring: Extreure funció o variable de bloc seleccionat
- Rebatejar: Canviar el nom d'una variable, funció o element XML sense trencar codi
- Solucions ràpides: per a alguns problemes freqüents
- Navegació pel codi: codi XQuery navegar ràpidament
- Mòdul d'importació: Importació de mòduls en una Xquery

- Seguretat
- Recarregar viva: Vistes prèvies instantànies
- Arxius deixar anar: Arrossegar i deixar anar arxius en l'editor
- Directori de càrregues: Conserva l'estructura de directoris

### 5.3 LLENGUATGE DE CONSULTES XPATH I XQUERY

Ambdós estàndards són per a accedir i obtenir dades des de documents XML, aquests llenguatges tenen en compte que la informació en els documents es troba semiestructurada o jerarquitzada com un arbre.

- **XPath**, és el llenguatge de rutes de XML, es fa servir per a navegar dins de l'estructura jeràrquica d'un fitxer XML.
- **XQuery** és a XML el mateix que SQL és a les bases de dades relacionals, és a dir, és un llenguatge de consulta dissenyat per a consultar documents XML. Abarca des de fitxers XML fins a bases de dades relacionals amb funcions de conversions de registres a XML. XQuery conté a XPath, tota expressió de consulta en XPath és vàlida en XQuery, tot i que aquesta permet molt més.

#### 5.3.1 Expressions XPath

XPath és un llenguatge que permet seleccionar nodes d'un document XML i calcular valors a partir del seu contingut. Hi ha les següents versions:

- **XPath 1.0** esdevingué una recomanació el 16 de novembre de 1999 .
- **XPath 2.0** esdevingué una recomanació el 23 de gener de 2007, està construït al voltant del *Model de Dades XQuery i XPath (XDM)* que té un sistema de tipus molt més ric. És de fet un subconjunt de XQuery 1.0.
- **XPath 3.0** esdevingué una recomanació el 8 d'abril de 2014, la novetat més important és el suport per a funcions com a valors first-class.
- **XPath 3.1** afegeix nous tipus de dades: mapes i matrius, en gran mesura per donar suport a JSON.

La manera en què XPath selecciona parts del document XML es basa en les representacions arbòries que es genera del document. A l'hora de recórrer un arbre XML, es pot trobar amb els següents tipus de node:

- **Node Arrel:** S'identifica per /. No s'ha de confondre el node arrel amb l'element arrel del document.
- **Node Element:** Qualsevol element d'un document XML es converteix en un node element dins de l'arbre.
- **Nodes text:** Tots els caràcters del document que no estan marcats amb alguna etiqueta (entre etiqueta).
- **Nodes atribut:** Són com a propietats afegides als nodes element, es representen amb @.

- **Nodes comentari:** Les etiquetes de comentaris.
- **Nodes espai de noms:** Conté espais de noms .
- **Nodes instruccions de procés:** Conté instruccions de procés, van entre les etiquetes <?.....?>.

Per exemple. Donat el següent document XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<universidad>
  <espacio xmlns="http://www.elmeulloc.cat"
            xmlns:prova="http://www.elmeulloc.cat/proves" />
  <!--DEPARTAMENTOS 1-->
    <departamento telefono="112233" tipo="A">
      <codigo>IFC1</codigo>
      <nombre>Informática</nombre>
      <empleado salario="2000">
        <puesto>Asociado</puesto>
        <nombre>Juan Parra</nombre>
      </empleado>
      <empleado salario="2300">
        <puesto>Profesor</puesto>
        <nombre>Alicia Martín</nombre>
      </empleado>
    </departamento>
  <!--DEPARTAMENTOS 2-->
    <departamento telefono="880833" tipo="B">
      <codigo>MAT2</codigo>
      <nombre>Análisis</nombre>
      <empleado salario="1900">
        <puesto>Asociado</puesto>
        <nombre>Laura Ruiz</nombre>
      </empleado>
      <empleado salario="2200">
        <puesto>Asociado</puesto>
        <nombre>Mario García</nombre>
      </empleado>
    </departamento>
  </universidad >
```

A la següent taula es classifiquen els diferents tipus de nodes:

Tipus node	
<b>Element</b>	<universidad> <nombre> <codigo> <nombre>
<b>Text</b>	MAT2 Informática Análisis IFC1
<b>Atribut</b>	telefono="880833" tipo="B" salario="2300"
<b>Comantari</b>	<!--DEPARTAMENTOS 1--> <!--DEPARTAMENTOS 2-->
<b>Espai de noms</b>	<espacio xmlns="http://www.elmeulloc.cat"             xmlns:prova="http://www.elmeulloc.cat/proves" />
<b>Instruccions de procés</b>	<?xml version="1.0" encoding="ISO-8859-1"?>
Tipus de node XPath.	
T a u l a 5 . 1	

Els test sobre els tipus de node poden ser:

- Nom del node, per a seleccionar un node concret, exe: `/universitat`
- **prefix:\***, per a seleccionar nodes amb un espai de nons determinat.
- **text()**, selecciona el contingut de l'element, és a dir el text, ex: `//nom/text()`.
- **node()**, selecciona tots els nodes, els elements i el text, ex: `/universitat/node()`.
- **processing-instruction()**, selecciona nodes que són instruccions de procés.
- **comment()**, selecciona els nodes de tipus comentari, `/universitat/comment()`.

La sintaxi bàsica de *XPath* és similar a la d'adreçament de fitxers. Fa servir descriptors de ruta o de camí que serveixen per a seleccionar els nodes o elements que es troben a certa ruta al document. Cada descriptor de ruta o pas de cerca pot a la vegada dividir-se en tres parts:

- **eix**: indica el node o els nodes en els que es fa la cerca.
- **node de comprobació**: especifica el node o els nodes seleccionats dins del eix.
- **predicat**: permet restringir les nodes de comprobació. Els predicats s'escriuen entre claudàtors.

Les expressions *XPath* es poden escriure fent servir una sintaxi abreujada, fàcil de llegir, o una sintaxi més completa en la que apareixen els noms dels eixos (*AXIS*), més completa. Per exemple, aquestes dues expressions tornen els departaments amb més de 3 empleats, la primera és la forma abreujada i la segona és la completa:

- `/universidad/departamento [count(empleado)>3]`
- `/child::universidad/child::departamento [count(child::empleado)>3]`

En aquest tema es comença amb la sintaxi abreujada, d'aquesta manera els descriptors es formen anomenat l'etiqueta separada per / (s'ha de posar el nom de l'etiqueta tal qual es troba en el document *XML*, cal recordar que es fa distinció entre majúscules i minúscules).

Si el descriptor comença amb / se suposa que és una **ruta des de l'arrel**. Per a continuar una ruta s'indica els diferents nodes de pas: `/pas1/pas2/pas3....` Si les rutes comencen amb / són rutes **absolutes**, en cas contrari seran **relatives**.

Si el descriptor comença amb // se suposa que la ruta descrita pot començar a **qualsevol part de la col·lecció**.

### **Exemples Xpath fent servir una sintaxi abreujada: CODI 5.1**

A partir de la col·lecció Proves, que té els documents: departaments.xml i empleats.xml, les estructures de les quals es mostren a la Figura 5.7. Es fa des de eXide – Xquery IDE les següents consultes:

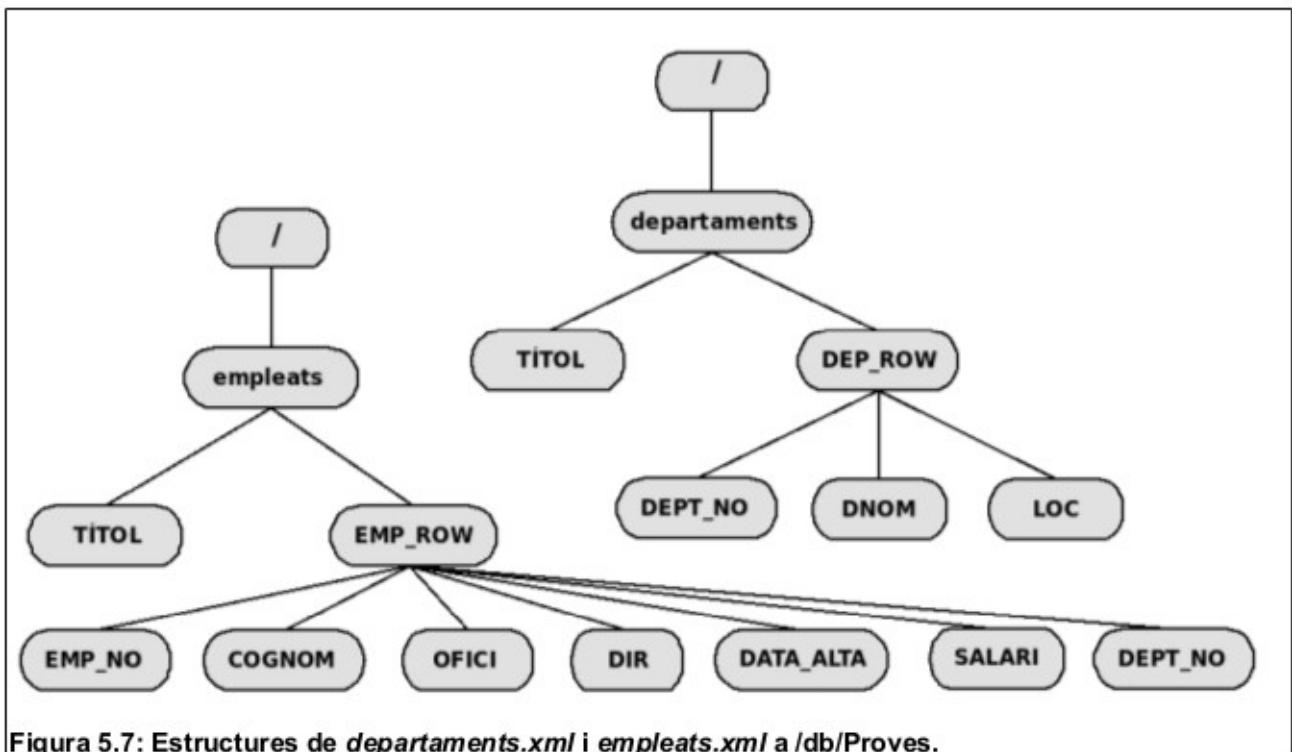


Figura 5.7: Estructures de *departaments.xml* i *empleats.xml* a /db/Proves.

/, si s'executa dins de /db/Proves torna les dades de departaments i empleats, inclou totes les etiquetes que pengen del node departaments i del node empleats, que es troben dins de la col·lecció Proves. Dins d'altre context el resultat serà diferent.

**/departaments**, torna totes les dades dels departaments, totes les etiquetes que pengen del node arrel departaments.

**/departaments/DEP\_ROW**, torna totes les etiquetes dins de cada *DEP\_ROW*.

**/departaments/DEP\_ROW/node()**, torna totes les etiquetes dins de cada *DEP\_ROW*, no inclòs *DEP\_ROW*.

**/departaments/DEP\_ROW/DNOM**, torna els noms de departaments de cada *DEP\_ROW*, entre etiquetes.

**/departaments/DEP\_ROW/DNOM/text()**, torna els noms de departaments de cada *DEP\_ROW*, ja sense etiquetes.

**//LOC/text()**, torna totes les localitats, de tota la col·lecció (//), només hi ha 4.

**//DEPT\_NO**, torna tots els nombres de departaments entre etiquetes. En aquest cas torna 18 fileres en lloc de 4, ja que dins de la col·lecció es troba també el document empleats.xml també amb etiquetes *DEPT\_NO*.

L'**operador \*** es fa servir per anomenar a qualsevol etiqueta, es fa servir com a comodí. Exemples:

- El descriptor **/\* / DEPT\_NO** selecciona l'etiqueta *DEPT\_NO* que es troba a nivell 1

de profunditat des de l'arrel, cap en aquest cas.

- `/*/*DEPT_NO` selecciona l'etiqueta `DEPT_NO` que es troba a dos nivells de profunditat des de l'arrel, en aquest cas 18.
- `/departaments/*` selecciona les etiquetes que van dintre de l'etiqueta departaments i les seves subetiquetes.

**Condicions de selecció.** Es fa servir els claudàtors per a seleccionar els elements concrets, a les condicions es pot fer servir els comparadors `<`, `>`, `<=`, `>=`, `=`, `!=`, `or`, `and` i `not` (`or`, `and` i `not` han d'escriure's en minúscula). Es fa servir el separador `|` per a unir diverses rutes. Exemples:

- `/EMPLEATS/EMP_ROW[ DEPT_NO=10 ]`, selecciona tots els elements o nodes (etiquetes) dins de `EMP_ROW` dels empleats del `DEPT_NO=10`.
- `/EMPLEATS/EMP_ROW/COGNOMS | /EMPLEATS/EMP_ROW/DEPT_NO`, selecciona els nodes `COGNOM` i `DEPT_NO` dels empleats.
- `/EMPLEATS/EMP_ROW[ DEPT_NO=10 ] /COGNOMS/text()`, selecciona els cognoms dels empleats del `DEPT_NO=10`.
- `/EMPLEATS/EMP_ROW[ not (DEPT_NO=10) ]`, selecciona tots els empleats (etiquetes) que NO són del `DEPT_NO` igual a 10.
- `/EMPLEATS/EMP_ROW[not (OFICI = 'ANALISTA') ] /COGNOMS/text()`, selecciona els `COGNOMS` dels empleats que NO són analistes.
- `/EMPLEATS/EMP_ROW[ DEPT_NO=10 ] /COGNOMS | /EMPLEATS/EMP_ROW[ DEPT_NO=10 ] /OFICI`, selecciona el `COGNOM` i l'`OFICI` dels empleats del `DEPT_NO=10`.
- `// *[ DEPT_NO=10 ]/DNOM/text()`, `/departaments/DEP_ROW[ DEPT_NO=10 ]/DNOM/text()`, aquestes dues consultes tornen el nom del departament 10.
- `// *[ OFICI='EMPLEAT']//EMP_ROW`, torna els empleats amb `OFICI` «`EMPLEAT`», per a cada empleat torna tots els seus elements. Cerca a qualsevol part de la col·lecció `//`. Això torna el mateix:  
`/EMPLEATS/EMP_ROW[ OFICI=>>EMPLEATS ] //EMP_ROW`.
- `/EMPLEAT/EMP_ROW[ SALARI>1300 and DEPT_NO=10 ]`, torna les dades dels empleats amb `SALARI` major de 1300 i del departament 10.
- `/EMPLEAT/EMP_ROW[ SALARI>1300 and DEPT_NO=20 ] /COGNOM | /EMPLEATS/EMP_ROW[ SALARI>1300 and DEPT_NO=20 ]/OFICI`, torna el `COGNOM` i l'`OFICI` dels empleats amb `SALARI` major de 1300 i del departament 20. Es fa servir el separador `|` per unir les dues rutes.

#### Utilització de funcions i expressions matemàtiques. CODI 5.2

- Un nombre dins de claudàtors representa la posició de l'element en el conjunt seleccionat. Exemple: `/EMPLEATS/EMP_ROW[1]`, torna totes les dades del primer empleat.  
`/EMPLEATS/EMP_ROW[5] /COGNOM/text()`, torna el `COGNOM` del cinquè empleat.

- La funció ***last()*** selecciona l'últim element del conjunt seleccionat. Exemple: **/EMPLEATS/EMP\_ROW[ last() ]**, selecciona totes les dades de l'últim empleat. **/EMPLEATS/EMP\_ROW[ last()-1 ]/COGNOM/text()**, toma el COGNOM del penúltim empleat.
- La funció ***position()*** torna un número igual a la posició de l'element actual. **/EMPLEATS/EMP\_ROW[ position()=3 ]**, obté els elements de l'empleat que ocupa la posició 3. **/EMPLEATS/EMP\_ROW[ position()<3 ] /COGNOM**, selecciona el cognom dels elements la posició dels quals és menor de 3, és a dir torna els cognoms del primer i el segon empleat.
- La funció ***count()*** compta el nombre d'elements seleccionats. Exemples: **/EMPLEATS/count(EMP\_ROW)**, torna el nombre d'empleats. **/EMPLEATS/count(EMP\_ROW [ DEP\_NO=10 ])**, compta el nombre d'empleats del departament 10. **/EMPLEATS/count(EMP\_ROW[ OFICI=>>>EMPLEAT & and SALARI>1300 ])**, compta el nombre d'empleats amb ofici EMPLEAT i SALARI major de 1300. **// \*[count(\*) = 3]**, torna elements que tenen 3 fills. **// \*[ count(DEP\_ROW) = 4 ]**, torna els elements que contenen 4 fills DEP\_ROW, tornarà l'etiqueta departaments i totes les subetiquetes.
- La funció ***sum()*** torna la suma de l'element seleccionat. Exemples: **sum(/EMPLEATS/EMP\_ROW/SALARI)**, torna la suma del SALARI. Si l'etiqueta a sumar la considera tipus *string* s'ha de convertir a un nombre fent servir la funció ***number***. **sum(/EMPLEATS/EMP\_ROW[ DEPT\_NO=20 ]/SALARI)**, torna la suma de SALARI dels empleats del DEPT\_NO=20.
- La funció ***max()*** torna el màxim, ***min()*** torna el mínim i ***avg()*** torna la mitjana de l'element seleccionat. Exemples:  
**max (/EMPLEATS/EMP\_ROW/SALARI)**, torna el salari màxim.  
**min (/EMPLEATS/EMP\_ROW/SALARI)**, torna el salari mínim.  
**min (/EMPLEATS/EMP\_ROW/[ OFICI=>>>ANALISTA ] /SALARI)**, torna el salari mínim dels empleats amb OFICI ANALISTA.  
**avg (/EMPLEATS/EMP\_ROW/SALARI)**, torna la mitjana del salari.  
**avg (/EMPLEATS/EMP\_ROW/[DEP\_NO=20] /SALARI)**, torna la mitjana del salari dels empleats del departament 20.
- La funció ***name()*** torna el nom de l'element seleccionat. Exemples:  
**// \*[name() ='COGNOM']**, torna tots els cognoms, entre les seves etiquetes.  
**Count(/ // \*[name() ='COGNOM'])**, compta les etiquetes amb nom COGNOM.
- La funció ***concat(cad1, cad2, ...)*** concatena les cadenes. Exemples:  
**/EMPLEATS/EMP\_ROW[DEPT\_NO=10]/concat(COGNOM, «-», OFICI)**, Torna el cognom i l'ofici concatenats dels empleats del departament 10.  
**/EMPLEATS/EMP\_ROW(concat(COGNOM, «-», OFICI, «-», SALARI))**, Torna la concatenació de cognom, ofici i salari dels empleats.
- La funció ***starts-with(cad1, cad2, ...)*** és veritable quan la cadena ***cad1*** té com a prefixe a la cadena ***cad2***. Exemples:  
**/EMPLEATS/EMP\_ROW[starts-with(COGNOM, 'A')]**, obté els elements dels empleats el COGNOM dels quals comença per 'A'.  
**/EMPLEATS/EMP\_ROW[starts-with(COGNOM, 'A')] /concat(COGNOM, « - «, OFICI)**,

obté COGNOM i NOMS concatenats dels empleats l'OFICI dels quals comença per 'A'.

- La funció ***contains(cad1, cad2, ...)*** és veritable quan la cadena cad1 conté a la cadena cad2. Exemples:  
**/EMPLEATS/EMP\_ROW[contains(OFICI, 'OR')]/OFICI**, torna els oficis que tenen la síl·laba 'OR'.  
**/EMPLEATS/EMP\_ROW[contains(COGNOM, 'A')]/COGNOM**, torna els cognoms que tenen una 'A'.
- La funció ***string-length(argument)*** torna el nombre de caràcters del seu argument.  
**/EMPLEATS/EMP\_ROW[concat(COGNOM, ' = ', string-length(COGNOM))]**, torna concatenat el cognom amb el seu nombre de caràcters.  
**/EMPLEATS/EMP\_ROW[string-length(COGNOM)<4]**, torna les dades dels empleats els COGNOM dels quals tenen menys de 4 caràcters.
- La funció ***div()*** fa una divisió en punt flotant.  
**/EMPLEATS/EMP\_ROW[concat(COGNOM, ' , ', SALARI, ' - ', div 12)]**, torna les dades concatenades de COGNOM, SALARI i salari dividit per 12.  
**sum(/EMPLEATS/EMP\_ROW/SALARI) div count(/EMPLEATS/EMP\_ROW)**, torna la suma de salaris dividits pel comptador d'empleats.
- La funció ***mod()*** calcula la resta de la divisió.  
**/EMPLEATS/EMP\_ROW[concat(COGNOM, ' , ', SALARI, ' - ', SALARI mod 12)]**, torna les dades concatenades de COGNOM, SALARI i la resta de dividir el SALARI per 12.  
**/EMPLEATS/EMP\_ROW[(SALARI mod 12) = 4]**, torna les dades dels empleats la resta dels quals de dividir el SALARI entre 12 és igual a 4 .

### Altres funcions.

- ***data (expressió Xpath)***, torna el text dels nodes de l'expressió sense etiquetes.
- ***number(argument)***, per a convertir a nombre l'argument, que pot ser cadena, booleà o un node.
- ***abs(num)***, torna el valor absolut del nombre.
- ***ceiling(num)***, torna l'enter més petit major o igual que l'expressió numèrica especificada.
- ***floor(num)***, torna l'enter més gran que sigui major o igual que l'expressió numèrica especificada.
- ***round(num)***, arrodoneix el valor de l'expressió numèrica.
- ***string(argument)***, converteix l'argument a cadena.
- ***compare(exp1, exp2)***, compara les dues expressions, torna **0** si són iguals, **1** si **exp1 > epx2** i **-1** si **exp2 > epx1**.
- ***substring(cadena, començament, num)***, extreu de la cadena, des de la posició indicada a començament el nombre de caràcters indicats a num.
- ***substring(cadena, començament)***, extreu de la cadena, des de la posició indicada a començament el nombre de caràcters fins al final.

- ***lower-case(cadena)***, converteix a minúscula la cadena.
- ***upper-case(cadena)***, converteix a majúscula la cadena.
- ***translate(cadena1, caract1, caract2)***, reemplaça dins de *cadena1*, els caràcters que s'expressen a *caract1*, per els corresponents que apareixen a *caract2*, un per un.
- ***ends-with(cadena1, cadena2)***, torna **true** si la *cadena1* acaba en *cadena2*.
- ***year-from-date(data)***, torna l'any de la data, el format de data és ANY-MES-DIA.
- ***month-from-date(data)***, torna el mes de la data.
- ***day-from-date(data)***, torna el dia de la data.

A l'enllaç <https://www.w3.org/2005/xpath-functions/#fo-summary> [link5.4] es pot trobar més funcions.

### EXERCICI 1

Puja el document *productos.xml* (*RecursosUnidad5*) dins de la col·lecció *Proves (eXist)*. Aquest document conté les dades dels productes d'una distribuïdora de components informàtics. Per cada producte hi ha el codi, la denominació, el preu, l'estoc actual, l'estoc mínim i el codi de zona. Les dades són:

```
<produc>
  <cod_prod>xxxx</cod_prod>
  <denominacion>xxxxxxxxxx</denominacion>
  <precio>xxxx</precio>
  <stock_actual>xxxx</stock_actual>
  <stock_minimo>xxxx</stock_minimo>
  <cod_zona>xxxx</cod_zona>
</produc>
```

Fes les següents consultes *XPath*:

- Obté els nodes denominacions i preus de tots els productes.
- Obté els nodes dels productes que siguin plaques base.
- Obté els nodes dels productes amb preu major de 60 € i de la zona 20.
- Obté el nombre de productes que siguin memòries i de la zona 10.
- Obté la mitjana del preu dels micros.
- Obté les dades dels productes l'estoc mínim dels quals sigui major que el seu estoc actual i segui de la zona 40.
- Obté el producte més car.
- Obté el producte més barat de la zona 20.
- Obté el producte més car de la zona 10.

### 5.3.2 Nodes atributs *XPath*

Un node pot tindre tants atributs com es vulgui i per a cada un per a cadascun se li crearà

un **node atribut**. Els nodes atribut NO es consideren com a fills, sinó més ben com a etiquetes afegides al node element. Cada node atribut consta d'un nom, un valor (el qual sempre és una cadena) i nu possible «espai de noms».

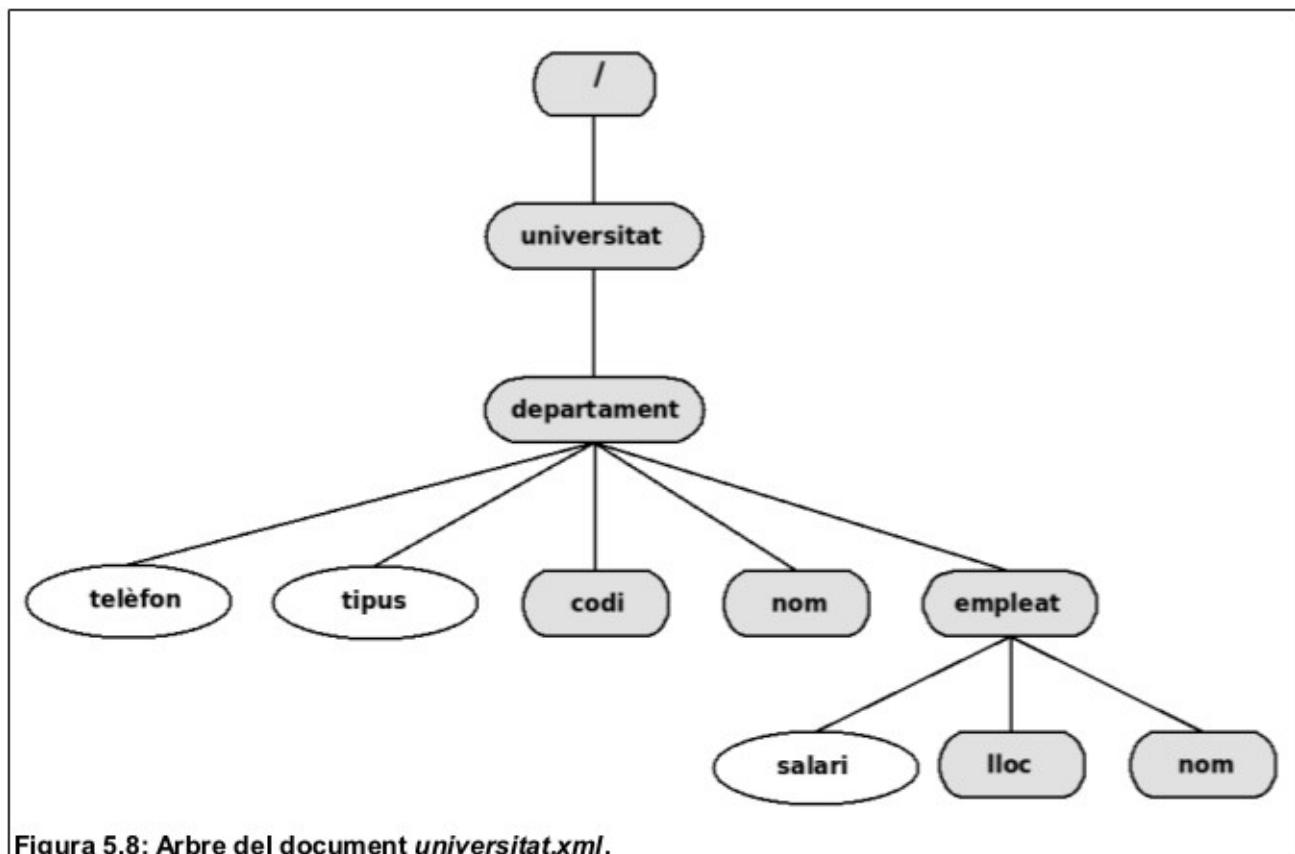
Partint del document *universidad.xml* (que es pot veure el codi XML tot seguit), que es troba a *RecursosUnidad5*. El document se suposa que ja es troba a la col·lecció *Proves* de la base de dades *eXist*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<universidad>
    <departamento telefono="112233" tipo="A">
        <codigo>IFC1</codigo>
        <nombre>Informática</nombre>
        <empleado salario="2000">
            <puesto>Asociado</puesto>
            <nombre>Juan Parra</nombre>
        </empleado>
        <empleado salario="2300">
            <puesto>Profesor</puesto>
            <nombre>Alicia Martín</nombre>
        </empleado>
    </departamento>

    <departamento telefono="990033" tipo="A">
        <codigo>MAT1</codigo>
        <nombre>Matemáticas</nombre>
        <empleado salario="1900">
            <puesto>Técnico</puesto>
            <nombre>Juan Parra</nombre>
        </empleado>
        <empleado salario="2100">
            <puesto>Profesor</puesto>
            <nombre>Mª Jesús Ramos</nombre>
        </empleado>
            <empleado salario="2300">
                <puesto>Profesor</puesto>
                <nombre>Pedro Paniagua</nombre>
            </empleado>
        <empleado salario="2500">
            <puesto>Tutor</puesto>
            <nombre>Antonia González</nombre>
        </empleado>
    </departamento>

    <departamento telefono="880833" tipo="B">
        <codigo>MAT2</codigo>
        <nombre>Análisis</nombre>
        <empleado salario="1900">
            <puesto>Asociado</puesto>
            <nombre>Laura Ruiz</nombre>
        </empleado>
        <empleado salario="2200">
            <puesto>Asociado</puesto>
            <nombre>Mario García</nombre>
        </empleado>
    </departamento>
</universidad >
```

En aquest document els elements o nodes que porten atributs són els següents: els elements departament porten els atributs telèfons i tipus, i els elements empleats porten l'atribut salari. L'arbre del document es pot veure a la *Figura 5.8*, els atributs es representen amb una el·ipse.



Per a referir-nos als atributs dels elements es fa servir @ abans del nom, per exemple @telefon, @tipus, @salari. A un descriptor de ruta els atributs s'anomenen com si fossin etiquetes fill però anteposant @.

#### Exemples: CODI 5.3

- **/universitat/departament[@tipus]**, s'obtenen les dades dels departaments que tinguin l'atribut tipus. Si es posa **data(/universitat/departament[@tipus])**, torna les dades sense etiquetes.
- **/universitat/departament/empleat[@salari]**, s'obtenen les dades dels empleats que tinguin l'atribut salari.
- **/universitat/departament[@telefon=>990033]**, s'obtenen les dades del departament el telèfon del qual és 990033. Si es posa **data(/universitat/departament[@telefon=>990033])**, torna el mateix però sense les etiquetes dels elements.
- **/universitat/departament[@telefon=>990033]/nom/text()**, s'obtenen el nom del departament el telèfon del qual és 990033.

- **//departament[@tipus=>B]**, s'obtenen les dades dels departaments el tipus dels quals és B.
- **/universitat/departament[@tipus=>A]/empleat**, s'obtenen les dades dels empleats dels departaments del tipus A.
- **/universitat/departament/empleat[@salari>2100]**, s'obtenen les dades dels empleats el salari dels quals és major de 2100.
- **/universitat/departament/empleat[@salari>2100]/nom/text()**, s'obtenen els noms dels empleats el salari dels quals és major de 2100.
- **/universitat/departament/empleat[@salari>2100]/concat(nom, ' ', @salari)**, s'obtenen les dades concatenades del nom d'empleat i el seu salari, dels empleats els quals el seu salari és major de 2100.
- **/universitat/departament[@tipus=>A]/count(empleat)**, torna el nombre d'empleats que hi ha als departaments tipus=A.
- **/universitat/departament[@tipus=>A]/concat(nom, ' ', count(empleat))**, torna per cada departament del tipus=A, la concatenació del seu nom i el nombre d'empleats.
- **/universitat/departament(concat(nom, ' ', count(empleat)))**, torna el nombre d'empleats per cada departament.
- **sum(/ /empelat/@salari)**, torna la suma total del salari de tots els empleats, fa el mateix que això: **sum(/universitat/departament/empleat/@salari)**.
- **/universitat/departament(concat(nom, ' Total= ', sum(empleat/@salari)))**, s'obtenen per a cada departament la concatenació del seu nom i el total salarial.
- **min(/ /empleat/@salari)**, torna el salari mínim de tots els empleats.
- **/universitat/departament(concat(nom, ' Mínim= ', min(empleat/@salari)))**  
**/universitat/departament(concat(nom, ' Màxim= ', max(empleat/@salari)))**, La primera obté per a cada departament la concatenació del seu nom i el mínim salari, i la segona el màxim salari.
- **/universitat/departament(concat(nom, ' Mitjana= ', avg(empleat/@salari)))**, obté per a cada departament la concatenació del seu nom i la mitjana salarial.
- **/universitat/departament/[count(empleat)>3]**, obté les dades dels departaments amb més de 3 empleats.  
**/universitat/departament/[count(empleat)>3]/nom/text()**, en aquest cas obté el nom dels departaments amb més de 3 empleats.
- **/universitat/departament/[@tipus=>A] and count(empleat)>2]/nom/text()**, torna el nom dels departaments de tipus A i amb més de 2 empleats.

## EXERCICI 2

Puja el document *sucursales.xml* (*RecursosUnidad5*) dins de la col·lecció *Proves* (*eXist*). Aquest document conté les dades de les sucursals d'un banc. Per cada sucursal hi ha el telèfon, el codi, el director de la sucursal, la població i els comptes de la sucursal. I per cada compte hi ha el tipus de compte ESTALVI o PESNSIÓ, el nom del compte, el número, el saldo - haver-hi i el saldo-deu. Les dades són:

```
<sucursales>
    <Sucursal telefono="xxxxxxx" codigo="xxxx">
        <director>xxxxxxxxxx</director>
        <poblacion>xxxxxxxxxx</poblacion>
            <cuenta tipo="xxxxxxxxxx">
                <nombre>xxxxxxxxxx</nombre>
                <numero>xxxxxx</numero>
                <saldohaber>xxxxxx</saldohaber>
                <saldodebe>xxxxxx</saldodebe>
            </cuenta>
            .....
        </Sucursal>
        .....
    </sucursales>
```

Fes les següents consultes *XPath*:

- Obté les dades dels comptes bancaris el tipus dels quals sigui ESTALVI.
- Obté per a cada sucursal la concatenació del seu codi i el nombre de comptes del tipus ESTALVI que té.
- Obté els comptes de tipus PESNSIÓ de la sucursal amb codi SUC3.
- Obté per cada sucursal la concatenació de les dades, codi sucursal, director i total saldo haver-hi.
- Obté tots els elements de les sucursals amb més de 3 comptes.
- Obté tots els elements de les sucursals amb més de 3 comptes del tipus ESTALVI.
- Obté els nodes del director i la població de les sucursals amb més de 3 comptes.
- Obté el nombre de sucursals la població de la qual sigui Madrid.
- Obté per a cada sucursal, el seu codi i la suma de les aportacions dels comptes del tipus PESNSIÓ.
- Obté els nodes número de compte, nom de compte i el saldo haver-hi dels comptes amb saldo haver-hi major de 1000.
- Obté per cada sucursal amb més de 3 comptes del tipus ESTALVI, el seu codi i la suma del saldo deu d'aquests comptes.

### 5.3.3 Axis *XPath*

Un *AXIS* o eix, especifica la direcció que es vol a avaluar, és a dir, si es mou cap a cal en la jerarquia o cap a baix, si inclourà el node actual o no, és a dir, defineix un conjunt de nodes relatiu al node actual. Els noms dels eixos es poden veure a la *Taula 5.2*.

AXIS	Resultat
<b>ancestor</b>	Selecciona els avantpassats (pare, avis, etc.) del node actual.
<b>ancestor-or-self</b>	Selecciona els avantpassats (pare, avis, etc.) del node actual i el node actual en si.
<b>attribute</b>	Selecciona els atributs del node actual.
<b>child</b>	Selecciona els fills del node actual.
<b>descendant</b>	Selecciona els descendents (fills, néts, etc.) del node actual.
<b>descendant-or-self</b>	Selecciona els descendents (fills, néts, etc.) del node actual i el node actual en si.
<b>following</b>	Selecciona tot el document després de l'etiqueta de tancament del node actual.
<b>following-sibling</b>	Selecciona tots els germans que segueixen al node actual.
<b>parent</b>	Selecciona el pare del node actual.
<b>self</b>	Selecciona el node actual.

Tipus d'eixos XPath.

T a u l a 5 . 2

La sintaxi per a fer servir eixos és la següent:

**nom\_d'eix::nom\_node[expressió]**

#### Exemples: CODI 5.4

- **/universitat/child::\***, el mateix que **/child::universitat/child::element()**. Torna tots els fills d'universitat, és a dir els nodes dels departaments.
- **/universitat/departament/descendant::\***, torna els descendents del node departament, això fa el mateix:  
**/child::universitat/child::departament/descendant::element()**.
- **/universitat/departament/descendant::empleat**, torna els nodes empleats descendents dels nodes departament.
- **/universitat/descendant::nom**, torna tots els elements nom descendents d'universitat, tant noms de departaments com a empleats. Si es posa els següent, torna el text del nom: **data(/universitat/descendant::nom)**.
- **/universitat/departament/following-sibling::\***, selecciona tots els germans de departament a partir del primer, seguint l'ordre del document.
- **//empleat/following-sibling::node()**, selecciona tots els germans dels elements empleats que es troben al context.
- **//empleat[nom=>Ana García]/following-sibling::\***, selecciona els nodes germans de Ana García.
- **//empleat[nom=>Ana García]/following-sibling::empleat/nom/text()**, selecciona

els noms dels empleats germans de Ana García.

- `//empleat[nom=>Ana García]/following-sibling::empleat[!loc=>Professor]/nom/text()`, selecciona els noms dels empleats germans de Ana García que són professors.
- `//empleat/parent::departament/nom`, selecciona els noms dels pares dels elements empleats.
- `//empleat[nom=>Ana García]/parent::departament/nom`, selecciona el nom del pare de l'empleada Ana García.
- `/descendant::departament[1]`, selecciona els descendents del departament que ocupa la posició 3 al document.
- `/child::universitat/child::departament[count(child::empleat) > 3]`, és el mateix que `/universitat/departament[count(empleat) > 3]`, obté els departaments amb més de 3 empleats.
- `/child::universitat/child::departament/child::nom`, obté les etiquetes amb els noms dels departaments. És el mateix que `/universitat/departament/nom`, i que `data(/universitat/departament/nom)`.
- `/child::universitat/child::departament/child::nom/child::text()`, obté els noms de departaments.
- `/child::universitat/child::departament[attribute::tipus=>B][count(child::empleat) >= 2]/child::nom/child::text()`, torna el nom dels departaments de tipus B i amb 2 o més empleats. És el mateix que `/universitat/departament[@tipus=>B] and count(empleat) >= 2]/nom/text()`.

#### 5.3.4 Consultes XQuery

Una consulta en XQuery és una expressió que llegeix dades d'un o més documents en XML i retorna com a resultat altra seqüència de dades en XML, a la Figura 5.9 es pot veure el procés bàsic d'una consulta XQUERY. XQuery conté XPath, tota expressió de consulta en XPath és vàlida i retorna el mateix resultat en XQuery. XQuery permet:

- Seleccionar informació basada en un criteri específic.
- Cercar informació en un document o conjunt de documents.
- Unir dades des de múltiples documents o col·leccions de documents.
- Organitzar, agrupar i resumir dades.
- Transformar i reestructurar dades XML en altre vocabulari o estructura.
- Fer càlculs aritmètics sobre números i dates.
- Manipular cadenes de caràcters a format text.

En XQuery les consultes segueixen la norma **FLWOR** (es llegeix com a flower), correspon a l'acrònim **For**, **Let**, **Where**, **Order** i **Return**. Permet a diferència de XPath manipular, transformar i organitzar els resultats de les consultes. La sintaxi general d'una estructura FLWOR és:

```

for <variable> in <expresió XPath>
let <variables vinculades>
where <condició XPath>
order by <expresió>
return <expresió de sortida>

```

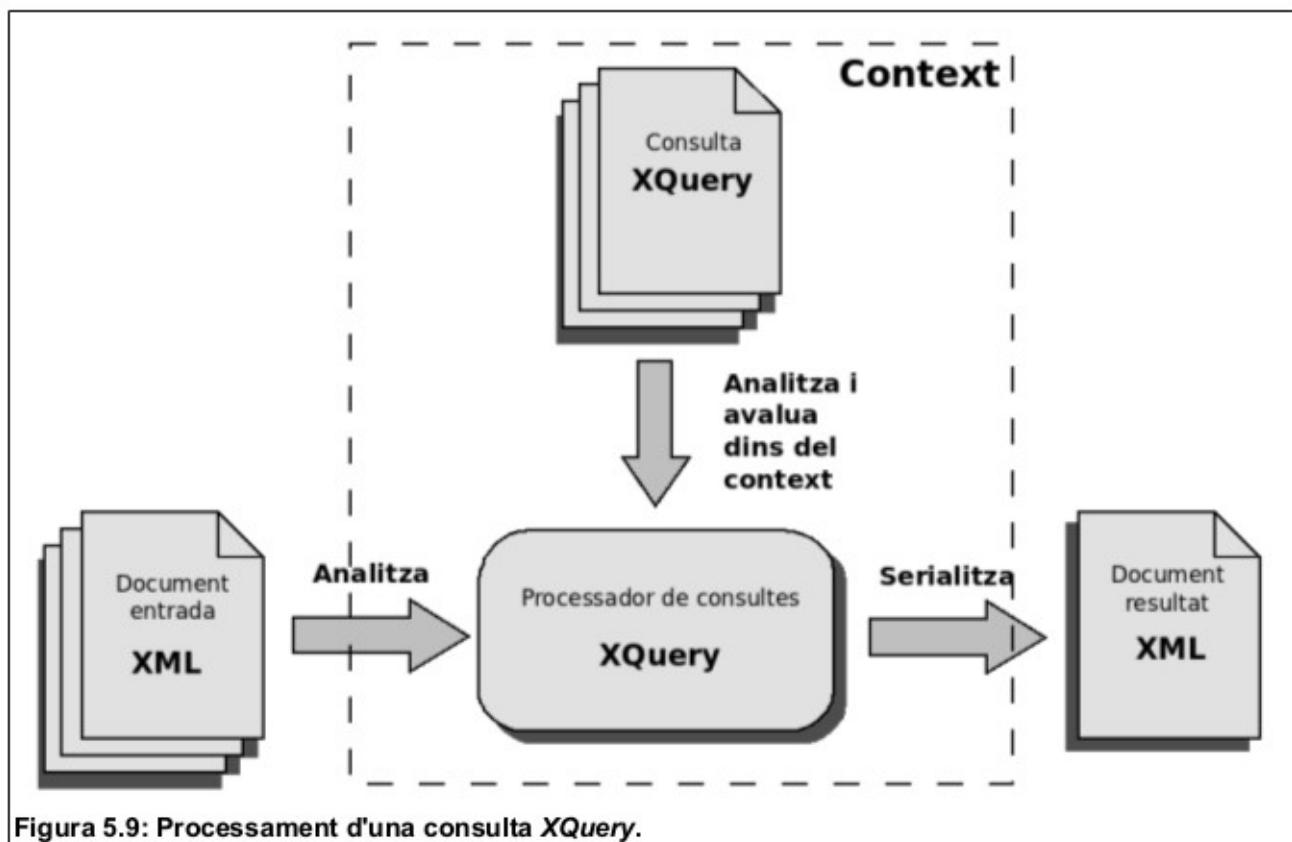


Figura 5.9: Processament d'una consulta XQuery.

### **for**

Es fa servir per a seleccionar nodes i emmagatzemar-los en una variable, similar a la clàusula `from` de SQL. Dins del `for` s'escriu una expressió XPath que selecciona els nodes. Si s'especifica més d'una variable al `for` s'actua com a producte cartesià. Les variables comencen amb `$`. Les consultes XQuery ha de portar de manera obligatòria l'ordre `return`, on s'indica el que es vol que torni la consulta. Per exemple a la Taula 5.3 la primera retorna els elements `EMP_ROW` i la segona els cognoms dels empleats. Una està escrita en XQuery i l'altra en XPath.

XQuery	XPath
<code>for \$emp in /EMPLEATS/EMP_ROW return \$emp</code>	<code>/EMPLEATS/EMP_ROW</code>
<code>for \$emp in /EMPLEATS/EMP_ROW return \$emp/COGNOM</code>	<code>/EMPLEATS/EMP_ROW/COGNOM</code>
Comparació de consultes en XQuery i XPath.	T a u l a 5 . 3

### **let**

Permet assignar valors resultats d'expressions *XPath* a variables per a simplificar la representació. Es pot ficar diverses línies **let** una per cada variable o separar les variables per comes.

Al següent exemple es creen dues variables, el *COGNOM* de l'empleat es guarda a **\$no**, i l'*OFICI* a **\$ofi**. La sortida surt ordenada per *OFICI* i es crea una etiqueta **<COG\_OFI></COG\_OFI>** que inclou el nom i l'ofici concatenat. Es fa servir les claus en el **return {}** per afegir el contingut de les variables. L'exemple és:

La clàusula **let** es pot fer servir sense **for** tal com es pot veure a la *Taula 5.4*:

sense For	amb For
<b>let \$ofi := /EMPLEATS/EMP_ROW/OFICI return &lt;OFICIS&gt;{\$ofi}&lt;/OFICIS&gt;</b>	<b>for \$ofi in /EMPLEATS/EMP_ROW/OFICI return &lt;OFICIS&gt;{\$ofi}&lt;/OFICIS&gt;</b>
La clàusula <b>let</b> vincula la variable <b>\$ofi</b> amb tot el resultat de l'expressió. En aquest cas vincula tots els oficis creant un element nou <b>&lt;OFICIS&gt;</b> amb tots ells.	La clàusula <b>for</b> vincula la variable <b>\$ofi</b> amb cada node ofici que trobi a la col·lecció de dades, creant un element per a cada ofici. Per això apareix l'etiqueta <b>&lt;OFICIS&gt;</b> per a cada ofici.
Comparació de consultes en <i>XQuery</i> amb la clàusula <i>let/for</i> .	T a u l a 5 . 4

### **where**

Filtra els elements, eliminant els valors que no compleixin amb les condicions donades.

### **order by**

Ordena les dades segons el criteri donat.

### **return**

Construeix el resultat de la consulta en *XML*, es pot afegir etiquetes *XML* a la sortida, si s'afegeix etiquetes, les dades per mostrar cal tancar-los entre claus **{}**. A més al **return** es pot afegir condicionals fent servir **if-then-else** i d'aquesta manera tenir més versatilitat a la sortida.

S'ha de tenir en compte que la clàusula **else** és obligatòria i ha d'aparèixer sempre a l'expressió condicional, això es deu al fet que tota expressió *XQuery* ha de tomar un valor. Si no hi ha cap valor a retornar com que no compleix la clàusula **if**, torna una seqüència buida amb **else()**.

El següent exemple torna els departaments de tipus A tancats dins una etiqueta:

```
for $dep in /universitat/departament
return if ($dep/@tipus='A')
    then <tipusA> {data($dep/nom)} </tipusA>
    else()
```

Es fa servir la funció **data()** per a extreure el contingut en text dels elements. També es fa servir **data()** per a extreure el contingut dels atributs, per exemple aquesta consulta *XPath* **//empleat/@salari** és errònia, ja que salari no és un node, però aquesta altra consulta **data("//empleat/@salari")** torna els salari.

Dins les assignacions *let* a les consultes XQuery es pot fer servir expressions del tipus *let \$var := //empleat/@salari* això dóna error, però si es vol extreure les dades, es pot escriure *let \$var := data(//empleat/@salari)* o *return data(\$var)*.

## CODI 5.5

entrada	sortida
<pre>for \$emp in /EMPLEATS/EMP_ROW order by \$emp/COGNOM return if (\$emp/OFICI='DIRECTOR') then &lt;DIRECT&gt;{\$emp/COGNOM/text() }&lt;/DIRECT&gt; else &lt;EMPLE&gt;{data (\$emp/COGNOM) }&lt;/EMPLE&gt;</pre>	<p>Torna els noms dels empleats, els que són directors entre etiquetes <code>&lt;DIRECT&gt;&lt;/DIRECT&gt;</code> i els que no ho són entre etiquetes <code>&lt;EMPLE&gt;&lt;/EMPLE&gt;</code>.</p> <p><code>&lt;EMPLE&gt;ALONSO&lt;/EMPLE&gt;</code>  <code>&lt;EMPLE&gt;ARROYO&lt;/EMPLE&gt;</code>  <code>&lt;DIRECT&gt;CEREZO&lt;/DIRECT&gt;</code>  <code>&lt;EMPLE&gt;GIL&lt;/EMPLE&gt;</code>  <code>&lt;DIRECT&gt;JIMENEZ&lt;/DIRECT&gt;</code>  <code>.....</code></p>
<pre>for \$de in doc('file:/home/user/departamentsNo us.xml')/departaments/DEP_ROW return \$de</pre>	<p>Torna els nodes <code>DEP_ROW</code> d'un document situat a una carpeta del disc dur.</p> <p><code>&lt;DEP_ROW&gt;</code>  <code>&lt;DEPT_NO&gt;10&lt;/DEPT_NO&gt;</code>  <code>&lt;DNOMBRE&gt;CONTABILIDAD&lt;/DNOMBRE&gt;</code>  <code>&lt;LOC&gt;SEVILLA&lt;/LOC&gt;</code>  <code>&lt;/DEP_ROW&gt;</code>  <code>&lt;DEP_ROW&gt;</code>  <code>&lt;DEPT_NO&gt;20&lt;/DEPT_NO&gt;</code>  <code>&lt;DNOMBRE&gt;INVESTIGACION&lt;/DNOMBRE&gt;</code>  <code>&lt;LOC&gt;MADRID&lt;/LOC&gt;</code>  <code>&lt;/DEP_ROW&gt;</code>  <code>.....</code></p>
<pre>for \$prof in /universitat/departament[@tipus='A'] /empleat let \$profe:= \$prof/nom, \$lloc:= \$prof/lloc where \$lloc='Professor' return \$profe</pre>	<p>Obté els noms d'empleats dels departaments de tipus A, el lloc dels quals és Professor. La següent consulta fa el mateix:</p> <pre>for \$prof in /universitat/departament[@tipus='A'] /empleat where \$prof/lloc='Professor' return \$prof/nom</pre> <p>El resultat és:</p> <p><code>&lt;nombra&gt;Alicia Martín&lt;/nombra&gt;</code>  <code>&lt;nombra&gt;Mª Jesús Ramos&lt;/nombra&gt;</code>  <code>&lt;nombra&gt;Pedro Paniagua&lt;/nombra&gt;</code></p>
<pre>for \$dep in /universitat/departament return if (\$dep/@tipus='A') then &lt;tipusA&gt;{data (\$dep/nom) }&lt;/tipusA&gt; else</pre>	<p>Torna el nom del departament tancat entre les etiquetes <code>&lt;tipusA&gt;&lt;/tipusA&gt;</code>, si és del tipus A, i <code>&lt;tipusB&gt;&lt;/tipusB&gt;</code>, si no ho és.</p> <p><code>&lt;tipusA&gt;Informática&lt;/tipusA&gt;</code>  <code>&lt;tipusA&gt;Matemáticas&lt;/tipusA&gt;</code>  <code>&lt;tipusB&gt;Análisis&lt;/tipusB&gt;</code></p>

<code>&lt;tipusB&gt;{data (\$dep/nom) }&lt;/tipusB&gt;</code>	
<pre>for \$dep in /universitat/departament let \$no:= \$dep/empleat return &lt;depart&gt;{data (\$dep/nom) } &lt;emple&gt;{count (\$no) }&lt;/emple&gt;&lt;/depart&gt;</pre>	Obté els noms de departament i els empleats que tenen entre etiquetes: <pre>&lt;depart&gt;Informática&lt;emple&gt;2&lt;/emple&gt;&lt;/depart&gt; &lt;depart&gt;Matemáticas&lt;emple&gt;4&lt;/emple&gt;&lt;/depart&gt; &lt;depart&gt;Análisis&lt;emple&gt;2&lt;/emple&gt;&lt;/depart&gt;</pre>
<pre>for \$dep in /universitat/departament let \$emp:= \$dep/empleat let \$sal:= \$dep/empleat/@salari return &lt;depart&gt;{data (\$dep/nom) }&lt;emple&gt; {count (\$emp) }&lt;/emple&gt;&lt;mitsal&gt; {avg(\$sal)}&lt;/mitsal&gt;&lt;/depart&gt;</pre>	Obté els noms de departament, els empleats que tenen i la mitjana del salari entre etiquetes: <pre>&lt;depart&gt;Informática&lt;emple&gt;2&lt;/emple&gt; &lt;mitsal&gt;2150&lt;/mitsal&gt; &lt;/depart&gt; &lt;depart&gt;Matemáticas&lt;emple&gt;4&lt;/emple&gt; &lt;mitsal&gt;2200&lt;/mitsal&gt; &lt;/depart&gt; &lt;depart&gt;Análisis&lt;emple&gt;2&lt;/emple&gt; &lt;mitsal&gt;2050&lt;/mitsal&gt; &lt;/depart&gt;</pre>

Diverses exemples de consultes en XQuery.

T a u l a 5 . 5

### 5.3.5 Operadors i funcions més comunes en XQuery

Les funcions i operadors suportats per XQuery són pràcticament els mateixos que els suportats per XPath. Suporta operadors i funcions matemàtiques, de cadenes, per al tractament d'expressions regulars, comparacions de dates i hores, manipulació de nodes XML, manipulació de seqüències, comprovació i conversió de tipus i lògica booleana. Els operadors i funcions més comunes són:

- Matemàtiques: +, -, \*, **div** (es fa servir *div* en lloc de la /), **idiv** (és la divisió sencera), **mod**.
- Comparació: =, !=, <, >, <=, >=, **not()**.
- Seqüència: **union()**, **intersect**, **except**.
- Arrodoniment: **floor()**, **ceiling()**, **round()**.
- Funcions d'agrupació: **count()**, **min()**, **max()**, **avg()**, **sum()**.
- Funcions de cadena: **concat()**, **string-length()**, **starts-with()**, **ends-with()**, **substring()**, **upper-case()**, **lower-case()**, **string()**.
- Ús general: **distinct-values()** extreu els valors d'una seqüència de nodes i crea una nova seqüència amb valors únics, eliminant els nodes duplicats. **empty()** torna cert quan l'expressió entre parèntesis està buida. I **exists()** torna cert quan una seqüència conté, el, menys, un element.
- Els comentaris en XQuery van tancats entre cares sonrients: (: Això és un comentari :).

**Exemples** (es fa servir el document *EMPLEATS.xml*): **CODI 5.6**

>> Noms d'ofici que comencen per *P*.

```

for $ofi in /EMPLEATS/EMP_ROW/OFICI
where starts-with (data($ofi), 'P')
return $ofi

sortida
<OFICIO>PRESIDENTE</OFICIO>

```

>> Obté els noms d'ofici i els empleats de cada ofici. Fa servir la funció *distinct-values* per a resoldre els distints oficis.

```

for $ofi in distinct-values(/EMPLEATS/EMP_ROW/OFICI)
let $comp:= count(/EMPLEATS/EMP_ROW[OFICI = $ofi])
return concat($ofi, ' = ', $comp)

sortida
EMPLEADO = 4
VENDEDOR = 4
DIRECTOR = 3
ANALISTA = 2
PRESIDENTE = 1

```

>> Obté el nombre d'empleats que té cada departament i la mitjana dels salaris arrodonida:

```

for $dep in distinct-values(/EMPLEATS/EMP_ROW/DEPT_NO)
let $comp := count(/EMPLEATS/EMP_ROW[DEPT_NO= $dep])
let $sala := round(avg(/EMPLEATS/EMP_ROW[DEPT_NO= $dep]/SALARI))
return concat('Departament: ', $dep, '. Núm. empleats = ',
$comp, '. Mitjana salari = ', $sala)

```

**sortida**

```

Departament: 20. Núm. empleats = 5. Mitjana salari = 2274
Departament: 30. Núm. empleats = 6. Mitjana salari = 1736
Departament: 10. Núm. empleats = 3. Mitjana salari = 2892

```

Si es vol retornar el resultat entre etiquetes, cal canviar la clàusula *return* com segueix:

```

return <depart><cod>{$dep}</cod><emples>{$comp}</emples>
      <mitjsala>{$sala}</mitjsala></depart>

```

**sortida**

```

<depart><cod>20</cod><emples>5</emples><mitjsala>2274</mitjsala></depart>
<depart><cod>30</cod><emples>6</emples><mitjsala>1736</mitjsala></depart>
<depart><cod>10</cod><emples>3</emples><mitjsala>2892</mitjsala></depart>

```

### EXERCICI 3

Puja el document *productos.xml* (*RecursosUnidad5*) dins de la col·lecció *Proves* (*eXist*). Cal fer les següents consultes XQuery:

- Obté per cada zona el nombre de productes que té.

- Obté la denominació dels productes entre etiquetes <zona10></zona10> si són del codi de zona 10, <zona20></zona20> si són de la zona 20, i el mateix per les zones 30 i 40.
- Obté per a cada zona la denominació del o dels productes més cars.
- Obté la denominació dels productes continguts entre les etiquetes <placa></placa> per a productes als quals a la seva denominació apareix la paraula Placa Base, <memoria></memoria>, per als que apareix la paraula Memòria, <micro></micro> per als que apareix Micro i <altres></altres> per a la resta de productes.

Fent servir el document *sucursals.xml*. Cal fer les següents consultes XQuery:

- Torna el codi de sucursal i el nombre de comptes que té de tipus ESTALVI i de tipus PENSIONS .
- Torna per cada sucursal el codi de sucursal, el director, la població, la suma del total deu i la suma del total haver dels seus comptes.
- Torna el nom dels directors, el codi de sucursal i la població de les sucursals amb més de 3 comptes.
- Torna per a cada sucursal, el codi de sucursal i les dades dels comptes amb més saldo deu.
- Torna el compte del tipus PENSIONS que ha fet més aportacions.

### 5.3.6 Consultes complexes amb XQuery

Dins de les consultes XQuery es pot treballar amb diversos documents XML per a extreure la seva informació, es pot incloure tantes sentències *for* com es vulgui, inclús dins de *return*. A més es pot afegir, esborrar i inclús modificar elements, ja sigui generant un document XML nou o fent servir les sentències d'actualització de eXist. Tot seguit es poden veure diversos exemples de diversa complexitat.

#### >> Joins de documents. CODI 5.7

→ Mostrar per a cada empleat del document *empleats.xml*, els seu cognom, el seu número de departament i el nom del departament que es troba al document *departaments.xml*:

```
for $emp in (/EMPLEATS/EMP_ROW)
let $emple := $emp/COGNOM
let $dep := $emp/DEPT_NO
let $dnom := (/departaments/DEP_ROW[DEPT_NO = $dep]/DNOM)
return <res>{$emple, $dep, $dnom}</res>

sortida
<res>
    <APELIDO>SANCHEZ</APELIDO>
    <DEPT_NO>20</DEPT_NO>
    <DNOMBRE>INVESTIGACION</DNOMBRE>
</res>
<res>
    <APELIDO>ARROYO</APELIDO>
    <DEPT_NO>30</DEPT_NO>
    <DNOMBRE>VENTAS</DNOMBRE>
</res>
```

→ Fent servir els documents *departaments.xml* i *empleats.xml*, aconseguir per cada departament, el nom de departament, el nombre d'empleats i la mitjana salarial:

```
for $dep in /departaments/DEP_ROW
let $d := $dep/DEPT_NO
let $tot := sum(/EMPLEATS/EMP_ROW[DEPT_NO=$d]/SALARI)
let $comp := count(/EMPLEATS/EMP_ROW[DEPT_NO=$d]/EMP_NO)
return <resul>{$dep/DNOM}<sumasalari>{$tot}</sumasalari>
      <totemple>{$comp}</totemple></resul>

sortida
<resul>
  <DNOMBRE>CONTABILIDAD</DNOMBRE>
  <sumasalari>8675</sumasalari>
  <totemple>3</totemple>
</resul>
<resul>
  <DNOMBRE>INVESTIGACION</DNOMBRE>
  <sumasalari>11370</sumasalari>
  <totemple>5</totemple>
</resul>
<resul>
  <DNOMBRE>VENTAS</DNOMBRE>
  <sumasalari>10415</sumasalari>
  <totemple>6</totemple>
</resul>
<resul>
  <DNOMBRE>PRODUCCION</DNOMBRE>
  <sumasalari>0</sumasalari>
  <totemple>0</totemple>
</resul>
```

→ Convertir la sortida de la consulta anterior, de manera que el total salarial, i el total d'empleats, seguin atributs de cada departament. La sortida serà una concatenació de les dades a mostrar:

```
for $dep in /departaments/DEP_ROW
let $d := $dep/DEPT_NO
let $tot := sum(/EMPLEATS/EMP_ROW[DEPT_NO=$d]/SALARI)
let $comp := count(/EMPLEATS/EMP_ROW[DEPT_NO=$d]/EMP_NO)
return concat('<departament sumasalari=>', $tot,
             '> totemple=>', $comp, '>>', data($dep/DNOM),
             '</departament>')

sortida
<departament sumasalari=>8675> totemple=>3>>CONTABILIDAD</departament>
<departament sumasalari=>11370> totemple=>5>>INVESTIGACION</departament>
<departament sumasalari=>10415> totemple=>6>>VENTAS</departament>
<departament sumasalari=>0> totemple=>0>>PRODUCCION</departament>
```

→ Fent servir els documents *departaments.xml* i *empleats.xml*, obtenir per cada departament, el nom d'empleat que més guanya:

```
for $emp in /EMPLEATS/EMP_ROW
let $d := $emp/DEPT_NO, $no := $emp/COGNOM, $sal := $emp/SALARI
let $ndep := (/departaments/DEP_ROW[DEPT_NO=$d]/DNOM)
let $salmax := max(/EMPLEATS/EMP_ROW[DEPT_NO=$d]/SALARI)
return if ($sal=$salmax) then
<depart>{data($ndep)}<salmax>{data($sal)}</salmax>
<emple>{data($no)}</emple></depart>
else ()

sortida
<depart>VENTAS<salmax>3005</salmax><emple>NEGRO</emple></depart>
<depart>INVESTIGACION<salmax>3000</salmax><emple>GIL</emple></depart>
<depart>CONTABILIDAD<salmax>4100</salmax><emple>REY</emple></depart>
<depart>INVESTIGACION<salmax>3000</salmax><emple>FERNANDEZ</emple></depart>
```

#### EXERCICI 4

Puja el document *zones.xml*, que conté informació de les zones on es venen els productes que apareixen al document *productos.xml*, (*RecursosUnidad5*) dins de la col·lecció *Proves* (*eXist*). Cal fer les següents consultes XQuery:

- Obté les dades de denominació, preu i nom de zona de cada producte, ordenats per nom de zona.
- Obté per cada zona, el nom de zona i el nombre de productes que té.
- Obté per cada zona, el nom de la zona, el seu codi i el nom del producte amb menor estoc actual.

>> Utilització de diversos *for*. La utilització de diversos *for* és molt útil per a consultes en documents XML niats i també quan es fa servir diversos documents units per una clàusula *where* com una combinació de taules en SQL. Exemples: **CODI 5.8**

→ Aquesta consulta mostra per cada departament del document *universitat.xml*, el nombre d'empleats que hi ha a cada lloc de treball. Es fa servir un *for* per a obtenir els nodes departament i el segon *for* per a obtenir els diferents llocs de cada departament.

```
for $dep in /universitat/departament
for $llo in distinct-values($dep/empleat/lloc)
let $comp := count($dep/empleado[lloc=$llo])
return <depart>{data($dep/nom)}<lloc>{data($llo)}</lloc>
<profes>{$comp}</profes></depart>

sortida
<depart>Informática<lloc>Asociado</lloc><profes>1</profes></depart>
<depart>Informática<lloc>Profesor</lloc><profes>1</profes></depart>
<depart>Matemáticas<lloc>Técnico</lloc><profes>1</profes></depart>
<depart>Matemáticas<lloc>Profesor</lloc><profes>2</profes></depart>
```

```

<depart>Matemáticas<lloc>Tutor</lloc><profes>1</profes></depart>
<depart>Análisis<lloc>Asociado</lloc><profes>2</profes></depart>

```

→ Aquesta consulta mostra per cada departament del document *universitat.xml*, el salari màxim i l'empleat que té aquest salari. El primer *for* obté els nodes departament i el segon *for* els empleats de cada departament. Per treure el màxim a la sortida es pregunta si el salari és el màxim.

```

for $dep in /universitat/departament
for $emp in $dep/empleat
let $emple := $emp/nom
let $sal := $emp/@salari
return if ($sal = $dep/max(empleat/@salari))
       then
         <depart>{data($dep/nom)}<salmax>{data($sal)}</salmax>
         <empleat>{data($emple)}</empleat></depart>
       else()

```

També es pot ficar els dos *for* de la següent manera:

```

for $dep in /universitat/departament, $emp in $dep/empleat

sortida
<depart>Informática
  <salmax>2300</salmax>
  <empleat>Alicia Martín</empleat>
</depart>
<depart>Matemáticas
  <salmax>2500</salmax>
  <empleat>Antonia González</empleat>
</depart>
<depart>Análisis
  <salmax>2200</salmax>
  <empleat>Mario García</empleat>
</depart>

```

→ Aquesta consulta mostra per cada lloc del document *universitat.xml*, l'empleat amb salari màxim i aquest salari. El primer *for* obté els diferents llocs de treball i el segon *for* obté els empleats que tenen aquest lloc de treball. Al *if* es pregunta si el salari de l'empleat és el salari màxim dels empleats de l'ofici del primer *for*.

```

for $llo in distinct-values(/universitat/departament/empleat/lloc)
for $emp in /universitat/departament/empleat[lloc=$llo]
let $sal := $emp/@salari
let $no := $emp/nom
return
if($sal = max(/universitat/departament/empleat[lloc=$llo]/@salari))
then <lloc>{data($llo)}<maxsalari>{data($sal)}</maxsalari>
     <emple>{data($no)}</emple></lloc>
else()

```

### **sortida**

```
<lloc>Asociado<maxsalari>2200</maxsalari><emple>Mario García</emple></lloc>
<lloc>Profesor<maxsalari>2300</maxsalari><emple>Alicia Martín</emple></lloc>
<lloc>Profesor<maxsalari>2300</maxsalari><emple>Pedro Paniagua</emple></lloc>
<lloc>Técnico<maxsalari>1900</maxsalari><emple>Juan Parra</emple></lloc>
<lloc>Tutor<maxsalari>2500</maxsalari><emple>Antonia González</emple></lloc>
```

També es pot resoldre la consulta fent servir un sol *for*, de la següent manera:

```
for $emp in (/universitat/departament/empleat)
let $lloc := $emp/lloc
let $sal := $emp/@salari
let $no := $emp/nom
order by $lloc
return
if($sal = max(/universitat/departament/empleat[lloc=$lloc]/@salari))
then <lloc>{data($lloc)}<maxsalari>{data($sal)}</maxsalari>
    <emple>{data($no)}</emple></lloc>
else ()
```

La primera consulta del punt «*Joins de documents*» es pot escriure amb dos *for* i el *where*:

```
for $emp in (/EMPLEATS/EMP_ROW), $dep in /departaments/DEP_ROW
let $emple := $emp/COGNOM
let $d := $emp/DEPT_NO
where data($d) = data($dep/DEPT_NO)
return <resul>{$emple, $d}{$dep/DNOM}</resul>
```

### **EXERCICI 5**

Fent servir el document *sucursals.xml*, cal fer les següents consultes XQuery:

- Obté per cada sucursal el saldo més gran haver-hi i el nom del compte que té aquest saldo.
- Obté per a cada sucursal el nom del compte del tipus ESTALVI el saldo deu del qual sigui el màxim. Treu també el màxim.

Fes servir els documents *productes.xml* i *zones.xml*

- Mostra els noms de productes amb el seu nom de zona. Fes servir dos *for* a la consulta.
- Mostra els noms de productes amb *estoc\_mínim* > 5, el seu codi de zona. El seu nom i el director d'aquesta zona. Fes servir dos *for* a la consulta.

### **>> Altes, baixes i esborrat de nodes en documents XML. CODI 5.9**

→ Aquesta consulta va a generar una sortida en la qual es vol incloure tots els nodes del document *departaments.xml*, més un nou node a inserir. A la variable **\$nou** s'afegeix el node a inserir, en **\$titol** es guarda el node *TITOL* del document *departaments.xml*. Al *return* es crea l'etiqueta anomenada *<departaments>* i s'inclou el títol, tots els nodes *DEP\_ROW* de *departaments.xml* (això s'obté amb el *for*) i el nou node. Si es vol guardar

la sortida generada a l'eina eXide cal clicar a save com es pot veure a la Figura 5.10:

```
let $nou := <DEP_ROW>
    <DEPT_NO>50</DEPT_NO><DNOM>COMPRES</DNOM><LOC>Reus</LOC>
</DEP_ROW>
let $titol := /departaments/TITOL
return <departaments>{$titol}
    {for $dep in (/departaments/DEP_ROW)
    return $dep}
    {$nou}
</departaments>
```

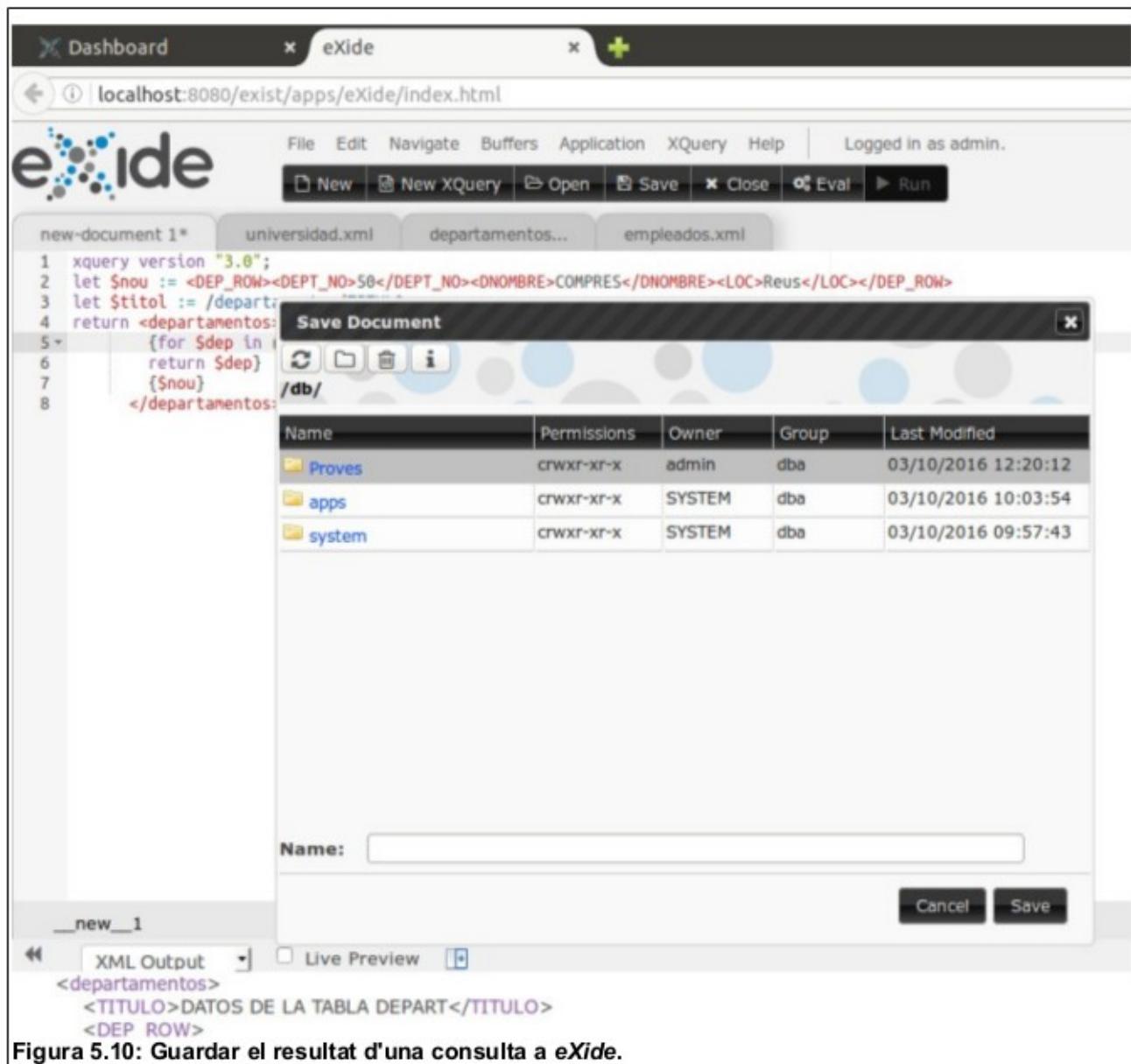


Figura 5.10: Guardar el resultat d'una consulta a eXide.

→ A la següent consulta s'elimina el node el nombre de departament del qual és el 10. Es fa servir el mateix document. Es tracta de generar una consulta que obtingui tots els nodes del document menys el node amb aquest número de departament. Es fa servir where per a no selecciona el departament 10:

```

let $titol := /departaments/TITOL
return <departaments> {$titol}
    {for $dep in (/departaments/DEP_ROW)
        where data((($dep/DEPT_NO)) != 10)
        return $dep
    }
</departaments>

```

→ A la següent consulta es vol actualitzar el departament el número de departament el qual és 20. Es vol canviar la localitat (*LOC*) per Valls. El que es fa és treure les dades del node a modificar, per després inserir aquest node modificat, en aquest cas es treu el nom del departament (el número i la localitat no es treuen doncs no es necessiten). Tot seguit es crea el node modificat a **\$modif** amb totes les etiquetes; de manera que quan es van generant les etiquetes amb els departaments quan arriba el departament número 20 es torna el node **\$modif**, si no és 20, es mostra el node **\$dep**:

```

let $titol := /departaments/TITOL
let $dno := /departaments/DEP_ROW[DEPT_NO=20]/DNOM/text()
let $modif := <DEP_ROW>
    <DEPT_NO>20</DEPT_NO><DNOM>{$dno}</DNOM><LOC>Valls</LOC>
    </DEP_ROW>
return <departaments> {$titol}
    {for $dep in (/departaments/DEP_ROW)
        return
            if ($dep/DEPT_NO/text()=20) then $modif
            else $dep
    }
</departaments>

```

→ A la següent consulta es vol actualitzar els salariis dels empleats del departament 10, se'ls incrementa a tots la quantitat de 200. El que es fa per a tots els empleats és treure els nodes a variables, de manera que si un empleat és del departament 10, es crearà un node nou <*EMP\_ROW*> amb les dades dels seus nodes i el salari actualitzat. Si l'empleat no és del departament 10, es retoma el node *EMP\_ROW* llegit:

```

for $em in /EMPLEATS/EMP_ROW
let $no := $em/EMP_NO, $cog := $em/COGNOM, $ofi := $em/OFICI, $di := $em/DIR
let $dat := $em/DATA_ALT, $de := $em/DEPT_NO, $sal := $em/SALARI
let $salnou := 200+number($em/SALARI/text())
return
    if ($em/DEPT_NO/text()=10) then
        <EMP_ROW>{$no, $cog, $ofi, $di, $dat }
            <SALARI>{$salnou}</SALARI>{$de}</EMP_ROW>
    else $em

```

→ A la següent consulta es fan els canvis necessaris per què la sortida sigui el document empleats.xml però amb el salari actualitzat:

```

let $titol := /EMPELATS/TITOL
return <EMPLEATS>
    {$titol}
    {
        for $em in /EMPLEATS/EMP_ROW

```

```

let $no := $em/EMP_NO, $cog := $em/COGNOM, $ofi := $em/OFICI, $di := $em/DIR
let $dat := $em/DATA_ALT, $de := $em/DEPT_NO, $sal := $em/SALARI
let $salnou := 200+number($em/SALARI/text())
return
    if ($em/DEPT_NO/text()=10) then
        <EMP_ROW>{$no, $cog, $ofi, $di, $dat }
            <SALARI>{$salnou}</SALARI>{$de}</EMP_ROW>
    else $em
}
</EMPLEATS>

```

## EXERCICI 6

Crea un nou document XML anomenat ***productes\_nou.xml***, que incorpori el següent producte: cod\_prod: 1023, denominació: HD USB 3.0 Sea 500GB, preu: 50, estoc\_actual: 80, estoc\_mínin: 20, i codi de zona ha de ser el codi de zona que correspon a Andalusia.

Crea un nou document XML anomenat ***productes\_30.xml***, amb el **<TITOL>DADES DE LA ZONA 30</TITOL>**, aquest document només ha de tenir els productes que tinguin 30 com el seu codi de zona.

Crea una consulta que mostri tot el document *productes.xml* però amb els següents canvis:

- Apuja el preu de cada producte un 3% més car.
- Afegeix 10 unitats als estoc\_actual de tots els productes.

## >> Sentències d'actualització de eXist. CODI 5.10

Aquestes sentències permeten fer altes, baixes i modificacions de nodes i elements en documents XML. Es pot fer servir les sentències d'actualització a qualsevol punt però es fa servir a la clàusula *RETURN* d'una sentència *FLWOR*, l'efecte de 'actualització' és immediat. Totes les sentències d'actualització comencen amb la paraula *UPDATE* i tot seguit la instrucció.

sentència	descripció
<pre>update insert &lt;zona&gt;&lt;cod_zona&gt;50&lt;/cod_zona&gt;&lt;nom&gt;Catalunya -SUD&lt;/nom&gt;&lt;director&gt;Alicia Ramos&lt;/director&gt;&lt;/zona&gt; into /zones</pre>	Insereix una zona a <i>zones.xml</i> , a l'última posició.
<pre>update insert &lt;compte tipus=&gt;PENSIONS&gt;&gt;&lt;nom&gt;Pepe Rubianes&lt;/nom&gt;&lt;num&gt;30506077&lt;/num&gt;&lt;aportacio &gt;400&lt;/aportacio&gt;&lt;/compte&gt; into /sucursals/sucursal[@codi=&gt;SUC1]&gt;</pre>	Insereix un compte al documents <i>sucursals.xml</i> del tipus PENSIONS a la sucursal SUC1.
<pre>for \$de in doc('file:///D:/XML/user/departamentsnous .xml')/departaments/DEP_ROW return update insert \$de into /departaments</pre>	Insereix en el document departaments de la base de dades els nodes DEP_ROW del document extern <i>departamentsnous.xml</i> localitzat a la carpeta <i>D:/XML/user/</i> .

Sentències d'actualització ***insert*** en eXist.

T a u l a 5 . 6

→ **insert**, es fa servir per a inserir nodes. El lloc de la inserció s'especifica amb: **into** (el contingut s'afegeix amb com a últim fill dels nodes especificats); **following** (el contingut s'afegeix tot just després els nodes especificats), o **preceding** (el contingut s'afegeix abans dels nodes especificats). El format és (exemples *insert* es poden veure a la *Taula 5.6*):

```
update insert ELEMENT into EXPRESIÓN
update insert ELEMENT following EXPRESIÓN
update insert ELEMENT preceding EXPRESIÓN
```

→ **replace**, substitueix el node especificat a NODE amb el VALOR\_NOU (veure format). NODE ha de tornar un únic ítem: si és un element, VALOR\_NOU ha de ser també un element. Si és un node de text o atribut el seu valor serà actualitzat amb la concatenació de tots els valors de VALOR\_NOU. El format és (exemples *replace* es poden veure a la *Taula 5.7*):

```
update replace NODE with VALOR_NOU
```

sentència	descripció
<code>update replace /zone/zona[cod_zona=50]/director with &lt;directora&gt;Pilar Cadenas&lt;/directora&gt;</code>	Canvia l'etiqueta director de la zona 50 i el seu contingut, al document zones.xml.
<code>update replace /departaments/DEP_ROW[DEPT_NO=10] with &lt;DEPT_ROW&gt;&lt;DEPT_NO&gt;10&lt;/DEPT_NO&gt;&lt;DNOM&gt;NOU10&lt;/DNOM&gt;&lt;LOC&gt;Amposta&lt;/LOC&gt;&lt;/DEPT_ROW&gt;</code>	Canvia el node complet DEP_ROW del departament 10, per les noves dades i les etiquetes que s'escriuen.
Sentències d'actualització <b>replace</b> en eXist.	T a u l a 5 . 7

→ **value**, actualitza el valor del node especificat en NODE amb un VALOR\_NOU. SI NODE és un node de text o atribut el seu valor serà actualitzar amb la concatenació de tots els valors de VALOR\_NOU. El format és (exemples *value* es poden veure a la *Taula 5.8*):

```
update value NODE with 'VALOR_NOU'
```

sentència	descripció
<code>update value /EMPLEATS/EMP_ROW[EMP_NO=7369]/COGNOM with 'Boadella Ramos'</code>	Canvia el cognom de l'empleat 7369, del document empleats.xml.
<code>update value /sucursals/sucursal[@codi='SUC3']/compte[1]/@tipus with 'NOUTIPUS'</code>	Canvia l'atribut tipus del primer compte de la sucursal SUC3, del document sucursals.xml.
<code>for \$em in /EMPLEATS/EMP_ROW[DEPT_NO=10] let \$sal := \$em/SALARI return update value \$em/SALARI with data(\$sal)+200</code>	Canvia el salari dels empleats del departament 10, del document empleats.xml, pujar 200.
Sentències d'actualització <b>value</b> en eXist.	T a u l a 5 . 8

→ **delete**, elimina els nodes indicats a l'expressió: **update delete EXPRESIÓN**

sentència	descripció
<code>update delete /zones/zona[cod_zona=50]</code>	Elimina la zona amb codi 50, al document <code>zones.xml</code> .
Sentències d'actualització <b>delete</b> en eXist.	T a u l a 5 . 9

→ **rename**, reanomena els nodes tornats a NODE (ha de retornar una relació de nodes o atributs) pel NOU\_NOM. El format és (exemples value es poden veure a la Taula 5.10):

`update rename NODE as NOU_NOM`

sentència	descripció
<code>update rename /EMPLEATS/EMP_ROW as 'fila_emple'</code>	Canvia de nom el node EMP_ROW, al document empleats.xml.
Sentències d'actualització <b>rename</b> en eXist.	T a u l a 5 . 1 0

## EXERCICI 7

A partir del document universitat.xml:

- Cal afegir un empleat al departament que ocupa la posició 2. Les dades són el salari 2550, el lloc Tècnic i el nom de l'alumne.
- Actualitzar el salari dels empleats del departament amb codi MAT1. Cal sumar al salari 150.
- Reanomena el node DEP\_ROW del document departaments.xml per filadepar.

## 5.4 ACCÉS A EXIST DES DE JAVA

Ja s'ha vist com llegir documents XML, accedint a la seva estructura i contingut fent servir els parser o analitzadors DOM (*Model d'Objectes de Documents*) i SAX (*API Simple per a XML*). En aquest apartat es volen veure diferents APIs que cedeixen a la base de dades eXist per a processar documents XML.

### 5.4.1 L'API XML:DB per a bases de dades XML

Des de 2001 el grup *XML:DB initiative* va començar el desenvolupament d'una interfície d'aplicació (API), independent del proveïdor par a bases de dades XML. A l'igual que amb altres aplicacions l'API simplement defineix un conjunt d'interfícies que cada proveïdor implementa. Actualment, l'API està sent utilitzada per la majoria de bases de dades. L'API proposada es basa en 3 paquets:

`org.xmldb.api`      `org.xmldb.api.base`      `org.xmldb.api.modules`

Per a conèixer més sobre aquests paquets i les seves classes cal anar als enllaços:

- <http://xmlDb.org.sourceforge.net/xapi/api/index.html> [link5.5]
- [http://exist-db.org/exist/apps/doc/devguide\\_xmlDb.xml](http://exist-db.org/exist/apps/doc/devguide_xmlDb.xml) [link5.6]

Els components bàsics fets servir pel codi *XML:DB API* són els drives, les col·leccions, els recursos i els serveis.

- Els **drivers** són implementacions de la interfície de base de dades que encapsula la lògica d'accés a la base de dades *XML*. Els proporciona el proveïdor del producte i ha de ser registrat amb el gestor de la base de dades. Exemple:

```
//Driver per a eXist
String driver = "org.exist.xmldb.DatabaseImpl";
//Carrega el driver
Class cl = Class.forName(driver);
//Instància de la BD
Database database = (Database) cl.newInstance();
//Registre del driver
DatabaseManager.registerDatabase(database);
```

- Una **col·lecció** és un contenidor de recursos i altres subcol·leccions. L'API defineix dos recursos diferents: **XMLResource** i **BinaryResource**. Un *XMLResource* representa un document *XML* o un fragment del document, seleccionats per l'execució d'una consulta *XPath*. Una vegada fet servir s'ha de tancar. Exemple:

```
//URI col·lecció
String URI = "xmlDb:exist://localhost:8080/exist/xmlrpc/db/Proves";
//Usuari
String usu = "admin";
//Contrasenya
String usuPass = "exemple";
col = DatabaseManager.getCollection(URI, usu, usuPass);
```

- Els **serveis** se sol·liciten per a tasques com a consultar una col·lecció amb *XPath* o la gestió d'una col·lecció. Exemples:

```
XPathQueryService servei =
    (XPathQueryService) col.getService("XPathQueryService", "1.0");
ResourceSet result =
    servei.query("for $em in /EMPLEADOS/EMP_ROW/[DEPT_NO=20] return $em");
```

De manera interna, *eXist* no distingeix entre les expressions *XPath* i *XQuery*. *XpathQueryService* i *XqueryService* són el mateix, tot i que la segona proporciona alguns mètodes addicionals.

Per a executar la consulta es crida al mètode **nom\_servei-query(xpath)**, aquest mètode torna un **ResourceSet**, que conté el recurs, a l'exemple anterior el resultat de la consulta es tornat en *result*. El següent codi s'escriu per a recórrer el recurs *result* tomat per la consulta anterior:

```
//Recórrer les dades del recurs
ResourceIterator i;
i = result.getIterator();
if (!i.hasMoreResources())
    System.out.println("LA CONSULTA NO TORNA RES");
while (i.hasMoreResources()) {
    Resource r = i.nextResource();
```

```

        System.out.println((String)r.getContent());
    }
}

```

Om ***result.getIterator()*** proporciona un iterador sobre el recurs, cada recurs conté el valor seleccionat per l'expressió *XPath*. El mètode ***getContent()***, torna el contingut del recurs, a l'exemple torna la consulta i el tipus és *String*.

Per a localitzar si hi ha un document a una col·lecció, es fa servir el següent codi:

```

col = DatabaseManager.getCollection(URI, usu, usuPass);
if (col==null)
    System.out.println("**** LA COL·LECCIÓ NO EXISTEIX ****");
}

```

Per a fer un programa que consulti la base de dades *eXist* s'ha d'incloure les següents llibreries (es poden trobar a la instal·lació de *eXist*):

- *exist.jar*
- *exist-optional.jar*
- *xmldb.jar*
- *xml-apis-1.4.01.jar*
- *xmlrpc-client-3.1.3.jar*
- *xmlrpc-common-3.1.3.jar*
- *ws-commons-util-1.0.2.jar*
- *log4j-1.2.17.jar*
- *xqjapi.jar*
- *xqj2-0.0.1.jar*
- *exist-xqj-1.0.1.jar*
- *exist-xqj-examples.jar*
- *\_xmlrpc-server-3.1.1.jar*

Al següent exemple ex pot veure com fer servir les classes ja vistes abans, fa una connexió amb la base de dades per a accedir al document *empleats.xml* i obtenir en *XML* els empleats del departament 10.

#### CODI 5.11

```

import org.xmldb.api.*;
import org.xmldb.api.base.*;
import org.xmldb.api.modules.*;

public class veureempleat10 {
    public static void main (String[] args) throws XMLDBException {
        //Driver per a eXist
        String driver = "org.exist.xmldb.DatabaseImpl";
        //Col·lecció
        Collection col = null;
        //URI col·lecció
        String URI = "xmldb:exist://localhost:8080/exist/xmlrpc/db/Proves";
        //Usuari
        String usu = "admin";
        //Contrasenya
        String usuPass = "exemple";

        try {
            //Carrega el driver

```

```

        Class cl = Class.forName(driver);
        //Instància de la BD
        Database database = (Database) cl.newInstance();
        //Registre del driver
        DatabaseManager.registerDatabase(database);
    } catch (Exception e) {
        System.out.println("Error en inicialitzar la base de dades exist");
        e.printStackTrace();
    }

    col = DatabaseManager.getCollection(URI, usu, usuPass);
    if (col==null)
        System.out.println("**** LA COL·LECCIÓ NO EXISTEIX ****");

    XPathQueryService servei =
    (XPathQueryService) col.getService("XPathQueryService", "1.0");
    ResourceSet result =
    servei.query("for $em in /EMPLEADOS/EMP_ROW/[DEPT_NO=20] return $em");

    //Recórrer les dades del recurs
    ResourceIterator i;
    i = result.getIterator();
    if (!i.hasMoreResources())
        System.out.println("LA CONSULTA NO TORNA RES");
    while (i.hasMoreResources()) {
        Resource r = i.nextResource();
        System.out.println((String)r.getContent());
    }
    //S'esborra
    col.close();
}
}

```

## EXERCICI 8

Cal fer els canvis necessaris a l'exercici anterior per a llegir del teclat un departament i mostrar els seus empleats. Cal fer servir l'entrada estàndard. El codi per a l'entrada estàndard és el següent:

```

//Import per a l'entrada estàndard
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

.....
//Es llegeix una dada tipus cadena
System.out.println("Escriu un departament: ");
String s = null;
try {
    BufferedReader in =
        new BufferedReader (new InputStreamReader(System.in));
    s = in.readLine();
} catch (IOException e) {
    System.out.println("Error en llegir");
    e.printStackTrace();
}
//Conversió a numèric
int dep = Integer.parseInt(s);

```

El programa fer en Java insereix, elimina i modifica departaments del document *departaments.xml*. Cal fer servir les sentències d'actualització de eXist. Les dades es llegiran de l'entrada estàndard del teclat. S'hi ha d'implementar la funció *main()* que cridi i executi els següents mètodes (no tornen res):

- ***insereixdep()***, aquest mètode llegirà un departament, el seu nom la seva localitat i hi haurà d'afegir-ho al document. Si el codi de departament existeix mostra un missatge indicant que no es pot inserir.
- ***esborradep()***, aquest mètode llegirà del teclat un departament i haurà d'esborrar si existeix, si no existeix mostra un missatge explicant-ho.
- ***modificadep()***, aquest mètode llegirà del teclat un departament, el seu nom nou i la localitat nova i haurà d'actualitzar totes les dades si existeix, en cas contrari mostra un missatge explicant-ho.

#### 5.4.1.1 Operacions sobre col·leccions i documents

L'API XML:DB permet a més de consultar documents a la base de dades, crear i eliminar col·leccions, i crear i eliminar documents.

- **Crear una col·lecció:** per a crear una nova col·lecció, es crea al mètode *createCollection* del servei ***CollectionManagementService***. El següent exemple crea la col·lecció NOVA\_COLLECCIO dins de la col·lecció col:

```
CollectionManagementService mgtService = (CollectionManagementService)
    col.getService("CollectionManagementService", "1.0");
mgtService.createCollection("NOVA_COLLECCIO");
```

- **Esborrar una col·lecció:** per a esborrar es fa servir el mètode *removeCollection*:

```
mgtService = (CollectionManagementService)
    col.getService("CollectionManagementService", "1.0");
mgtService.removeCollection("NOVA_COLLECCIO");
```

- **Crear un nou document:** aquest exemple, afegeix un document nou, a la col·lecció col, el document s'anomena *NOUS\_DEP.xml*, i es troba al disc dur, a la carpeta on es troba el programa. Es fa servir el paquet *java.io.File*, per a declarar ell fitxer a pujar a la base de dades, i el mètode *createResource* per a crear el recurs:

```
import org.xmldb.api.DatabaseManager;
import org.xmldb.api.base.Collection;
import org.xmldb.api.base.Resource;
import org.xmldb.api.base.XMLDBException;
import org.xmldb.api.modules.CollectionManagementService;
import java.io.File;
```

.....

```

File arxiu = new File("NOUS_DEP.xml");
if (!arxiu.canRead())
    System.out.println("ERROR en llegir el fitxer");
else {
    Resource nousRecursos =
        col.createResource(arxiu.getName(), "XMLResource");
    //Comprova si és un arxiu
    nousRecursos.setContent(arxiu);
    col.storeResource(nousRecursos);
}

```

- **Esborrar un document de la col·lecció:** aquest exemple esborra el document creat abans, també comprova si existeix. Es fa servir el mètode *removeResource*:

```

try {
    Resource recursosPerEsborrar = col.getResource("NOUS_DEP.xml");
    col.removeResource(recursosPerEsborrar);
} catch (NullPointerException e) {
    System.out.println("No es pot esborrar. No es troba.");
}

```

## EXERCICI 9

Cal fer un programa Java que generi la col·lecció GIMNÀS i posi els documents que es troben a la carpeta **ColleccioGimnas**. Els documents són:

- *socios\_gim.xml* conté activitats dels socis que assisteixen a fer esport a un gimnàs.
- *actividades\_gim.xml* conté informació de les activitats que poden fer al gimnàs. Hi ha 3 tipus d'activitat:
  1. Són activitats de lliure horari, el soci no paga quota addicional per fer-les, exemple aparell o piscina.
  2. Representen activitats que es fan en grup, exemple aerobic o pilates. El soci paga una quota addicional de 2€ per cada hora d'activitat.
  3. Representen activitats en la que es lloga un espai, per exemple pàdel o tennis. El soci paga una quota addicional de 4€ per cada hora que faci d'activitat.
- *uso\_gimnasio.xml* conté les activitats que fan els socis en el gimnàs durant l'any, cada fila representa una activitat feta pel soci amb la data (dd/mm/yy), l'hora d'inici (ex. 18) i l'hora de finalització (ex. 19).

A partir d'aquests documents cal fer un mètode Java per a obtenir per cada soci la quota que ha de pagar. Obtenir CODSOCI, la QUOTA\_FINAL.

Aquesta QUOTA\_FINAL serà igual a la suma de la QUOTA\_FIXA i les QUOTA\_ADDICIONAL que dependrà de les activitats fetes pel soci.

El programa Java ha de crear un document XML intermedi que obtingui la quota addicional de cada activitat feta pel soci, el document ha de tindre la següent estructura:

```

<dades><COD>xxxxx</COD><NOMSOCI>xxxxxxxxxx</NOMSOCI>
<CODACTIV>xxxxxxxxxx</CODACTIV><NOMACTIVITAT>xxxxxxxxxxxxx</NOMACTIVITAT>
<hores>xxx</hores><tipusacti>xxxxxxxx</tipusacti><quota_addicional>xxxx

```

```
</quota_adicional>  
</dades>
```

Cal afegir aquest document a la col·lecció GIMNÀS. Una vegada creat i afegit el document, el programa Java ha d'obtenir la quota final, que serà la suma de les quotes addicionals de les activitats més la quota fixa. Cal obtenir les següents etiquetes:

```
<dades><COD>xxxxx</COD>  
  <NOMSOCI>xxxxxxxxxx</NOMSOCI><QUOTA_FIXA>xxxxxxxxxx</QUOTA_FIXA>  
  <suma_quota_addic>xxxxxx</suma_quota_addic><quota_total>xxxxxx  
</quota_total>  
</dades>
```

#### 5.4.2 L'API XQJ (XQUERY)

L'API XQJ és una proposta d'estandardització d'interfície Java per accedir a bases de dades XML natives basades en el model de dades XQuery. L'objectiu és aconseguir un mètode senzill i estable d'accés a base de dades XML natives, aquest estàndard compleix els següents objectius:

- Estàndard independent del fabricant.
- Suporta l'estàndard XQuery 1.0
- Fàcil de fer servir.
- Potencia l'ús d'aquesta tecnologia a la comunitat Java.

Com també en JDBC la filosofia gira entorn de l'origen de dades i la connexió a aquest, i partint de la connexió poder llançar peticions al sistema.

Per a descarregar l'API cal anar a l'enllaç: <http://xqj.net/exist/> [link5.7]. Cal baixar el paquet *exist-xqj-api-1.0.1.zip*, dins es pot trobar a la carpeta **lib** un seguit d'arxius JAR que s'han d'afegir al *Build Paths*. Recordar que a *Eclipse* al projecte **botó dret** del ratolí i se selecciona **Build Paths>Configure Build Path...>Add External JARs...** llavors es selecciona tots els JARs del la carpeta **exist-xqj-api-1.0.1/lib**. Es pot executar el codi de comprovació que es pot veure a l'enllaç *link5.6*.

Es farà servir les següents interfícies per a connectar i obtenir les dades de la base de dades:

```
import javax.xml.xquery.XQConnection;  
import javax.xml.xquery.XQDataSource;  
import javax.xml.xquery.XQException;  
import javax.xml.xquery.XQPreparedExpression;  
import javax.xml.xquery.XQResultItem;  
import javax.xml.xquery.XQResultSequence;  
import javax.xml.namespace.QName;  
import net.xqj.exist.ExistXQDataSource;
```

#### Configurar una connexió:

- **XQDataSource**. Identifica l'origen de dades a partir del qual es crea la col·lecció. Cada implementació definirà les propietats necessàries per a fer la connexió. Per exemple, aques codi fa la connexió amb eXist, s'ha de posar el nom del servidor (*localhost*), el port de la base de dades (*8080*), l'usuari (*admin*) i la contrasenya (*exemple*). Tot i que només és obligatori la propietat *serverName* si s'està en local.

```
XQDataSource server = new ExistXQDataSource();
server.setProperty("serverName", "localhost");
server.setProperty("port", "8080");
server.setProperty("user", "admin");
server.setProperty("password", "exemple");
```

- **XQConnection**. Representa una sessió amb la base de dades, mantenint la informació d'estat, transaccions, expressions executades i resultats. Per exemple:

```
XQConnection conn = server.getConnection();
```

També es pot indicar l'usuari i la contrasenya que obre la sessió:

```
XQConnection conn = server.getConnection("admin", "exemple");
```

La connexió es tanca escrivint:

```
conn.close();
```

#### Processar els resultats d'una consulta:

- **XQPreparedExpression**: Objecte creat d'una connexió per a l'execució d'una expressió múltiples vegades. Torna un **XQResultSetSequence** amb les dades obtingudes. Igual que en **XQExpression**, l'execució es produeix cridant al mètode **executeQuery** i s'avalua tenint en compte el context estàtic en vigor.
- **XQResultSecuence**: Resultat de l'execució d'una sentència; conté, un conjunt de 0 o més **XQResultItem**. És un objecte que es pot recorre. Aquest exemple obté els empleats del departament 10, es fa servir **getItemAsString** perquè torni els elements com a cadenes:

```
XQPreparedExpression consulta;
XQResultSequence result;
consulta = conn.prepareExpression("/EMPLEADOS/EMP_ROW[DEPT_NO=10]");
resultat = consulta.executeQuery();
while (result.next()) {
    System.out.println("Element: "+resultat.getItemAsString(null));
```

- **XQResultItem**: Objecte que representa un element d'un resultat, immutable, vàlid fins que es crida al mètode **close** seu o de la **XQResultSetSequence** a la qual pertanyi. L'exemple anterior es pot escriure així:

```
consulta = conn.prepareExpression("/EMPLEADOS/EMP_ROW[DEPT_NO=10]");
resultat = consulta.executeQuery();
XQResultItem r_item;
while (resultat.next()) {
    r_item = (XQResultItem) resultat.getItem();
    System.out.println("Element: "+r_item.getItemAsString(null));
}
```

- **XQItem**: Representa un element en **XQuery**. És immutable i una vegada creat el seu estat intern no canvia.

- **XQSequence**: Representació d'una seqüència del mètode de dades *XQuery*, conté un conjunt de 0 o més *XQItem*. És un objecte que es pot recorre.

Amb *XQJ* no fa falta seleccionar la col·lecció dels documents *XML*, la cerca la fa en totes les col·leccions. Tampoc admet l'ús de sentències d'actualització d'*eXist*, amb la qual cosa les insercions, esborrats i actualitzacions seran bastant angoixoses. El següent exemple mostra les dades del document *productes.xml*, si n'hi ha productes en diferents, documents es mostrarà les etiquetes de tots els documents:

#### CODI 5.12

```
import javax.xml.xquery.XQConnection;
import javax.xml.xquery.XQDataSource;
import javax.xml.xquery.XQException;
import javax.xml.xquery.XQPreparedExpression;
import javax.xml.xquery.XQResultSequence;

import net.xqj.exist.ExistXQDataSource;

public class productes {
    public static void main (String[] args) throws XQException {
        XQDataSource server = new ExistXQDataSource();
        server.setProperty("serverName", "localhost");
        XQPreparedExpression consulta;
        XQResultSequence resultat;
        XQConnection conn = server.getConnection();

        System.out.println("----- Consulta documents productes.xml -----");
        consulta = conn.prepareExpression("for $de in /productes return $de");
        resultat = consulta.executeQuery();

        while (resultat.next()) {
            System.out.println("Element: "
                +resultat.getItemAsString(null));
        }
        conn.close();
    }
}
```

#### EXERCICI 10

Cal fer un programa *Java* que tingui una funció *main()* que llegeixi l'entrada estàndard el tipus de departament els empleats dels quals es volen mostrar. Ha d'invocar a mètode *mostrar(tipus)*, que rep el tipus i mostra els empleats dels departaments del tipus indicat. Si no té empleats o si el tipus no existeix, ha de mostrar un missatge que ho indiqui. S'ha de fer servir el document *universitat.xml*.

El següent exemple crea un fitxer *XML* al disc dur amb el nom *NOU\_EMPLA10.xml*, a partir de les dades extretes d'una consulta. Per a crear el fitxer es fa servir la classe **File** del paquet **java.io**. La consulta torna els empleats del departament 10, més el títol del document *empleats.xml*, de la base de dades, la consulta és la següent:

```

let $titol := /EMPLEATS/TITOL
return <EMPLEATS>{$titol}
    {for $em in /EMPLEATS/EMP_ROW[DEPT_NO=10] return $em}
</EMPLEATS>

```

El programa es pot veure tot seguit:

#### CODI 5.13

```

import java.io.BufferedReader;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import javax.xml.xquery.*;
import net.xqj.exist.ExistXQDataSource;

public class crearFitxerExtern {
    public static void main (String[] args) throws XQException {
        String nom_arxiu = "NOU_EMPE10.xml";
        File fitxer = new File(nom_arxiu);

        XQDataSource server = new ExistXQDataSource();
        server.setProperty("serverName", "localhost");
        server.setProperty("port", "8080");
        XQConnection conn = server.getConnection();
        XQPreparedExpression consulta = conn.prepareExpression
            ("let $titol := /EMPLEATS/TITOL return <EMPLEATS>{$titol} "+
            "{for $em in /EMPLEATS/EMP_ROW[DEPT_NO=10] "+
            "+ return $em}</EMPLEATS>");
        XQResultSequence result = consulta.executeQuery();
        if (fitxer.exists()) {
            //S'esborra el fitxer si existeix i es crea un de nou
            if (fitxer.delete()) System.out.println(
                "Arxiu esborrat. Es crea un de nou.");
            else System.out.println("Error en esborrar l'arxiu");
        }
        try {
            BufferedWriter bw = new BufferedWriter(
                new FileWriter(nom_arxiu));
            bw.write("<?xml version='1.0' encoding='ISO-8859-1'?>"+"\n");
            while (result.next()) {
                String cad = result.getItemAsString(null);
                //Mostrar
                System.out.println(" output "+cad);
                //Es grava el fitxer
                bw.write(cad+"\n");
            }
            //Es tanca el fitxer
            bw.close();
        } catch (IOException ioe) {ioe.printStackTrace();}
        conn.close();
    }
}

```

## EXERCICI 11

A partir dels documents productes.xml i zones.xml.

Cal fer un programa Java que faci un document extern amb nom zones20.xml. Ha de contenir els productes de la zona 20 i les següents etiquetes per a cada producte: <cod\_prod>, <denomicacio>, <preu>, <nom\_zona>, <director> i <estoc>, aquest estoc ha de ser calculat de l'estoc\_actual – estoc\_minim.

### 5.4.3 Tractament d'excepcions

Fins ara als exemples que s'han fet servir la paraula clau **throws** a la funció *main()* per a indicar que podia llençar l'excepció **XMLDBException** o **XQException**. Els següents exemples es tracta aquestes excepcions dins d'un bloc **try-catch**.

#### XMLDBEXCEPTION:

Aquesta excepció es llença quan es produeix un error a la API XML:DB. Conté dos codis d'error, un és el codi d'error XML de l'API, i l'altre és definit pel proveïdor específic. L'error del proveïdor vindrà definit per *ErrorCodes.VENDO\_ERROR*. **XMLDBException** esten **java.lang.Exception**.

Quan es produeix un error amb **XMLDBException** pot accedir a certa informació fent servir el mètode *getMessage()* que torna una cadena que escriu l'error.

Al següent exemple es pot veure com capturar els possibles errors que poden ocórrer:

#### CODI 5.14

```
import org.xmldb.api.*;
import org.xmldb.api.base.*;
import org.xmldb.api.modules.*;

public class provaExcepcions {
    protected static String driver = "org.exist.xmldb.DatabaseImpl";
    public static String URI = "xmlDb:exist://localhost:8080/exist/xmlrpc";
    private static Database database;
    private static String usu = "admin";
    private static String pwd = "exemple";
    private static Class cl = null;
    private static XPathQueryService service;
    private static ResourceSet result = null;
    private static Collection col = null;

    public static void main (String[] args) {
        try {
            cl = Class.forName(driver);
        } catch (ClassNotFoundException e) {
            System.out.println("No es pot trobar la classe del driver: "
                +e.getMessage());
        }

        try {
            database = (Database) cl.newInstance();
            DatabaseManager.registerDatabase(database);
        } catch (InstantiationException e) {
            System.out.println("Error instanciant el driver.");
        }
    }
}
```

```

} catch (NullPointerException e) {
    System.out.println("Error al instanciar la classe del driver: "
                       +e.getMessage());
} catch (IllegalAccessException e) {
    System.out.println("S'ha produït una IllegalAccessException.");
} catch (XMLDBException e) {
    System.out.println("Error XMLDB: "+e.getMessage());
}

try {
    col = DatabaseManager.getCollection(URI+"/db/Proves", usu, pwd);
} catch (XMLDBException e) {
    System.out.println("Error XMLDBException en getCollection: "
                       +e.getMessage());
}

try {
    service = (XPathQueryService) col.getService("XPathQueryService", "1.0");
} catch (NullPointerException n) {
    System.out.println("Error en getService, no es pot crear el servei.");
} catch (XMLDBException e) {
    System.out.println("Error XMLDBException, en getService: "
                       +e.getMessage());
}

//Consulta a la base de dades
try {
    result = service.query("for $b in
                           /EMPLEATS/EMP_ROW[COGNOM='TOVER'] return $b");
} catch (NullPointerException n) {
    System.out.println("Error en query,
                       no s'ha inicialitzat la base de dades o el servei.");
} catch (XMLDBException e) {
    System.out.println("Error XMLDBException, a la query: "
                       +e.getMessage());
}

try {
    ResourceIterator i;
    i = result.getIterator();
    if (!i.hasMoreResources()) System.out.println("La consulta no
                                                 torna res.");
    while (i.hasMoreResources()) {
        Resource r = i.nextResource();
        System.out.println((String) r.getContent());
    }
} catch (NullPointerException n) {
    System.out.println("Error en getIterator.
                       Problemes amb el servei.");
} catch (XMLDBException e) {
    System.out.println("Error XMLDBException, a getIterator: "
                       +e.getMessage());
}

try {
    col.close();
} catch (NullPointerException n) {
    System.out.println("Error en tancar la col·lecció.");
} catch (XMLDBException e) {
    System.out.println("Error XMLDBException, a col.close: "

```

```

        +e.getMessage());
    }
}

```

- L'error **NullPointerException**, és l'error que es dispara en tots els casos i sempre que no s'hagi inicialitzat la base de dades, és el cas d'escriure mal el driver.
- Si s'escriu malament l'*URI*, es dispara **XMLDBException** a l'hora d'assignar la col·lecció, en *getCollection*, i a partir d'aquí es disparen tots els **NullPointerException**, doncs el servei per aquesta col·lecció no s'ha pogut crear.
- Si s'escriu malament la carpeta on es troba la col·lecció, es dispara els **NullPointerException** següents, ja que no s'ha pogut crear el servei.
- Si s'escriu malament la query es dispara l'error **XMLDBException**, a l'hora de crear el *service.query*, al missatge mostra en quina línia de la consulta està l'error. Per exemple si s'escriu aquesta consulta:

```
result=service.query("ffor $b in /EMPLEADOS/EMP_ROW[APELLIDO='TOVAR'] return $b");
```

Mostra el següent missatge:

```
Error XMLDBException, a la query: Failed to invoke method queryP in class
org.exist.xmlrpc.RpcConnection: org.exist.xquery.StaticXQueryException:
exerr:ERROR org.exist.xquery.XPathException: err:XPST0003 unexpected
token: $ [at line 1, column 6]
```

## EXERCICI 12

Cal escriure i provar aquesta classe al vostre ordinador i prova les diferents situacions.

Per saber més sobre aquest tipus d'excepcions cal anar a l'enllaç: <http://xmlDb.org.sourceforge.net/xapi/api/org/xmlDb/api/base/XMLDBException.html> [link5.8].

### XQEXCEPTION:

Quan es produeix un error amb **XQException** es disposa de dos mètodes que permeten saber la causa de l'error i poder-ho arranjar. Aquests mètodes són **getMessage()** que torna una cadena que descriu l'error i **getCause()** que indica la causa de l'error per poder solucionar-ho.

Al següent exemple es pot veure com capturar els possibles errors a l'hora de sobretot fer consultes:

**CODI 5.15**

```
import javax.xml.xquery.XQConnection;
import javax.xml.xquery.XQDataSource;
import javax.xml.xquery.XQException;
import javax.xml.xquery.XQPreparedExpression;
import javax.xml.xquery.XQResultSequence;
import net.xqj.exist.ExistXQDataSource;

public class provaXQJExcepcions {
    public static void main (String[] args) {
        XQConnection conn = null;
        XQDataSource server = new ExistXQDataSource();
        XQPreparedExpression consulta = null;
        XQResultSequence rs = null, resultat = null;

        try {
            server.setProperty("serverName", "localhost");
        } catch (XQException e) {
            System.out.println("Error al servidor. Missatge: "
                               +e.getMessage());
            System.out.println("Error al servidor. Causa: "+e.getCause());
        }

        try {
            conn = server.getConnection();
        } catch (XQException e) {
            System.out.println("Error a la connexió: "+e.getMessage());
        }

        System.out.println("----- Exemple Consulta Productes -----");

        try {
            consulta = conn.prepareExpression("for $de in
                                              /productes/produc return $de");
        } catch (XQException e) {
            System.out.println("Error a la expressió. Missatge: "
                               +e.getMessage());
            System.out.println("Error a la expressió. Causa: "+e.getCause());
        }

        try {
            resultat = consulta.executeQuery();
            while (resultat.next()) {
                System.out.println("Element "+resultat.getItemAsString(null));
            }
        } catch (XQException e) {
            System.out.println("Error a l'execució. Missatge: "
                               +e.getMessage());
            System.out.println("Error a l'execució. Causa: "+e.getCause());
        }

        try {
            conn.close();
        } catch (XQException e) {
            System.out.println("Error en tancar la connexió.");
        }
    }
}
```

### **EXERCICI 13**

Cal escriure i provar al vostre ordinador les *XQException* i provocar situacions d'error, escrivint malament la consulta, el nom del servidor o tancant la base de dades.

Per saber més sobre aquest tipus d'excepcions cal anar a l'enllaç:  
<http://www.stylusstudio.com/xquery.html/docs/ddxq3.0/javadoc/com/ddtek/xquery3/XQException.html> [link5.9].