Generalitat de Catalunya
Departament d'Educació
Institut Milà i Fontanals. Igualada

Dossier

Informàtica

ICB0 Desenvolupament d'aplicacions multiplataforma

M06 Accés a dades

UF3 Persistència en BD natives XML

Diari d'activitats

Isidre Guixà / Pere Ollé

Curs 2023/24

Milà i Fontanals IGUALADA
Unió Europea
Fons Social Europeu
ISO 9001 BUREAU VERITAS Certification

SGBD XML instal·lats en escriptori Isard W10 – SGBD XML	27/10/23

- **BaseX 10.7** (necessita Java11 o superior) - https://basex.org

  Port per connectar via client-servidor: 1984
  Ports per connectar via HTTP: 8994 (start)/8995 (stop)
  Fins a versió 9, els ports via HTTP per defecte eren 8994/8995
  Des de versió 10, els ports via HTTP per defecte són 8080/8081.
  En aquest escriptori Isard amb BaseX 10.7 s'han configurat amb 8994/8995 per no usar 8080 que usa eXist-db
  Usuari: `admin` – Contrasenya: `admin`

  - Executeu *BaseX Server (Start)*. Per comprovar que s'ha engegat, obriu una consola `cmd` i executeu `netstat –ano`. La màquina ha d'estar escoltant pels ports 1984, 8984 i 8985 des de 0.0.0.0
  - Executeu *BaseX Server (Stop)* i comproveu via `netstat –ano` que ja no escolta pels ports.

  No cal tenir el servidor *BaseX* engegat per accedir localment a les BD usant *BaseX Gui*

- **eXist-db 6.2.0 -** https://exist-db.org

  Port HTTP: 8080 (també per connectar via client-servidor)
  Port SSL: 8043
  Usuari: `admin` – Contrasenya: `admin`

  L'execució *eXist-db Launcher* de l'escriptori posa en marxa un tauler de control per `eXist-db`, accessible per l'àrea de notificacions de Windows, que inclou, entre altres:
  - Opcions per engegar/aturar servidor
  - Opció per posar en marxa client Java per interactuar amb eXist-db

  Comproveu via `netstat –ano` que la màquina està escoltant pels ports 8080 i 8043 des de 0.0.0.0

  Cal tenir servidor *eXist-db* engegat per accedir a les BD (localment i remota)

- **Sedna 3.5.161 -** https://www.sedna.org

  Port 5050 (accés) i 5151 (ping). Cal tenir servidor engegat per accedir a les BD (localment i remota).
  Usuari: `SYSTEM` – Contrasenya: `MANAGER`

  Comproveu via `netstat –ano` que la màquina escolta pel port 5050 des de 0.0.0.0 i 5151 en localhost

  Alerta: Editant *EngegarSedna* es pot veure que cal usar paràmetre `–listen-address 0.0.0.0` per a que escolti per a qualsevol de les IPs de la màquina.

BD-XML natives. API Java específica del SGBD	
Apartats 1.1 i 1.2 del dossier Persistència en SGBD-XML a càrrec de l'alumne.	27/10/23
Introducció al SGBD-XML natiu eXist-db El dossier fa referència a una versió anterior a la que utilitzem, i el procés d'instal·lació és diferent, però excepte els apartats *Instal·lació* i *Primeres passes*, la resta és vigent. Només permet 1 BD, de nom db, que crea el procés d'instal·lació. Permet jerarquia de col·leccions.	
Introducció al SGBD-XML natiu BaseX El dossier fa referència a una versió anterior però és plenament vigent (gairebé tot idèntic) Permet vàries BD. No reconeix el concepte col·lecció però permet carregar fitxers en una jerarquia de carpetes, equivalent al concepte col·lecció.	
Introducció al SGBD-XML natiu Sedna El dossier es correspon amb la versió 3.5 que utilitzem. Permet vàries BD i cada BD permet vàries col·leccions com a filles de la BD, sense permetre crear una jerarquia de col·leccions, però es pot simular la jerarquia usant `/` dins el seu nom: `col1/col2`. Petit "bug" en Sedna-Admin: Si no hi ha cap BD creada, no apareix el node del servidor i no està activa l'opció per crear BD. Per aconseguir crear la primera BD cal intentar tornar a engegar el servidor des de Sedna Admin, donarà error si ja està engegat però llavors ja es pot crear BD.	

Pràctica (arxius `empresa.xml` i `mondial.xml` a Classroom) **pel 30/10/23**

Preparar cada SGBD amb l'estructura següent:

	Base de Dades	Col·lecció	Arxiu
BaseX	BD (a crear)		`economia/empresa.xml`
			`geografia/mondial.xml`
eXist-db	db (per defecte)	`economia`	`empresa.xml`
		`geografia`	`mondial.xml`
Sedna	BD (a crear)	`economia`	`empresa.xml`
		`geografia`	`mondial.xml`

Requeriments previs per poder cursar aquesta UF

Coneixement dels llenguatges XPath i XQuery

Diferències, segons SGBD, a l'hora d'accedir/gestionar la informació dins les BD

- **BaseX**: La interfície gràfica permet gestionar una BD oberta o tancada.

Per efectuar recerca sobre un document XML concret d'una BD (oberta o no):

```
doc('pathIntroduintNomBD')
```

Per exemple:

```
doc('BD/economia/empresa.xml')...
```

Si s'utilitza la funció `doc` i el camí no inclou el nom de la `BD`, concatena el camí a la carpeta on ha estat instal·lat BaseX, per exemple: `C:\BaseX`

Si no s'indica la funció `doc`, cal tenir una BD oberta, sobre la qual s'aplica la cerca.

- **eXist-db**: Cal tenir el servidor engegat.

Per efectuar recerca sobre un document XML concret:

```
doc('/db/path')
```

Per exemple:

```
doc('/db/economia/empresa.xml')...
```

Si no s'indica la funció `doc` o si camí no comença amb `/db`, la cerca s'efectua en el punt on s'està ubicat.

- **Sedna**: Cal tenir el servidor engegat i la BD oberta i tenir sessió oberta amb la BD.

Per efectuar recerca sobre un document XML concret:

```
doc('nomDocument','nomCol·lecció')
```

Per exemple:

```
doc('empresa.xml',economia ')...
```

Cal indicar la funció `doc`. Si no s'indica la col·lecció, la cerca s'efectua a l'arrel de la BD.

- **BaseX & eXist-db & Sedna**

Per trobar els documents existents en una determinada BD o col·lecció:

```
for $i in collection('path')
return document-uri($i)
```

Si no s'indica `path`:

- **BaseX**: Mostra els documents de la BD oberta.
- **eXist-db**: Mostra els documents a partir del punt on s'està ubicat
- **Sedna**: És obligatori indicar la col·lecció en el `path`. Si es vol conèixer els documents que conté l'arrel de la BD (fora de cap col·lecció i excloent el document de sistema):

25/10/22

```
for $i in doc('$documents')/documents/document
where $i/(@name)!="$db_security_data"
return $i/data(@name)
```

En **eXist-db**, per obtenir tots els documents de la BD fora de les col·leccions `system` i `apps` que incorpora eXist-db:

```
for $i in collection('/db')
where not(starts-with(document-uri($i),"/db/system"))
  and not(starts-with(document-uri($i),"/db/apps"))
return document-uri($i)
```

Pràctica d'instruccions d'actualització

Cal seguir la documentació del dossier Annex 07 – Instruccions Update XML

Exercicis desenvolupats sobre l'arxiu `empresa.xml` instal·lat en els diversos SGBD

1. Inserir un departament (inserir un node sencer): departament `d50` de `RRHH` a `Igualada`, després del departament de codi `d40`.

Sedna:

```
update insert
(<dept codi="d50"><nom>RRHH</nom><localitat>Igualada</localitat></dept>)
following doc('empresa.xml','economia')//dept[@codi="d40"]
```

exist-db:

```
update insert
(<dept codi="d50"><nom>RRHH</nom><localitat>Igualada</localitat></dept>)
following doc('/db/economia/empresa.xml')//dept[@codi="d40"]
```

BaseX: Com que la inserció només afecta un lloc (després del d40), no cal utilitzar FLWOR

```
insert node
(<dept codi="d50"><nom>RRHH</nom><localitat>Igualada</localitat></dept>)
after doc('BD/economia/empresa.xml')//dept[@codi="d40"]
```

2. Inserir atributs al departament d50: `pais` amb valor `ES` i `tel` amb valor `938030000`.

Sedna:

```
update insert (attribute pais {'ES'}, attribute tel {'938030000'})
following doc('empresa.xml','economia')//dept[@codi="d50"]/@codi
```

eXist-db:

```
update insert (attribute pais {'ES'}, attribute tel {'938030000'})
following doc('/db/economia/empresa.xml')//dept[@codi="d50"]/@codi
```

BaseX: Com que la inserció només afecta un lloc (d50), no cal utilitzar FLWOR

```
insert node (attribute pais {'ES'}, attribute tel {'938030000'})
after doc('BD/economia/empresa.xml')//dept[@codi="d50"]/@codi
```

L'anterior instrucció no funciona¿?¿?¿? En canvi, la següent sí:

```
insert node (attribute pais {'ES'}, attribute tel {'938030000'})
as last into doc('BD/economia/empresa.xml')//dept[@codi="d50"]
```

3. Inserir un node i un atribut en el departament d50: Atribut `fax` amb valor `938040000` i node `<adreça><carrer>Av. Emili Vallès, 4</carrer><cp>08700</cp></adreça>` en el departament de codi `d50`.

Com indica l'annex, en Sedna i eXist-db, per evitar problemes usem `into` i cada SGBD ubica on li sembla. Per "afinar" la ubicació, caldria fer-ho per separat...

30/10/23
03/11/23
06/11/23

Sedna:

```
update insert
(attribute fax {'938040000'},
 <adreça><carrer1>Av. Emili Vallès, 4</carrer1><cp>08700</cp></adreça>)
into doc('empresa.xml','economia')//dept[@codi="d50"]
```

eXist-db:

```
update insert
(attribute fax {'938040000'},
 <adreça><carrer1>Av. Emili Vallès, 4</carrer1><cp>08700</cp></adreça>)
into doc('/db/economia/empresa.xml')//dept[@codi="d50"]
```

BaseX: Com que la inserció només afecta un lloc (d50), no cal utilitzar FLWOR

```
insert node
(attribute fax {'938040000'},
 <adreça><carrer1>Av. Emili Vallès, 4</carrer1><cp>08700</cp></adreça>)
after doc('BD/economia/empresa.xml')//dept[@codi="d50"]/localitat
```

4. Canviar el nom del elements "adreça" per "domicili"

Sedna:

```
update rename doc('empresa.xml','economia')//adreça
on domicili
```

eXist-db:

```
update rename doc('/db/economia/empresa.xml')//adreça
as 'domicili'
```

BaseX: Com que només hi ha un departament amb `adreça`, no caldria utilitzar FLWOR, però la utilitzem i practiquem:

```
for $n in doc('BD/economia/empresa.xml')//adreça
return rename node $n as 'domicili'
```

5. Canviar el nom dels atributs "tel" dels "dept" per "telèfon"

Sedna:

```
update rename doc('empresa.xml','economia')//dept/@tel
on telèfon
```

eXist-db:

```
update rename doc('/db/economia/empresa.xml')//dept/@tel
as 'telèfon'
```

Alerta!!! Després d'executar aquesta instrucció, la BD pot quedar "k.o."... No apareix la localitat "Igualada" en el departament on hem canviat "tel" per "telèfon". ¿?¿? Reindexant s'arregla!!!

BaseX: La modificació només afecta un lloc (d50) i no cal usar FLWOR, però la usem i practiquem:

```
for $n in doc('BD/economia/empresa.xml')//dept/@tel
return rename node $n as 'telèfon'
```

6. Tots els departaments d'Igualada es traslladen a Tokio

Sedna: No permet substituir només els valors...

eXist-db:

```
update value
doc('/db/economia/empresa.xml')//dept[localitat='Igualada']/localitat
with 'Tokio'
```

o

```
update value doc('/db/economia/empresa.xml')//localitat[text()='Igualada']
with 'Tokio'
```

BaseX: Com que la modificació només afecta un lloc (d50), no cal utilitzar FLWOR:

```
replace value of node
doc('BD/economia/empresa.xml')//dept[localitat='Igualada']/localitat
with 'Tokio'
```

7. El departament de RRHH pateix una gran reestructuració i passa a ser:

codi="d99", nom="Festes", localitat="Marbella" i sense fax, ni telèfon ni país

Sedna:

```
update replace $n in doc('empresa.xml','economia')//dept[nom="RRHH"]
with
<dept codi="d99"><nom>Festes</nom><localitat>Marbella</localitat></dept>
```

eXist-db:

```
update replace doc('/db/economia/empresa.xml')//dept[nom="RRHH"]
with
<dept codi="d99"><nom>Festes</nom><localitat>Marbella</localitat></dept>
```

BaseX: Com que la modificació només afecta un lloc (d50), no cal utilitzar FLWOR:

```
replace node doc('BD/economia/empresa.xml')//dept[nom="RRHH"]
with
<dept codi="d99"><nom>Festes</nom><localitat>Marbella</localitat></dept>
```

8. Eliminar el departament de Festes

Sedna:

```
update delete doc('empresa.xml','economia')//dept[nom='Festes']
```

eXist-db:

```
update delete doc('/db/economia/empresa.xml')//dept[nom='Festes']
```

BaseX:

```
delete node doc('BD/economia/empresa.xml')//dept[nom
```

Com enviar/recuperar, via Java, codi XML amb caràcters d'escapament

La classe `StringEscapeUtils` de la llibreria `Apache Commons Text` facilita utilitats que podem usar per enviar/recuperar codi XML que contingui caràcters d'escapament. Concretament:

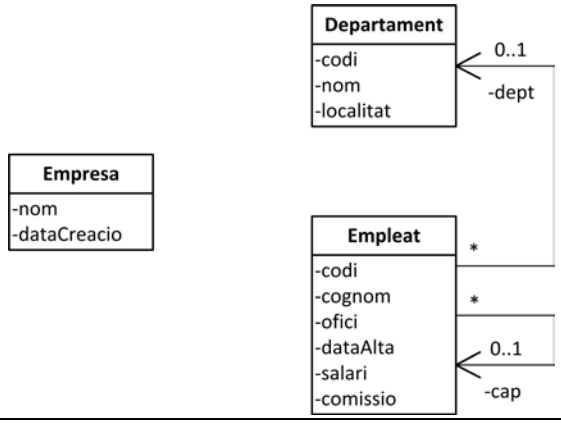
- Utilitat `escapeHtml4` per generar codi "escapat" a enviar a la BD
- Utilitat `unescapeHtml4` per recollir codi "escapat" recuperat de la BD i "desescapar-lo"

És a dir, si volem inserir `M&F` en algun element o atribut, la instrucció a enviar a la BD hauria d'incorporar `M&F`. Utilitzant `escapeHtml4` escriuríem:

```
StringEscapeUtils.escapeHtml4("M&F").
```

En recollir codi des de la BD, si s'utilitza la funció `string()`, el codi ja arriba "desescapat", però si no s'utilitza aquesta funció o s'utilitza la funció `text()`, el codi arriba "escapat" i podem usar la instrucció `StringEscapeUtils.unescapeHtml4(codiRecuperat)`.

La classe `StringEscapeUtils` es troba dins `Apache Commons Text` i per usar-la és requeriment tenir també instal·lada la llibreria `Apache Commons Lang3`.

Apartat del dossier: **1.3.1. API Java del SGBD BaseX** Els exemples del dossier contenen codi de la versió 7.1, mentre que nosaltres treballarem amb la versió 10.7. Algunes coses han canviat, així que: - Mireu les explicacions del dossier - Enlloc del codi del dossier, mireu el codi dels exemples facilitats en el projecte: `231108_1_ExemplesAPI_BaseX` Disposeu del `javadoc` de la versió 10.7 a l'apartat informació de l'aula de Classroom. Els exemples que acompanyen aquests materials i que utilitzin l'API Java per a BaseX, pressuposen l'existència d'una llibreria en el Netbeans anomenada exactament `BaseX 10.7`	08/11/23
Enunciat global per desenvolupar durant tota la UF Dissenyar capes de persistència per gestionar dades inherents a una empresa amb els seus departaments i empleats dissenyats en projecte `Empresa01` facilitat, que segueix el model de la imatge i que es pot trobar emmagatzemat en diversos SGBD-XML, en un fitxer XML (validable per `empresa.dtd` facilitat dins el projecte), de manera que: • Nom de la BD ha de ser configurable • Nom de la col·lecció on és ubicat ha de ser configurable. • Nom del fitxer XML ha de ser configurable 	10/11/23
Capa de persistència en SGBD-XML BaseX via l'API de BaseX Capa de persistència `EPBaseX` amb fitxer de propietats, constructors i mètode `close()` ➢ La capa de persistència ha de contenir un camp `ClientSession` (diguem-li `con` per connexió o com vulgueu) que mantingui la connexió amb la BD. ➢ El(s) constructor(s) de la classe son encarregats d'establir la connexió i guardar-la a `con`. ➢ Per establir la connexió, el(s) constructor(s) necessiten conèixer la màquina, el port, l'usuari i la contrasenya, que cercaran en un fitxer de propietats i així s'aconsegueix que la capa de persistència sigui independent de la màquina, port, usuari i contrasenya. ➢ I un constructor, com sap el nom del fitxer de propietats que ha de consultar? Normalment amb una de les dues possibilitats següents: o El fitxer de propietats té un nom prefixat, per exemple `EPBaseX.properties`. o El constructor rep, per paràmetre, el nom del fitxer de propietats. Implementarem les dues possibilitats, és a dir, la nostra capa tindrà dos constructors: o Un sense paràmetres que busqui el fitxer de propietats amb nom indicat. o Un amb 1 paràmetre que busqui el fitxer de propietats indicat per paràmetre ➢ El mètode `closeCapa()` de la capa de persistència ha de ser l'encarregat de tancar la sessió. ➢ Qualsevol error ha de generar excepció de la classe `EPBaseXException`. ➢ Si es creu necessari, el fitxer de propietats pot contenir més propietats que les indicades. ➢ TOTS els mètodes de la capa han d'estar correctament documentats. Projectes solució: `231110_1_EPBaseX` (capa) i `231110_1_EPBaseXApp` (programa prova)	10/11/23
Exercici per dilluns, 13 de novembre Implementar mètode: `public Empresa getEmpresa()` i comprovar el seu funcionament dins el programa de prova. Projectes solució: `231113_1_EPBaseX` (capa) i `231113_1_EPBaseXApp` (programa prova)	13/11/23

Comentaris als projecte: - Hem vist que dins la capa de persistència, a més de la variable `con` per mantenir la connexió, era convenient disposar d'una variable amb la ruta d'accés al fitxer. L'hem anomenat `path`. - La variable `path` s'ha de construir a partir del nom de la BD, la/les col·lecció/ons i el nom del fitxer. Podríem tenir aquesta informació per separat dins el fitxer de propietats i construir-la dins la capa (quan el constructor recupera la informació del fitxer de propietats), però sembla més adequat tenir ja la variable `path` emplenada correctament dins el fitxer de propietats. - El constructor, una vegada establerta la connexió i recuperada la variable `path`, comprova que el document indicat per `path` existeix dins la BD. Del contrari, tanca la connexió i avorta l'obertura de la capa de persistència. - En desenvolupar el mètode `getEmpresa` i donat que totes les dates dins l'XML han d'estar en format `yyyy-mm-dd`, es fa convenient disposar d'un objecte `SimpleDateFormat` creat permanentment (`static`) i així no haver-lo de crear a cada mètode on calgui fer algun tipus de conversió. - La classe `ClientQuery` conté mètode `close` per tancar l'objecte quan ja no sigui necessari i així alliberar recursos. Cal, doncs, invocar aquest mètode quan la `ClientQuery` ja no sigui necessària i per facilitar aquesta tasca, hem incorporat el mètode `private void tancarQuery(ClientQuery q)` que usarem sempre que correspongui tancar la `ClientQuery`.	

Exercici per dimecres, 15 de novembre
 	
Implementar mètode:
 	
`public Departament getDepartament(int codi);`
`/* Retorna NULL si no existeix un tal departament */`
 	
i comprovar el seu funcionament dins el programa de prova.
 	
Idea: Podeu usar l'API `JDom` per aconseguir de manera força fàcil, a partir d'un node `dept` obtingut via consulta XPath, la informació adequada per poder generar l'objecte `Departament`,
 	
Projecte solució: `231115_1_EPBaseX` (capa) i `231115_1_EPBaseXApp` (programa prova)
	15/11/23

Continuem la implementació de la capa de persistència EPBaseX amb mètodes:
 	
`public Empleat getEmpleat (int codi);`
`/* Retorna NULL si no existeix un tal empleat */`
 	
En aquest cas, si analitzem què implica recuperar un `Empleat` ens trobarem amb varis problemes:
 	
- Tenir objectes `Empleat` en memòria implica tenir també els corresponents objectes `Departament` i `Cap`, si s'escau, en memòria.
 	
- I si un mateix `Departament` és apuntat per varis empleats, és adequat tenir-lo "repetit" tantes vegades com empleats tingui o... en memòria mai hauríem de tenir repetit un mateix objecte?
 	
- I... imaginem que la relació entre `Departament` i `Empleat` o entre `Empleat` i `Empleat`, que segons model que estem seguint, són unidireccionals fossin bidireccionals. Això implicaria que a `Departament` tindríem accés a la col·lecció dels seus objectes `Empleat` i a `Empleat` tindríem accés a la col·lecció dels seus objectes `Empleat` subordinat i, per tant:
 	
- En carregar un `Departament` en memòria ens veuríem obligats a carregar tots els seus objectes `Empleat` i per cada `Empleat`, els seus objectes `Empleat` subordinat i per cada subordinat, que a la vegada pot tenir subordinants, els seus objectes `Empleat` subordinat i...
 	
- En carregar un `Empleat` en memòria ens veuríem obligats a carregar el seu `Departament` i tots els seus objectes `Empleat` subordinat...
 	
IMPOSSIBLE!!!!
 	
La solució passa per treballar amb "falsos objectes" que no carreguin les dades fins que les necessitem, com permet fer l'eina `Hibernate` sobre SGBDR (futura UF2 del M06). Però això són paraules majors!!!
	15/11/23

Proposta d'actuació per solucionar una mica les problemàtiques

Per a que la capa de persistència controli la no repetició d'objectes en memòria, afegir a la capa de persistència unes col·leccions `HashMap` que guardin els objectes `Departament` i `Empleat` carregats en memòria i també una referència a `Empresa` per guardar l'objecte `Empresa` si s'ha carregat en memòria. Concretament:

`Empresa empresa` per guardar l'objecte `Empresa`.
`HashMap<Integer,Empleat>` per guardar els `Empleat`, accedits pel seu codi
`HashMap<Integer,Departament>` per guardar els `Departament`, accedits pel seu codi

Refem els mètodes desenvolupats fins ara, gestionant el contingut d'aquestes variables i incorporem el mètode `getEmpleat` requerit.

Projectes solució: `231510_2_EPBaseX` (capa) i `231510_2_EPBaseXApp` (programa prova)

Exercici per divendres, 17 de novembre — 17/11/23

Continuem la implementació de la capa de persistència EPBaseX amb mètodes:

```
public int comptarSubordinats(int codi);
```

Compta la qtat. d'empleats dels que és cap l'empleat amb codi indicat

Alerta:
- Si no existeix l'empleat amb codi indicat, no pot retornar zero. Hauria de generar una excepció.
- I... com que és molt possible que calgui comprovar en més d'una ocasió l'existència d'un empleat... potser estaria bé disposar del mètode:

```
public boolean existeixEmpleat(int codi);
```

Projectes solució: `231117_1_EPBaseX` (capa) i `231117_1_EPBaseXApp` (programa prova)

Exercici per dilluns, 20 de novembre — 20/11/23, 22/11/23

Continuem la implementació de la capa de persistència EPBaseX amb mètodes:

```
public void eliminarEmpleat(int codi, int actCap);
```

Com el nom indica, intenta eliminar l'empleat amb el codi indicat. Però... I si aquest empleat té subordinats? Cal prendre alguna decisió, que la defineix el paràmetre `actCap`:
`actCap < 0`: Excepció
`actCap = 0`: Cal deixar als empleats subordinats sense cap (factible segons disseny de la BD)
`actCap > 0`: Canviar el cap de tots els seus subordinats pel nou cap indicat a `actCap`, comprovant:
- existeixi empleat amb codi `actCap`
- no sigui un subordinat (ni directe ni indirecte) de l'empleat a eliminar (es podria millorar... però no cal)

El programa de proves pot pressuposar que la BD conté el fitxer `empresa.xml` original, de manera que després d'executar-lo caldrà refer la BD amb l'arxiu original si es vol repetir la comprovació.

Per comprovar que `actCap` no sigui un subordinat (ni directe ni indirecte) de l'empleat a eliminar, pot ser interessant disposar del mètode:

```
public boolean esSubordinatDirecteIndirecte(int cap, int emp);
```

que, com el nom indica, comprova si empleat `emp` és subordinat (directe o indirecte) de `cap`.

Projectes amb aquest mètode: `231120_1_EPBaseX` (capa) i `231120_1_EPBaseXApp` (programa prova)

Mètode `eliminarEmpleat` usant `esSubordinatDirecteIndirecte` per dijous, 22 de novembre

Respecte les TRANSACCIONS en XQUF – Actualitzacions en BaseX

XQUF no facilita instruccions específiques per gestionar transaccions i s'entén que cada execució d'una instrucció XQUF és una transacció. En cas que vulguem executar en una transacció diverses instruccions, cal posar-les en seqüència, separades per una coma.

Això és el que cal aplicar en l'eliminació d'un empleat que té subordinats, per garantir que: - Eliminació d'empleat i eliminació del cap en els subordinats formin part d'una transacció - Eliminació d'empleat i canvi del cap en els subordinats formin part d'una transacció Recordeu que BaseX no permet efectuar modificacions via la seva API si la BD està oberta Projectes solució: `231122_1_EPBaseX` (capa) i `231122_1_EPBaseXApp` (programa prova)	

Apartat del dossier: **1.3.2. API Java del SGBD Sedna**
Exemples en projecte `231122_2_ExemplesAPI_Sedna`
Els exemples que acompanyen aquests materials i que utilitzin l'API Java per a Sedna suposen l'existència d'una llibreria en el Netbeans anomenada exactament: `Sedna 3.5.161`
 	
Observacions importants addicionals al què s'explica en el dossier:
 	
- El servidor SEDNA s'ha d'engegar escoltant per la IP que correspongui:
➢ Per accedir des de la mateixa màquina, cal engegar-lo amb
`se_gov -listen-address 127.0.0.1`
➢ Per accedir des d'una altra màquina, cal engegar-lo amb
`se_gov -listen-address IP_on_connectar`
`se_gov -listen-address 0.0.0.0` per permetre connexió des de qualsevol IP
- Si es vol que en engegar el servidor Sedna ja posi en marxa una BD concreta, es pot executar:
`se_sm nomBaseDeDades`
- Des de l'aplicació *SednaAdmin*, per connectar amb una BD cal indicar en el camp `host` la IP per on s'ha engegat el servidor.
- Sedna avorta la transacció activa en produir-se una `DriverException`.
- Sedna avorta la transacció activa, si existeix, en tancar connexió.
- Mentre s'està processant un `SednaSerializedResult`, NO es pot processar un altre `SednaSerializedResult`. En cas de fer-ho, el primer queda en un estat inconsistent, i en intentar continuar el seu procés, el programa queda "penjat" (no respon).
- En un `SednaSerializedResult`, els resultats a partir del 2n, comencen amb un salt de línia, que és diferent segons S.O. (`\r\n` en Windows, `\r` en Mac, `\n` en Linux).
	22/11/23

Capa de persistència en SGBD-XML Sedna via l'API de Sedna Desenvolupament de la capa de persistència en Sedna similar a la desenvolupada en BaseX. Per a que un programador que utilitzi la capa, pugui decidir quan vol fer `commit` i quan vol fer `rollback`, cal facilitar aquests mètodes (no existien a BaseX) i cal controlar en tot moment si hi ha una transacció oberta (doncs no es pot tornar a obrir si n'hi ha una d'oberta), de manera que: - Els mètodes, si no existeix transacció activa, obriran transacció i mai la tancaran. - Seran els mètodes `commit` i `rollback` els que tancaran la transacció activa. Per efectuar aquest control, podem creat la variable de classe `transOn` (`boolean`) que gestionarem en els mètodes de la capa i que serveix per saber si hi ha transacció oberta. Sempre que es produeix una `DriverException`, Sedna avorta la transacció activa i, en conseqüència, caldrà actualitzar la variable `transOn = false`.	24/11/23
Capa EPSedna – Fase 1 - Fitxer de propietats dins projecte amb programa de proves amb contingut adequat. - Constructors - Mètode3 `closeCapa` Observació: - A diferència de BaseX on cal tancar les `ClientQuery`, en Sedna no existeix aquest concepte. Projectes solució: `231124_1_EPSedna` (capa) i `231124_1_EPSednaApp` (programa prova)	24/11/23
Capa EPSedna – Fase 2 - Exercici per dilluns, 27 de novembre - Cada vegada que hem de començar un mètode, hem d'obrir transacció però abans cal comprovar si està oberta... Té molt sentit disposar d'un mètode privat `obrirTrans` que faci aquesta feina.	27/11/23

- Mètodes `commit` i `rollback` que no existien a BaseX - Mètodes `getEmpresa, getDepartament, getEmpleat, existeixEmpleat` Projectes solució: `231127_1_EPSedna` (capa) i `231127_1_EPSednaApp` (programa prova)	
Capa EPSedna - Fase 3 – Exercici per dimecres, 29 de novembre - Mètodes `comptarSubordinats`, `esSubordinatDirecteIndirecte`, `eliminarEmpleat` Projectes solució: `231129_1_EPSedna` (capa) i `231129_1_EPSednaApp` (programa prova)	29/11/23
Apartats del dossier: **2.1. API XQJ** Els materials del dossier fan referència a BaseX7.1 que incorpora una API XQJ amb deficiències. En aquest moment estem utilitzant BaseX9.4.6 que aporta l'API XQJ dissenyada per Charles Foster. Treballarem, doncs, amb aquesta nova versió. **SGBD XML** / **Classe que implementa XQDataSource** Sedna: `net.xqj.sedna.SednaXQDataSource` BaseX: `net.xqj.basex.BaseXXQDataSource` eXist-db: `net.xqj.exist.ExistXQDataSource` Oracle: `oracle.xml.xquery.xqjdb.OXQDDataSource` <= Funcionament horrible Podem descarregar-nos les versions actuals de les llibreries XQJ pels diversos SGBD: • Desenvolupades per Charles Foster i que es troben a http://xqj.net: - Llibreria XQJ per eXist-db: http://xqj.net/exist/exist-xqj-api-1.0.1.zip - Llibreria XQJ per BaseX: http://xqj.net/basex/basex-xqj-1.4.0.zip - Llibreria XQJ per Sedna: http://xqj.net/sedna/sedna-xqj-api.zip • **FYI** - Oracle: Veure Documentació 18c – Documentació 21c Les llibreries indicades a la documentació cal cercar-les dins la carpeta `dbhomeXE` d'on és instal·lat l'Oracle. Malgrat la documentació digui que cal usar JRE1.6, no cal. Es pot usar un Java superior i en cas que en execució aparegui error `java.lang.NoClassDefFoundError: javax/activation/MimeTypeParseException` motivat per que el paquet javax.activation ja no forma part de la versió Java que s'usa, cal incorporar la llibreria `activation` descarregable aquí, a més de les llibreries indicades per Oracle. ´/ En els exemples d'XQJ que segueixen, s'incorpora la seva execució en Oracle com a demostració que l'API XQJ per Oracle és totalment inoperant. En els diversos exemples es mostra la desfeta. Per poder executar els exemples, cal tenir instal·lat, en Oracle, el repositori XML-DB. El repositori XML-DB és un receptacle dins Oracle per incorporar documents XML. Oracle incorpora des de fa moltes versions, dues possibilitats d'incorporar codi XML: - Com a SGBDR-XML habilitat, incorporant codi XML dins taules utilitzant el tipus `XMLType` facilitat per Oracle, que permet tenir taula d'objectes `XMLType` o columnes d'objectes `XMLType`. , com els tipus que l'alumnat ha definit a M02-UF4, només que en aquest cas són tipus que ja porta incorporat l'Oracle. - Com a SGBD-XML natiu, com BaseX/eXist-db/Sedna que faciliten un receptacle on poder ubicar fitxers XML. És l'anomenat repositori XML-DB Explicació detallada de com accedir al repositori XML-DB i com introduir-hi documents, en guió adjunt `Oracle_GestióRepositoriXML-DB.sql` L'alumnat NO cal que es configuri el seu Oracle ni executi exemples... a no ser que tingui el cuquet... A classe es mostra el repositori amb els documents `mondial.xml` i `empresa.xml` ja incorporats dins les col·leccions `geografia` i `economia` respectivament. Els exemples que acompanyen aquests materials i que utilitzin l'API XQJ suposen l'existència de les llibreries següents en el Netbeans:	29/11/23

- `XQJ` que conté l'API XQJ1 (`xqjapi.jar` i `javadoc`) definida per JSR-225. L'estàndard, però, es queda curt i Charles Foster ha anat desenvolupant una versió 2, però no totes les API de la versió 2 estan en el mateix punt. Per això veurem diferents versions d'aquesta extensió.
- `XQJ Sedna`, que conté l'API específica per Sedna:
  `sedna-xqj-1.0.0.jar` i `xqj2-0.0.1.jar`
- `XQJ BaseX`, que conté l'API específica per BaseX:
  `basex-xqj-1.4.0.jar` i `xqj2-0.2.0.jar`
- `XQJ eXist-db`, que conté l'API específica per eXist-db:
  `exist-xqj-1.0.1.jar` i `xqj2-0.0.1.jar`
- `XQJ Oracle`, que conté l'API específica per **Oracle 18c** o **Oracle 21c** segons documentació vista.

En cas que en un projecte coexisteixi l'API XQJ BaseX amb alguna de les altres API XQJ (Sedna i/o eXist-db), cal que l'API XQJ BaseX estigui ubicada per damunt de les altres.

Problema en JDK17 i BaseX: El connector per BaseX necessita la llibreria Apache Xerces que JDK17 no incorpora. No es nota l'absència si s'afegeixen les llibreries d'Oracle (per què alguna la deu contenir), però sense les llibreries d'Oracle, en BaseX quan s'intenta executar `createExpression`, apareix error:

`java.lang.ExceptionInInitializerError`

causat per què no es pot instanciar la `SAXParserFactory`

`com.sun.org.apache.xerces.internal.jaxp.SAXParserFactoryImpl`

No passa en JDK11.

Per aquest motiu, els projectes d'aquests materials fan us de la llibreria:
- `Apache Xerces 2.12.2`, amb `xercesImpl.jar`.

Tutorial sobre XQJ: https://www.progress.com/xquery/resources/tutorials/xqj-tutorial

- `231129_2_ComEstablirConnexioViaXQL`: Mostra com aconseguir una connexió, però com que no estem creant cap `XQDataSource` específic, no assolim la connexió. Com que no connecta amb cap SGBD concret, no necessita cap dels connectors XQJ específics per un SGBD-XML.

Per connectar amb un SGBD, cal saber el nom de la classe `XQDataSource` del fabricant:

```
XQDataSource  xqs = new nomClasseXQDataSourceSegonsDriver();
```

Però això implica tenir el nom de la classe dins el codi... i això NO interessa, doncs llavors el programa només seria pel SGBD-XML corresponent. Projectes exemple:

- `231129_3_ProgramaXQJconnectantBaseX`, intenta connexió amb BaseX
- `231129_4_ProgramaXQJconnectantSedna` intenta connexió amb Sedna
- `221123_5_ProgramaXQJconnectantExist` intenta connexió amb eXist-db
- `221124_6_ProgramaXQJconnectantOracle` intenta connexió amb Oracle

Tots peten per què NO hem subministrat dades de connexió. El missatge d'error que donen no és el mateix, doncs cada driver fa les comprovacions que creu pertinents.

Si es vol desenvolupar capa o programa ÚNIC per qualsevol SGBD no es pot incloure el nom de la classe dins el codi ni fer `import` de la classe i cal, doncs, efectuar càrrega dinàmica de classes.

Exemples d'utilització de l'API XQJ en diversos SGBD

Projecte `231201_1_ExemplesAPI_XQJ_1` amb 8 configuracions d'execució preparades

- P1 (preparat amb 4 configuracions d'execució): Esbrina les propietats necessàries per connectar amb cada SGBD i que cal informar al `XQDataSource` abans d'establir connexió.
- P2 (preparat amb 4 configuracions d'execució): Estableix connexió (a partir de propietats en fitxers de propietats) i esbrina l'estat d'`autocommit` només connectar (mètode `getAutocommit`)

  En el fitxer de propietats per a cada SGBD incorporem el nom de la classe corresponent (`classeName`) al `XQDataSource` i totes les propietats per poder assolir la connexió.

01/12/23

Fixem-nos que la interfície `XQDataSource` incorpora un mètode `setProperty` que podem usar per anar assignant propietat a propietat... però també té mètode `setProperties` per assignar un objecte `Properties` amb totes les propietats. Aquesta segona opció és la que usarem, però primer caldrà eliminar del `Properties` que tenim en memòria la propietat `className` doncs assignar una propietat no vàlida provoca error en establir la connexió. Si comprovem en Oracle, informa que no té implementat `getAutocommit`. Desastre núm. 1	
Exemples de consultes (I) de l'API XQJ en diversos SGBD Projecte `231201_2_ExemplesAPI_XQJ_2` amb 8 configuracions d'execució preparades Observació: En el fitxer de propietats per a cada SGBD, hem afegit: - `path` amb la ruta on es troba el fitxer XML (que és diferent per a cada SGBD) En conseqüència, una vegada carregades les propietats en memòria i abans d'establir connexió, cal eliminar la propietat `path` de l'objecte `Properties` doncs provocaria error a la connexió. En Oracle, cal eliminar l'element `<DOCTYPE>` dels fitxers `.xml`, doncs els analitza erròniament. EJEM! El projecte inclou: - P1: Consulta per aconseguir els noms dels continents. - P2: Consulta **poc eficient** per aconseguir els noms dels països d'un continent, l'identificador del qual es demana a l'usuari. En projecte següent, versió eficient.	01/12/23

Exemples de consultes (II) de l'API XQJ en diversos SGBD – Per dilluns, 4/12
 	
Els SGBD (tots tipus si són eficients), es guarden en memòria les instruccions que van executant, amb el programa adequat per assolir el què la instrucció demana, de manera que, en rebre una instrucció:
 	
1. Miren si la sentència rebuda ja l'han analitzat-executat anteriorment i si és així, aprofiten el programa d'execució elaborat prèviament.
2. Si no la tenen en memòria, analitzen si la instrucció és correcta, elaboren un pla d'execució, se'l guarden en memòria i l'executen.
 	
Però per a què aprofitin un anàlisi anterior, cal que la sentència que es rep coincideixi totalment amb una sentència emmagatzemada. Així, imagineu que rep les sentències:
 	
`select * from xxx where codi = 20;`
`select * from xxx where codi = 30;`
`select * from xxx where codi = 40;`
 	
En aquest cas, la segona i tercera sentència NO poden aprofitar el programa que ha elaborat per la primera, per què NO estan exactament escrites... Per solucionar això i ser més eficients, les APIs que permeten connectar amb SGBD acostumen a incorporar la possibilitat de dissenyar consultes estàndards (preparades). En l'exemple anterior, podria ser similar a:
 	
`select * from xxx where codi = $variable;`
 	
Aquesta consulta s'envia al SGBD i ell en prepara l'execució, una única vegada i la podem executar tantes vegades com calgui, assignant prèviament valor a la variable, de manera que a cada execució no ha d'elaborar cap pla d'execució; només l'ha elaborat una vegada.
 	
Això és el què passa en el programa P2 de projecte anterior, doncs si l'usuari va demanant els països per a diversos continents, a cada canvi de continent, el SGBD ha de tornar a elaborar el pla d'execució.
 	
Projecte `231201_3_ExemplesApiXQJ_3`
 	
El programa P1 és una nova versió del programa P2 de l'anterior projecte on es defineix una consulta preparada (`XQPreparedExpression`). Fixem-nos que la consulta conté una variable (`$id`). El nom de la variable pot ser qualsevol.
 	
Abans d'executar la consulta preparada, hem d'enllaçar (`bindTipus`) la variable de la consulta (`id`) amb el valor que interessi (variable `idContinent` emplenada per l'usuari) i per fer aquest enllaç es necessita disposar d'un `XQItemType` del tipus corresponent a la variable.
	Cap de setmana

L'execució es fa via `executeQuery()` i el resultat és un `XQResultSequence` com en el programa P2 del projecte anterior.

Problemes detectats en Oracle (versions 11gR2 – 12c – 18c – 21c):

- `XQPreparedExpression`: Abans de cada execució, cal preparar-la de nou!!! Desastre núm. 2

Observacions importants abans de procedir a executar instruccions d'actualització – Pel 4/12

- `XQJ` incorpora `executeCommand` per executar qualsevol instrucció "no consulta" (similar a `executeUpdate` de JDBC).
- `XQJ` incorpora les instruccions `commit()` i `rollback()` per finalitzar les transaccions. No existeix el concepte de `beginTransaction`, és a dir, sempre s'està en una transacció, que finalitza en fer `commit` o `rollback`.
- `XQJ` avorta la transacció activa en cas de produir-se una `XQException`.
- `XQUF` és un llenguatge de consulta i, per tant, malgrat siguin instruccions de "no consulta", caldrà utilitzar mètode `executeQuery`.
- `XUpdate` i `Update` de Patrick Lehthi és un llenguatge de "no consulta" i caldrà utilitzar mètode `executeCommand`.
- El llenguatge XQUF no contempla el concepte de transacció i quan es vol executar vàries instruccions en una sola transacció, cal posar-les en seqüència, separades per `','` (coma).
- No tots els SGBD-XML són transaccionals (Sedna ho és, però eXist-db no).
- Els SGBD-XML amb llenguatge XQUF mai són transaccionals (BaseX i Oracle).
- En un SGBD no transaccional (XQUF o eXist-db), els mètodes `commit` o `rollback` no tenen sentit i la seva invocació no hauria de fer res, però segons el SGBD, podrien provocar una excepció. Per tant, millor no utilitzar-los en sistemes no transaccionals.

Si volem un programa (més endavant capa de persistència) que pugui treballar amb diversos SGBD, haurà de poder distingir:
- Si el SGBD és transaccional o no.
- Si el SGBD utilitza XQUF o llenguatge de Patrick Lehthi.

Per això, té sentit incorporar, en els projectes que segueixen i que pretenen poder ser usats en diversos SGBD-XML, dues propietats més en el fitxer de propietats:
- `updateVersion`, amb valors `XQUF` o `PL`.
- `transactional`, amb valors `N` o `Y`.

Cap de setmana

Operacions d'actualització via `XQPreparedExpression`?

- No es pot utilitzar `XQPreparedExpression` si les instruccions d'actualització són via `executeCommand`, com passa en Sedna i eXist-db.
- Sí es pot utilitzar `XQPreparedExpression` si s'utilitza XQUF com passa en BaseX i Oracle.

Problemes detectats en Sedna:

Després de fer un `commit` o `rollback`, no s'assabenta que està en una nova transacció (en XQJ això ha de ser automàtic doncs no hi ha forma d'obrir transacció) i:

- si s'intenta fer un nou `commit`, es genera excepció:
  `SE4610 There is no transaction to commit`
- si s'intenta fer un `rollback` o tancar la connexió (on intenta fer `rollback`), es genera excepció
  `SE4611 There is no transaction to rollback`

- Com actuar, pensant en Sedna, per saber si podem demanar `commit`/`rollback` o no, mentre no aparegui una nova versió d'API que solucioni aquest funcionament erroni?

  Doncs fent el mateix que fèiem en la capa de persistència per Sedna, via variable `transOn`, que tots els mètodes posessin a `cert` i fent `commit` o `rollback` si aquesta variable te valor `cert`.

  Donat que en produir-se una `XQException`, XQJ avorta la transacció activa, caldrà actualitzar la variable `transOn a false`.

• I com actuar en tancar la connexió? Dues possibles solucions: ➢ En tancar la connexió, interceptar i evitar l'excepció `SE4611`. És la solució adoptada en el 4t projecte d'exemple i la que es proposa usar per la capa ➢ Abans de fer un `commit` o `rollback`, generar una consulta simple-simple, per evitar l'error... **Problemes detectats en Oracle (versions 11gR2 – 12c – 18c – 21c)**: - Les instruccions XQUF aparentment s'executen però NO actualitzen res!!! Desastre núm. 3 Queda clar que no podrem utilitzar l'extensió XQUF d'Oracle!!! Quina llàstima!!!	
Exemples de sentències "no consulta" XQJ en diversos SGBD – Per dilluns, 4/12 Projecte `231201_4_ExemplesApiXQJ_4` Aquest projecte conté 3 programes exemple: - P1: Insereix un nou continent amb una `XQExpression` Incorpora la configuració d'execució per Oracle i es pot comprovar que no es queixa però no s'efectua cap modificació a la BD, com s'ha comentat més amunt. - P2: Insereix un país i l'inclou a les Nacions Unides, fet que volem gestionar en una sola transacció. El programa mostra com actuar quan una instrucció no és idèntica en els SGBD amb llenguatges d'actualització tipus "PL": intentar una versió i si no funciona intentar l'altra... - P3: Insereix un nou continent amb una `XQPreparedExpression` És una reversió de P1, però com s'ha dit més amunt, `XQPreparedExpression` no funciona en instruccions a executar via `executeCommand`. Per tant, aquest P3 només aporta una manera diferent de fer respecte P1 per als llenguatges XQUF.	Cap de setmana
Capa de persistència en SGBD-XML en SGBD amb connectors XQJ via l'API de XQJ Desenvolupament de la capa de persistència en XQJ similar a les desenvolupades. Cal comprovar funcionament en els SGBD BaseX – Sedna – eXist-db: - Un únic programa de prova (el mateix que en les capes anteriors) - Disposar de 3 configuracions d'execució diferents, cadascuna de les quals passi el fitxer de propietats corresponent al SGBD a comprovar.	04/12/23
Capa EPXQJ - Fase 1 – Exercici per dilluns, 4 de desembre Es facilita l'esquelet dels projectes `EPXQJ` i `EPXQJApp` equivalents als de les capes anteriors. La capa `EPXQJ` ja conté: - Fitxers de propietats adequats pels 3 SGBD-XML (BaseX-Sedna-eXist-db) - Constructors - Mètodes privats que puguin ser convenients (si cal, com `tancarQuery` en `EPBaseX` o `obrirTrans` en `EPSedna`) - Mètodes `closeCapa, commit, rollback` El projecte de prova EPXQJApp conté: - El programa ProvaEPXQJ és el què conté el joc de proves com en capes anteriors - Els altres 3 programes són per invocar ProvaEPXQJ amb el fitxer de configuració adequat. Desenvolupem la resta de mètodes: - Mètodes `getEmpresa`, `getDepartament` Projectes solució: `231204_1_EPXQJ` (capa) i `231204_1_EPXQJApp` (programa prova)	04/12/23
Capa EPXQJ - Fase 2 – Per dilluns, 11 de desembre - Desenvolupar la resta de mètodes, com en capes `EPBaseX` i `EPSedna`.	11/12/23

Consell: - En referència al mètode `eliminarEmpleat`, i donat que el procés d'actualització és diferent segons el tipus de SGBD-XML, és aconsellable –per la part diferent–, invocar mètode privat específic: `eliminarEmpleatXQUF` pels SGBD que actualitzen segons XQUF `eliminarEmpleatPL` pels SGBD que actualitzen segons PL	
FYI	
Observacions i curiositats a tenir en compte En el desenvolupament de les capes anteriors, hem pressuposat que les dades XML residien en un únic fitxer XML, però això no té per què ser així. En cas que les dades resideixin en més d'un fitxer, caldrà tenir carregats en memòria diverses rutes d'accés als fitxers.	
Connexió via protocol WebDav – BaseX / eXist-db / Oracle Cal seguir les instruccions del dossier Annex 04 – Utilització del client WebDav de Windows L'annex 05 – Utilització del client WebDav de NetDrive correspon a una versió antiga. A data d'avui, la instal·lació de la versió actual permet utilització de prova durant 7 dies i cal efectuar registre. Una vegada registrat, cal sol·licitar una llicència de prova i llavors es pot usar el programari. Les imatges no s'assemblen gens a les de l'annex 05, però els camps a emplenar i el funcionament és molt-molt-molt similar. Comprovat a data 26/02/2021.	