Memòria

Enginyeria Tècnica
Informàtica de Gestió.

TFC e-learning estàndards

Anàlisi de la qualitat de les metadades presents als documents LOM

Tutoria del projecte
Julià Minguillón Alonso

Autor
Daniel Rodríguez Calafat

Centre de realització del projecte
Universitat Oberta de Catalunya - UOC

Data
1er semestre 2004/05

Anàlisi de la qualitat de les metadades presents als documents LOM

Aquest projecte pretén realitzar l'estudi de la qualitat de les metadades presents als documents que representen objectes de coneixement LOM (Learning Object Metadata) mitjançat l'anàlisi de les dades presents a un repositori de documents LOM heterogeni, contribuït per una gran varietat de creadors de continguts educatius. Els documents LOM representen la informació disponible sobre un cert contingut educatiu essent una eina imprescindible per facilitar l'accés unificat a informació educativa distribuïda al llarg del mon. La correcta utilització de l'estàndard LOM ha de garantir que les eines d'accés als recursos educatius siguin útils i eficients. L'avaluació de les metadades presents a documents LOM redactats per persones sense formació específica en l'ús de l'estàndard LOM ha de permetre extreure conclusions relatives a la conveniència de l'adopció de l'estàndard a gran escala. L'extracció de dades dels documents LOM, basats en el llenguatge etiquetat de definició d'estructures jeràrquiques, XML (eXtensible Markup Language) i la reestructuració de les mateixes, necessària per l'estudi, es durà a terme en paral·lel amb l'anàlisi de la viabilitat del llenguatge XQuery (XML Query) com a eina de consulta per documents estructurats sota l'estàndard XML. L'anàlisi revelarà la importància d'XQuery com a substitut de les alternatives relacionals en la manipulació d'informació estructurada i posarà en evidència que XQuery permet l'accés i l'anàlisi a dades prescindint del format concret en que aquestes estan emmagatzemades. Finalment l'estudi presentarà resultats en forma d'eines d'anàlisi i resultats numèrics damunt el subconjunt de metadades relatives a la internacionalització dels documents LOM tot provant la dificultat del seguiment de l'estàndard i la escassa utilitat de les eines generadores de documents LOM. La conclusió general serà que els repositoris LOM són poc portables i no afavoreixen un repositori global de coneixement malgrat l'abast global de l'estàndard LOM. Un altra conclusió que extraurà el projecte és que l'accés a dades XML ha de passar forçosament per l'ús d'XQuery de forma que l'èxit de l'estàndard LOM passa per eines de consulta de repositoris LOM basades en XQuery envers altres alternatives menys portables.

Paraules clau: LOM, XML, XQuery, coneixement, aprenentatge, educació, e-learning.
Matèries de l'informe: 1. Estudi de l'estàndard LOM; 2.Estudi de tècniques tradicionals de manipulació de dades XML; 3. Estudi d'XQuery; 4. Jocs de proves i resultats.; 5. Conclusions.

Análisis de la calidad de los meta datos presentes en los documentos LOM

Este proyecto pretende realizar un estudio sobre la calidad de los meta datos presentes en los documentos que representan objetos del conocimiento LOM (Learning Object Metadata) mediante el análisis de de los datos presentes en un repositorio de documentos LOM heterogéneo, contribuido por una amplia variedad de creadores de contenidos educativos. Los documentos LOM representan la información disponible sobre un cierto contenido educativo siendo una herramienta imprescindible para facilitar el acceso unificado a información educativa distribuida a lo largo del mundo. La correcta utilización del estándar LOM debe garantizar que las herramientas de acceso a los recursos educativos sean útiles y eficientes. La evaluación de los meta datos presentes en los documentos LOM debe permitir la extracción de conclusiones relativas

a la conveniencia de la adopción del estándar a gran escala. La extracción de datos de los documentos LOM, basados en el lenguaje etiquetado de definición de estructuras jerárquicas, XML (eXtensible Markup Language) y la reestructuración de los mismos, necesario para el estudio, se llevara a cabo en paralelo junto al análisis de la viabilidad de XQuery (XML Query) como herramienta para la consulta de documentos estructurados bajo el estándar XML. El análisis revelará la importancia de XQuery como sustituto de las alternativas relacionales en la manipulación de información estructurada y pondrá en evidencia el hecho de que XQuery facilita el acceso a la información contenida en documentos XML mas allá de cual sea la estructura concreta con la que estos documentos han sido almacenados. Finalmente el estudio presentara resultados en forma de herramientas de análisis y resultados numéricos sobre el subconjunto de datos relativo a la internacionalización de documentos LOM demostrando la dificultad de seguimiento del estándar y la escasa utilidad de las herramientas generadoras de documentos LOM. La conclusión general será que los documentos repositorios de documentos LOM son poco portables y no favorecen la creación de un repositorio global de conocimiento a pesar de ser LOM un estándar de índole mundial. Otra conclusión que se desprenderá del estudio es que el acceso a datos almacenados en documentos XML ha de pasar forzosamente por el uso de XQuery de forma que el éxito de estándar LOM pasa por el uso de herramientas de consulta basadas en XQuery en lugar de otras alternativas menos portables.

Palabras clave: LOM, XML, XQuery, conocimiento, aprendizaje, educación, e-learning.

Quality analysis of present metadata in LOM documents

Project tries to make a study on the quality of data present in documents that represent objects of knowledge LOM (Learning Object Metadata) by means of the analysis of the present data in repositories from heterogeneous documents LOM, contributed by an ample variety of creators of educative contents. LOM documents represent the information available on a certain educative content being a tool essential to facilitate the unified access to distributed educative information throughout the world. The correct use of standard LOM must guarantee that the tools to access the educative resources are useful and efficient. Evaluation of data present in documents LOM must generate conclusions relative to the convenience for the adoption of the standard on great scale. The extraction of data from LOM documents, based on the markup language to definition of hierarchic structures, XML (eXtensible Markup Language) and the reconstruction of it, such necessary for the study, was carried out in parallel with the analysis of the viability of XQuery (XML Query) as tool for query document structured under XML standard. The analysis will reveal the importance of XQuery as substitute of the relational alternatives in the manipulation of structured information and will put in evidence the fact that XQuery beyond facilitates the access to the information contained in documents XML no matter as is the concrete structure with which these documents have been stored. Finally the study presented numerical results in form of analysis tools and results on the subgroup of metadata relative to the document LOM internationalization demonstrating the difficulty of pursuit the standard and the poor utility from the LOM generating document tools. The general conclusion will be that

the LOM repositories are not easily portable and they do not facilitate the creation of global knowledge repositories in spite of being LOM a standard for world-wide purposes. Another conclusion which it will be come off from the study is that the access to data stored in documents XML has to happen unavoidably through the use of XQuery so that the success of standard LOM happens through the use of querying tools based on XQuery instead of other less portable alternatives.

Keywords: LOM, XML, XQuery, knowledge, learning, education, e-learning

Índex

Glossari

LOM: Learning Object Metadata. Documents que descriuen objectes que poden tindre una utilitat educativa. Abasta des de programes complerts d'aprenentatge fins a una imatge d'una obra d'art.

XML: Extensible markup Language. És un llenguatge de definició de llenguatges estructurats en forma jeràrquica basat en etiquetes de text que delimiten l'estructura de les dades que representen. Aquesta mena de documents són molt portables, intel·ligibles per humans i en conjunt amb altres eines com DTD o XSLT tenen un camp immens d'aplicació.

E-learning: Relatiu als continguts educatius que poden ser distribuïts damunt la xarxa d'Internet.

XQuery: XML Query és un llenguatge emergent que pretén respondre a la necessitat que ha creat XML de tindre una manera d'accedir a les dades incloses als seus documents de forma tan senzilla com es fa amb SQL damunt les dades d'una base de dades relacional.

SQL: Llenguatge estructurat de consulta damunt bases de dades relacionals.

Base de dades relacional: És un repositori de dades distribuïdes en forma de taules. El terme relacional fa referència a d'inclusió de camps i/o taules auxiliars que mantenen dades referents a la relació entre les diferents dades emmagatzemades. Una de les principals qualitats de les bases de dades és la seva capacitat d'indexació de les dades que manipulen i la seva rapidesa en respondre consultes damunt un nombre molt elevat de camps.

Base de dades nativa XML: És un repositori de dades en format XML. Es distingeix d'un repositori clàssic de dades XML en que la base de dades realitza tasques d'indexació per a facilitar un accés eficient a les dades.

Prefaci

La finalitat dels documents LOM és crear un estàndard accessible a nivell global que pugui donar lloc a un repositori mundial de coneixement.

La dificultat de treballar amb documents lliurement estructurats i extensibles com LOM tant en el moment de la redacció com en el de la consulta dona lloc a aquest projecte d'investigació.

Els fets indican que LOM no te tant de seguiment ni és tan portable com s'hauria pogut esperar. Quines són les causes d'aquest fet? És un problema circumstancial degut a la falta d'eines especifiques per la redacció d'aquestos documents o és un problema relatiu a l'estàndard i la seva definició en si mateixa. És la llibertat i ambigüitat de l'estàndard la causa de la seva pobre adopció?

Es tractarà de respondre aquestes preguntes al llarg de l'estudi.

Introducció

Adaptar-se a l'estàndard LOM per part de companyies o entitats amb un gran nombre de dades als seus repositoris implica un gran esforç i no queden clars quins beneficis aporta l'adopció de l'estàndard.

Aquest fet possibilita que no hi hagi una tendència real cap a l'adopció de documents LOM. Hi ha pocs documents, anomenats LOM estricte, que compleixin totalment l'estàndard LOM. En lloc d'això trobem documents, anomenats LOM compatible, que respecten LOM tot introduint variacions personals i/o propietàries.

Un document compatible amb LOM és aquell que conté les etiquetes descriptives que defineix l'estàndard i algunes més depenent de la companyia que genera el document.

El perill resideix en que un document pot ésser compatible LOM però tindre gairebé totes les etiquetes pròpies de LOM buides i només emprar les etiquetes descriptives propietàries, emprades a forma d'extensió.

Les extensions són problemàtiques per dos motius principals:

a) S'empra una extensió quan LOM no satisfà un requeriment intern del contribuïdor.
b) L'ús d'extensions dificulta la creació d'aplicacions de cerca globals i fa tornar a la situació inicial on els repositoris només són accessibles de forma efectiva per buscadors dependents del repositori.

L'objectiu d'aquest projecte és analitzar les metadades presents a un conjunt considerablement gran i divers de documents LOM per tractar de descobrir possibles problemes derivats de:

a) Requeriments de LOM no satisfets o satisfets de forma complexa que acaben generant l'ús d'extensions.
b) Usos inadequats de LOM per part dels contribuïdors.
c) Requeriments de LOM totalment en desús.

Paral·lelisme

En familiaritzar-se amb els documents LOM sorgeixen problemes per entendre l'objectiu del projecte. Un possible paral·lelisme més proper seria el de l'estàndard HTML i les seves variacions.

- Les etiquetes ALT dins les imatges són un exemple d'etiqueta mal emprada per part dels usuaris.
- Les etiquetes blink van caure totalment en desús per la seva poca utilitat en pantalla.
- Les etiquetes div que havien de separar el documents, com les seccions d'un document de MSWord, i han acabat convertint-se en les etiquetes de les capes malgrat que la solució proposada per Netscape amb l'extensió i l'afegit de l'etiqueta layer era més adequada.

LOM és als repositoris globals de coneixement el que HTML és a la web.

Mètode utilitzat per a la realització del treball

En primer lloc l'estudi obtindrà un conjunt suficient de documents LOM. Seguidament les dades contingudes dins aquestos documents seran processades per a facilitar-ne l'anàlisi. Finalitzada la recaptació d'informació l'anàlisi extraurà les dades valoratives i procedirà a extreure conclusions.

Fase d'observació i presa de contacte

Anàlisi de la problemàtica

Per a poder realitzar un estudi aprofundit sobre els documents LOM es requereixen certs requisits previs.

1. Els documents LOM són documents XML. En aquest sentit es necessari conèixer l'estructura dels documents XML. És aconsellable conèixer DTD i XSLT.
2. És necessari conèixer LOM en detall i conèixer com s'ha de realitzar la redacció de documents LOM.
3. Per realitzar l'anàlisi és aconsellable tindre coneixements d'alguna llibreria d'extracció de dades de documents XML com ara SAX o JAXB sobre JAVA.
4. Per realitzar consultes damunt les dades contingudes als documents LOM és necessari familiaritzar-se amb XQuery.
5. Un requisit bàsic per la realització del projecte és el repositori de dades LOM sobre el que realitzar les proves. De la qualitat del repositori en depèn que les dades obtingudes siguin representatives de la població.

Abans d'iniciar el TFC l'estudiant comptava amb nocions intermèdies de XML i havia treballat amb el suport de XML que proveu el llenguatge Python. Degut a la bona documentació i a la portabilitat i robustesa de JAVA l'estudiant decideix treballar amb el llenguatge de programació JAVA i la llibreria de manipulació de dades XML anomenada SAX.

Tots els coneixements relatius a LOM es troben redactats a l'estàndard de redacció de documents LOM.

L'estudiant inicia el projecte sense coneixements sobre XQuery. Essent la facilitat de l'estudiant per familiaritzar-se amb aquest nou llenguatge una tasca crítica en el desenvolupament del projecte.

El tutor procura a l'alumne 3500 documents LOM provinents del repositori d'ARIADNE. Aquestos documents no eren pas documents LOM natius, sinó documents en format ARIADNE que havien estat transformats en documents LOM de forma mecanitzada mitjançant un algorisme de conversió ideat per membres del projecte ARIADNE.

Fase d'intervenció

El tutor de projecte considera que la mostra de documents LOM que ha procurat inicialment a l'estudiant no és suficient per realitzar un bon estudi. El tutor del projecte encarrega a l'estudiant la cerca de nous repositoris que millorin la qualitat de l'estudi amb una mostra més representativa de la població.

L'estudiant estableix contactes amb personal responsable de diversos repositoris de documents LOM i aconsegueix una mostra de 4000 documents LOM sota condició de no fer-los públics i emprar-los exclusivament per a realitzar el projecte d'estudi.

El tutor és advertit de les condicions en que són cedits els documents i n'aprova la seva utilització.

El procés d'adquisició de la mostra adequada consumeix més temps del previst degut a la poca difusió dels documents LOM i a la falta de cooperació per part dels repositoris privats d'altres universitats o centres culturals. Els documents són fàcilment consultables, però hi ha reticència a cedir les fonts dels mateixos.

L'estudiant realitza una primera aproximació al llenguatge XQuery i aquest li genera una gran desconfiança. Es tracta d'un llenguatge nou, amb poques implementacions en el mercat. L'estudiant decideix realitzar un bolcat de dades XML a format relacional per a tal de poder-les consultar amb el llenguatge SQL que li és molt més familiar.

L'estudiant inicia el projecte recreant una estructura relacional per albergar les dades contingudes als documents LOM que ha de processar.

Passades dues setmanes l'estudiant es troba amb un problema que no havia previst en iniciar el projecte. La llibertat d'estructura d'un document XML LOM fa que sigui necessària una taula per a cada node de l'estructura jeràrquica que conforma el document LOM.
Nota: consultar el resultat d'aquesta fase prèvia a l'apèndix I d'aquest mateix document.

L'estudiant es troba amb un nombre de taules proper a 60. Les relacions entre les taules són complexes i la seva manipulació via consultes SQL és inviable si volem extreure un gran nombre de dades d'un document LOM o recompondre'l. La reconstrucció d'un sol document LOM implica una consulta sobre almenys 60 taules diferents, sense tindre en compta que les relacions de node/subnode es tornen relacions mestre/detall al model relacional tot complicant encara més les consultes.

Tot i el gran nombre de taules i la inclusió de camps "source" i "language" per a cada camp susceptible de contindre variacions de l'estàndard la solució basada en bases de dades relacionals no és prou flexible per manipular documents LOM no estrictes. Es requeriria una generació dinàmica de l'estructura de les taules que han de contindre el document processat.

L'estudiant inicia aleshores una recerca que hauria de contestar la pregunta: com han resolt aquesta problemàtica altres usuaris de documents LOM?

L'estudiant examina programaris editors de documents LOM, cercadors de repositoris LOM, eines JAVA per fer bolcats de documents LOM a objectes JAVA i altres eines relacionades amb el procés de llegir, escriure, cercar documents LOM.

La conclusió a la que arriba és que es fa un ús simplista i/o segmentat de l'estàndard per fer-lo manipulable en termes d'estructures de dades tancades i inextensibles dinàmicament.

Traçant un nou paral·lelisme amb HTML és com si els editors HTML només permetessin ficar 3 imatges a cada document i redactar taules amb un nombre màxim de 5 columnes.

Se'n dedueix que transportar tota la complexitat dels documents LOM al camp de les bases de dades relacionals és una tasca complexa i previsiblement ineficient que ningú du a terme.

Havia arribat el moment de replantejar l'ús d'alguna alternativa per consultar dades de documents LOM de forma més extensible i senzilla.

Mentre l'estudiant cerca nous documents LOM per a realitzar l'estudi estableix comunicacions amb Norman d'Arcy de la Universitat de Calgary. Aquestes breus converses tenien per intenció convèncer al personal de la universitat de que cedissin documents LOM a l'estudi. Fruit d'aquestes converses l'estudiant és instat a abandonar SQL com a eina de consulta per documents LOM i se l'incita a utilitzar XQuery amb el que s'han obtingut bons resultats a la Universitat de Calgary.

L'estudiant comunica al seu interlocutor la dificultat que té per trobar implementacions de XQuery que responguin a les seves expectatives. En resposta a aquesta inquietud l'estudiant és convidat a provar dues solucions XQuery disponibles al mercat: XStreamDB i eXist.

L'estudiant avalua inicialment les dues opcions abans d'aprofundir en l'ús de les eines i opta per eXist. Estimant preferible una solució de programari lliure que una solució propietària amb límit temporal d'us com la que oferta XStreamDB.

El canvi cap a eXist resulta molt profitós. L'estudiant crea la base de dades i importa el 7500 documents LOM de que disposa. En aquestos moments ja és possible realitzar consultes damunt la totalitat dels documents LOM. XQuery facilita l'accés a les dades emmagatzemades als documents LOM de la mateixa manera que SQL a una base de dades relacional.

L'estudiant detecta una dificultat inesperada en el tractament dels documents LOM de que disposa però en troba la solució: s'ha de treure la referència a espais de noms externs que es fa a les metadades de l'etiqueta <lom/> dels documents LOM. Aquesta referència és perfectament correcta als documents LOM però interfereix amb el motor de bases de dades del programari eXist.

Aquest document no és accessible per eXist.

```
<lom xmlns="http://ltsc.ieee.org/xsd/LOMv1p0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://ltsc.ieee.org/xsd/LOMv1p0
http://rubens.cs.kuleuven.ac.be:8989/ariadne/xs/lom.xsd">
```

Aquest document és accessible per eXist:

```
<lom xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ltsc.ieee.org/xsd/LOMv1p0
http://rubens.cs.kuleuven.ac.be:8989/ariadne/xs/lom.xsd">
```

L'estudiant empra l'eina shareware XReplace per efectuar els canvis damunt els 7500 documents LOM de que disposa i torna a generar les bases de dades abans de continuar.

L'estudiant realitza diverses consultes senzilles damunt el repositori de dades i no se'n adona de que els seus avanços no estan lligats a la finalitat del projecte.

L'estudiant pot extreure amb facilitat el llistat de continguts educatius escrits en anglés que tinguin un autor anomenat "William". Les consultes clàssiques tenen un alt interès per aquella gent que accedeix a un repositori LOM cercant continguts educatius, però no li permeten extreure dades significatives per la realització de l'anàlisi de la qualitat dels documents. En particular no pot respondre qüestions relatives a l'estructura del document LOM, els seus camps buits, quins nodes essencials s'han suprimit o quins nous camps no estàndards s'han afegit.

L'estudiant comunica les seves angoixes al tutor de projecte que li dona indicacions respecte de quines consultes hauria de poder realitzar. En particular l'estudiant és convidat a manipular atributs nuls i metadades buides. En aquest sentit s'ha de puntualitzar que donat el següent text XML:

```
<arrel>
     <node id=”1”>
          <camp1></camp1>
          <camp2/>
     </node>
     <node id=”2”>
          <camp3>valor</camp3>
     </node>
</arrel>
```

Es consideren camps buits d'un document XML els camps 1 i 2 del node amb id=”1”.
Ara bé, en termes d'anàlisi s'hauria de considerar també que el node amb id=”1” te un camp3 buit i que el node amb id=”2” te els camps 1 i 2 buits. Els camps no emprats no són considerats camps buits per cap processador de documents XML tot i que aquesta consideració és molt important per l'anàlisi que es vol dur a terme.

L'estudiant prova aleshores de manipular atributs nuls i metadades buides seguint els exemples i tutorials de llenguatge XQuery que troba a Internet i a llibres de consulta. Malauradament la implementació d'XQuery oferta per part d'eXist difereix de la sintaxi descrita als tutorials. La documentació d'XQuery no és gaire clara en aquest aspecte.

XQuery ofereix la possibilitat d'ampliar el llenguatge emprant funcions. L'estudiant estima necessari aprendre a escriure funcions per millorar el seu accés a les dades en el

context d'XQuery. L'estudi de llibres comercials d'XQuery obre a l'estudiant un ampli espectre d'oportunitats i el fa ser optimista damunt els resultats que podrà obtindre.

Al llarg de tota aquesta fase d'aprenentatge l'estudiant no és conscient de que eXist no està preparat per suportar l'extensió d'XQuery mitjançant funcions. Convé puntualitzar que eXist és un projecte viu que potser no estava preparat en el moment de redactar aquest document, Gener del 2005, però podria estar-hi en un moment posterior.

La única extensió d'XQuery viable passa per emprar el motor de bases de dades d'eXist des de dins d'una aplicació JAVA.

L'API d'eXist per a JAVA resulta massa complexa i difícil de fer funcionar. El seu funcionament passa per l'ús de servlets de JAVA i comunicacions client/servidor.

Decidit a emprar una API d'XQuery per a JAVA l'estudiant decideix provar XQEngine projecte del que ha llegit bons comentaris en els llibres d'XQuery que ha estudiat.

Amb XQEngine l'estudiant troba la resposta a les seves necessitats. La potència de programació de JAVA unida a la facilitat de consulta de XQuery lligada a una API d'accés a les dades XML còmoda i intuïtiva. En qüestió de minuts l'estudiant està finalment en condicions de dur a terme el seu projecte.

Aquest procés fins a trobar una solució adient al problema de la consulta i accés a les dades del repositori LOM ha consumit la major part del recursos temporals del projecte.

Exemple d'aplicació pràctica d'XQEngine.

Essent conscient de la desorientació que esta patint l'estudiant el tutor de projecte li encarrega una tasca concreta per que hi centri els seus esforços. És una tasca que tracta de resoldre de manera simplista el problema dels camps buits en tots els seus aspectes. S'espera que sigui una prova de concepte per posteriors ampliacions de la solució donada.

Descripció de la tasca

Partint d'un conjunt d'elements etiquetats amb un estàndard basat en XML. Per exemple:

```
<arrel>
        <element id="elem1">
                <camp1>valor1_1</camp1>
                <camp2>valor2</camp2>
        </element>

        <element id="elem2">
                <camp1>valor1_2</camp1>
                <camp3>valor3</camp3>
        </element>
</arrel>
```

L'objectiu seria generar, a partir d'aquest repositori, la sortida següent:

element_id	camp1	camp2	camp3
elem1	valor1_1	valor2	NULL
elem2	valor1_2	NULL	valor3

és a dir, una fila per cada element, i una columna per cada atribut/camp possible, omplint els inexistents a NULL o qualsevol altre valor que ho identifiqui, amb una primera fila que conté els noms dels atributs/camps. L'ordre de les columnes no importa, ja que la sortida generada conté aquesta primera línia que permet identificar cada valor.

L'estudiant tracta de resoldre el problema amb la única ajuda d'XQuery. La seva primera valoració del problema es que és un problema "senzill" de resoldre amb les eines que ofereix XQuery. Fa falta un cert temps per que l'estudiant sigui conscient de les limitacions d'XQuery, sobretot tenint en compta que no es disposa d'una implementació complerta del llenguatge XQuery que descriuen el llibres.

Arribats a un punt sense sortida l'estudiant decideix emprar la API d'XQEngine. XQEngine és un conjunt de llibreries que indexen i realitzen consultes XQuery sobre documents basats en XML. A més de la interfície de consulta el programador pot accedir a totes les eines internes d'XQEngine per a fabricar les seves pròpies solucions.

La API d'XQEngine està basada en SAX, la llibreria de manipulació de documents XML de JAVA, però n'ofereix un accés amb un grau d'encapsulació molt pràctic.

Finalment la solució passa per recórrer l'arbre del document anotant tots els camps i atributs disponibles per després tornar a recórrer el document generant la sortida

esperada. Es tracta d'una feina a un nivell molt més baix del que oferia XQuery i per tant és una solució menys portable i depenent de la implementació d'XQEngine.

Un cop resolt el problema inicial l'estudiant és convidat a tractar de resoldre el problema incloent-hi subnodes al procés. És a dir donada la següent taula s'ha d'obtindre el resultat citat a continuació:

```
<arrel>
<lom atributlom1="valorlom1">
        <camp1 atribut1="valor1">
                <camp11>
                        <camp111 atribut111="valor111"/>
                        <camp112>
                                valor112
                        </camp112>
                </camp11>
        </camp1>
</lom>
<lom>
        <camp1>
                <camp12>
                        valor12
                </camp12>
                <camp13 atribut13="valor13">
                        valor13b
                </camp13>
        </camp1>
        <camp2 atribut2="valor2"/>
</lom>
<lom>
        <camp1 atribut1="valor1"/>
        <camp2 atribut2="valor2"/>
        <camp3 atribut3="valor3"/>
</lom>
</arrel>
```

Per cada camp, 0 o 1 indica si el té o no, i si no té el valor de l'atribut o el camp es denota amb el valor especial NULL. Els camps contXXX (contingut) apareixen només si es troba contingut no "marcat" entre <camp> i </camp>.

```
atributlom1 camp1 atribut1 camp11 camp111 atribut111 camp112 cont112  camp12 cont12
camp13 atribut13 conti13  camp2 atribut2 camp3 atribut3

valorlom1   1     valor1   1      1       valor111   1       valor112 0      NULL    0
NULL      NULL     0     NULL     0     NULL
NULL        1     NULL     0      0       NULL       0       NULL     1      valor12 1
valor13   valor13b 1     valor2   0     NULL
NULL        1     valor1   0      0       NULL       0       NULL     0      NULL    0
NULL      NULL     1     valor2   1     valor3
```

L'estudiant resol el problema tot fent-hi una lleugera variació per reduir el nombre de columnes a mostrar. Es tracta d'un canvi en el format i no en el contingut de la resposta.

Per cada camp/atribut -1 indica que aquest camp/atribut no apareix al document. NULL indica que el camp hi apareix però buit. Un valor en el camp indica que aquest és el valor del camp/atribut i que el camp/atribut apareix al document. És a dir, la taula anterior quedaria d'aquesta manera:

```
atrib_atributlom1     atrib_atribut1 atrib_atribut111      atrib_atribut13
       atrib_atribut2 atrib_atribut3 camp_camp1     camp_camp11    camp_camp111
       camp_camp112   camp_camp12    camp_camp13    camp_camp2     camp_camp3
valorlom1      valor1 valor111       NULL    NULL    NULL    NULL    NULL    NULL
       valor112       -1     -1     -1      -1
NULL    NULL    NULL    valor13 valor2 NULL    NULL    -1      -1      -1      valor12
       valor13b       NULL    -1
NULL    valor1 NULL    NULL    valor2 valor3 NULL    -1      -1      -1      -1      -1
       NULL    NULL
```

Exemple documentat

Nota: Els fitxers SampApp.java, Elemento.java i ejemplo.xml lligats a aquest llistat s'han lliurat al tutor del projecte juntament amb la memòria i un fitxer de text amb el resultat de l'execució del codi sobre l'exemple.xml format per 7500 documents LOM per facilitar el seguiment del projecte. El resultat s'ha generat en format cvs –texte entre cometes separat per punts i coma- per poder ésser fàcilment importable per les eines d'anàlisi numèric.

```
/* Estructurar un document XML en format tabulat */

import java.io.*;
import java.util.*;
import javax.xml.parsers.*;

import com.fatdog.xmlEngine.ResultList;
import com.fatdog.xmlEngine.*;
import com.fatdog.xmlEngine.XQEngine;
import com.fatdog.xmlEngine.exceptions.*;

import org.xml.sax.XMLReader;

/**
 * Classe SampApp.
 * Basada en el SampApp de XQEngine. Converteix nodes de documents XML a format
 * texte senzill amb camps tabulats.
 * @author Daniel R.C.
 */
public class SampApp {

  /**
   * PRETTYPRINT: S'empra per mostrar la sortida XML de les consultes XQuery.
   * És un parametre que reb un metode emitXML de la classe ResultList
   */
  boolean PRETTYPRINT = true;

  /**
   * Tots els documents que volem consultar s'han d'indexar en primer lloc.
   * Per veure maneres d'indexar documents podem mirar el SampApp que proveu
   * el paquet d'XQEngine.
   */
  String EJEMPLO = "ejemplo.xml"; // Exemple sobre el que treballam.

  /**
   * m_engine: És el motor de consulta a dades. Emprarem instancies d'XQEngine
   * per afegir documents a la Base de Dades i per executar consultes que retornin un
   * ResultList.
   */
  XQEngine m_engine;

  /**
   * Metodo main per a proves. Crida el metode run() on hi ha les
   * cridades a inicialitzacions i el cos de l'exemple.
   *
   * @param args String[]
   * @throws Exception
   */
  public static void main(String[] args) throws Exception {
    new SampApp().run();
  }

  /**
   * query
```

```
 * És un metode que encapsula el proces de consulta.
 *
 * @param query String
 * @throws InvalidQueryException
 * @return ResultList
 */
ResultList query(String query) throws InvalidQueryException {
  ResultList results = m_engine.setQuery(query);
  return results;
}

/**
 * escribeCabecera
 * Aquest metode és trucat despres d'haver fet un primer recorregut de l'arbre
 * i d'haver anotat tots els camps i atributs existents al document -sense
 * repeticio-. Escriu la capçalera amb els camps separats per tabulacions.
 *
 * @param atributos Vector
 * Conte el llistat de tots els atributs de node que
 * podem trobar al document processat. (sense repeticions)
 *
 * @param campos Vector
 * Conte el llistat de tots els camps de node que
 * podem trobar al document processat. (sense repeticions)
 */
public void escribeCabecera(Vector atributos, Vector campos) {
  //Dades de DEBUG
  //System.out.println("---------------CABECERA-----------------");
  for (ListIterator i = atributos.listIterator(); i.hasNext(); ) {
    System.out.print("atrib_" + i.next() + "\t");
  }
  for (ListIterator i = campos.listIterator(); i.hasNext(); ) {
    System.out.print("camp_" + i.next() + "\t");
  }
  System.out.println();
}

/**
 * buscaAtributo
 * Per saber si un node te o no te un atribut determinat feim una cerca de
 * l'atribut damunt els elements atribut d'aquell node.
 *
 * @param elementos Vector
 * És un vector d'instancies de la classe Elemento que conte tots els elements
 * que no són NODE del document.
 *
 * @param idNodo int
 * identificador del node damunt el que fem la cerca.
 *
 * @param atributo String
 * Atribut que estem cercant.
 *
 * @return String
 * Valor de l'atribut si el trobam.
 * NULL si no el trobam.
 */
public String buscaAtributo(Vector elementos, int idNodo, String atributo) {
  Elemento elemento;

  //Dades de DEBUG
  //System.out.println(" Busco el atributo: " + atributo);
  //System.out.println(" del nodo: " + idNodo);
  //System.out.println(" En el vector: " + elementos.toString());

  boolean encontrado = false; // Cert si trobam l'atribut.
  for (ListIterator i = elementos.listIterator(); i.hasNext(); ) {
    elemento = (Elemento) i.next();
    int identificador = elemento.getId();
    //Dades de DEBUG
    //System.out.println("El identificador es:"+identificador);
    //System.out.println("El tipo de elemento es:"+elemento.getTipo());
    if ( (idNodo == identificador) &&
        (elemento.getTipo().equalsIgnoreCase("atributo"))) {
      if (elemento.getNombre().equalsIgnoreCase(atributo)) {
        encontrado = true;
        return elemento.getValor();
      }
```

```
      }
    }
    if (!encontrado) {
      return "NULL";
    }
    //Aixo és per evitar problemes si es passa un vector que no és d'elements
    //Al metode. Aixi almenys es garanteix que hi ha un retorn valid.
    return "";
  }

  /**
   * buscaCampo
   * Aquest metode cerca un camp determinat per un node determinat.
   *
   * @param elementos Vector
   * Vector que conte tots els elements del document. Els elements venen
   * representats per instancies de la classe auxiliar Elemento.
   *
   * @param idNodo int
   * identificador del node sobre el que fem la cerca
   *
   * @param campo String
   * nom del camp que estem cercant.
   *
   * @return String
   * Retorna el valor del camp si el trobam
   * Retorna NULL si el node no posseix aquell camp.
   *
   */
  public String buscaCampo(Vector elementos, int idNodo, String campo) {
    Elemento elemento;

    //Dades de DEBUG
    //System.out.println(" Busco el campo: " + campo);
    //System.out.println(" del nodo: " + idNodo);
    //System.out.println(" En el vector: " + elementos.toString());

    boolean encontrado = false;
    for (ListIterator i = elementos.listIterator(); i.hasNext(); ) {
      elemento = (Elemento) i.next();
      int identificador = elemento.getId();
      //Dades de DEBUG
      //System.out.println("El identificador es:"+identificador);
      //System.out.println("El tipo de elemento es:"+elemento.getTipo());
      if ( (idNodo == identificador) &&
          (elemento.getTipo().equalsIgnoreCase("campo"))) {
        if (elemento.getNombre().equalsIgnoreCase(campo)) {
          encontrado = true;
          return elemento.getValor();
        }
      }

    }
    if (!encontrado) {
      //Aquest és el valor que donam a un camp que no apareix en un node.
      //Si el camp apareix en el node llavors retornam el seu valor.
      return "-1";
    }
    //Aixo és per evitar problemes si es passa un vector que no es d'elements
    //Al metode. Aixi almenys es garanteix que hi ha un retorn valid.
    return "";
  }

  /**
   * escribeFila
   * Aquest metode és cridat per volcar a pantalla les files corresponents als
   * valors de cada node.
   *
   * Primer recorrem tots els atributs i els cercam pel node que estem escrivint.
   * Despres recorrem tots els camps i els cercam pel node que estem escrivint.
   * El resultat és impres en pantalla com a camps separats per tabulacions.
   *
   * @param elementos Vector
   * Vector amb tots els elements del document.
   *
   * @param idNodo int
   * Identificador del node que estem analitzant en aquest moment.
```

```
   *
   * @param atributos Vector
   * Vector amb tots els atributs del document sense repeticions.
   *
   * @param campos Vector
   * Vector amb tots els camps del document sense repeticions.
   */
  public void escribeFila(Vector elementos, int idNodo, Vector atributos,
                          Vector campos) {
    for (ListIterator i = atributos.listIterator(); i.hasNext(); ) {
      System.out.print(buscaAtributo(elementos, idNodo, (String) i.next()));
      System.out.print("\t");
    }
    for (ListIterator i = campos.listIterator(); i.hasNext(); ) {
      System.out.print(buscaCampo(elementos, idNodo, (String) i.next()));
      System.out.print("\t");
    }
    //Cambio de linea para la siguiente fila.
    System.out.println();
  }

  /**
   * run
   * Aqui és on instanciam el motor de cerca XQuery, el configuram i cridam
   * els documents que han de ser analitzats.
   *
   * Aquest metode el proveia l'autor de la llibreria al seu exemple.
   * Només se n'han fet modificacions menors per simplificar-ho i adaptar-ho als
   * requisists de l'exemple.
   *
   * @throws Exception
   */
  public void run() throws Exception {
    m_engine = new XQEngine();

    // no particular reason for setting "2" here; just showing that it can be done.
    m_engine.setMinIndexableWordLength(2);
    m_engine.setShowFileIndexing(true);
    m_engine.setUseLexicalPrefixes(true); // no namespace decls required

    //Consulta per mostrar només el node que tingui un id igual a elem1.
    //Només és per mostrar les capacitats de consulta de l'implementacio
    //d'XQuery de l'XQEngine.
    String query = "//*";

    installSunXMLReader();

    long startTime = System.currentTimeMillis();

    try {
      //Fem el nostre document accesible pel motor de consulta e indexacio.
      //Podem trobar exemples més complexes de com afegir multiples documents al
      //SampApp que proveu el paquet original d'XQEngine.
      m_engine.setDocument(EJEMPLO);
    }
    catch (FileNotFoundException e) {
      System.err.println("\nSampleApp.run(): " + e);
      return;
    }
    catch (CantParseDocumentException e) {
      System.out.println(
          "\n\nSampleApp.run(): CantParseDocumentException: " +
          e.getMessage());
      return;
    }
    catch (MissingOrInvalidSaxParserException e) {
      System.out.println("\nSampleApp.run() exception: " + e.getMessage());
      return;
    }
    long elapsed = System.currentTimeMillis() - startTime;

    //Mostra en pantalla estadistiques respecte del document LOM analitzat.
    //m_engine.printSessionStats(elapsed, null);

    doQuery(query);
  }
```

```
  /**
   * installSunXMLReader
   *
   * Emprat a nivell intern per XQEngine.
   * Podem veure com és truca desde el metode run()
   *
   * Prepara un parser per llegir el document XML que de no ser valid
   * no podria ser processat.
   *
   * @throws Exception
   */
  private void installSunXMLReader() throws Exception {
    SAXParserFactory spf = SAXParserFactory.newInstance();
    try {
      SAXParser parser = spf.newSAXParser();
      XMLReader reader = parser.getXMLReader();

      m_engine.setXMLReader(reader);
    }
    catch (Exception e) {
      throw e;
    }
  }

  /**
   * doQuery
   *
   * Dins aquest metode he dut a terme tot el proces d'analisi del document XML
   * de l'exemple.
   * Aquest metode conte exemples i anotacions respecte del gran part de la API
   * d'XQEngine
   *
   * @param query String
   * El parametre query és una consulta XQuery.
   */
  void doQuery(String query) {
    ResultList results;

    try {
      //El resultat d'una consulta és una mena de recordset tractat dins de la
      //classe ResultList
      results = query(query);
      if (results.getNumValidItems() == 0) {
        System.out.println("\nNo hits!");
        return;
      }
    }
    catch (InvalidQueryException iqe) {
      System.err.println("\n" + iqe);
      System.out.println("InvalidQueryException\n");
      return;
    }
    catch (Exception e) {
      System.out.println(e);
      System.out.println("Oh! Oh! Algo salio mal :-(\n");
      return;
    }

    // Descomentar aquestes linies per veure una sortada amb PRETTYPRINT
    //System.out.println("esto es emitXML con el prety print true");
    //System.out.println(results.emitXml(PRETTYPRINT) + "\n");

    // Descomentar aquestes linees per veure una sortida sense PRETTYPRINT
    //System.out.println("esto es una salida sin prety print");
    //System.out.println(results.toString() + "\n");

    //Per poder treballar amb el document XML a més baix nivell creo un objecte
    //DocItems que ve a correspondre a un document XML
    DocItems doc;

    //Obtenc el document damunt el qual s'ha fet la consulta. En aquest cas
(ejemplo.xml)
    //No era necesari pero volia provar el funcionament dels metodes de results.
    //No se com obtindre un arbre amb el conjunt de més d'un document si no és llegint
un document
    //de varis Mb amb tots els documents en 1. He preparat un script de python i ara
existeix un
```

```
    //unic document de 30Mb que recull els 7500 LOMs disponibles.
    doc = results.getDocumentWithId(0);

    //Descomentar per veure dades que ens permetin entendre que fa cada metode de l'API
    //System.out.println("Inicio del doc");
    //System.out.println(doc.toString());
    //System.out.println("Fin del doc");

    //Per accedir als nodes individuals del document creo un objecte NodeTree
    //amb getTree puc accedir a l'arbre del document XML ja preprocessat i preparat per
ésser
    //treballat.
    NodeTree nodeTree;
    nodeTree = doc.getTree(); //Obtenc l'arbre prepocessat del document ejemplo.xml

    //Descomentar aixo per veure la sortida de nodeTree.toString()
    //És una sortida molt interessant que ens permet veure com treballa XQEngine.
    //Basicament veim una llista indexada d'elements amb un tipus, un seguent node, un
pare i
    //altres dades imprescindibles per recorrer l'arbre facilment.

////////////////////////////////////////////////////////////////////////////
    //System.out.println("Vista completa del arbol generado por XQEngine.");
    //System.out.println(nodeTree.toString());
////////////////////////////////////////////////////////////////////////////

    // Taula de valors dels tipus d'elements del document XML
    // ATRIB : 1
    // TEXTO : 2
    // ELEMENTO: 0

    /**
     * Vector de atributos distintos en el documento.
     */
    Vector atributos = new Vector();

    /**
     * Vector de elementos del documento.
     */
    Vector elementos = new Vector();

    /**
     * Vector de campos distintos en el documento.
     */
    Vector campos = new Vector();

    /**
     * Vector de identificadores de NODOS distintos en el documento.
     */
    Vector ids = new Vector();

    //Declar unes variables auxiliars que empro en els seguents recorreguts
    //de nodes.
    int idNodo = 0;
    String nombre;
    String valor;

    /**
     * El seguent recorregut extreu TOTS els atributs distints del document.
     * Tambe s'aprofita per crear el vector d'elements del document.
     */

    //Recorregut dels nodes de l'arbre del document ejemplo.xml
    //que és sobre el que es fa la consulta.
    for (int i = 0; i < nodeTree.getNodeCount(); i++) {
      if (nodeTree.getParent(i) == 0) {
        //Volem conservar el valor de l'identificar -intern- del NODE
        //per poder lligar tots els elements al seu node PARE.
        //Com que el format de NodeTree ens dona els elements ordenats per NODE
        //un cop detectat un NODE tots els seguents elements són seus fins que
        //es canvia de NODE.
        idNodo = i;
        ids.add(new Integer(idNodo)); //Anotam el node a la llista de nodes del
document.
      }

      // Tipus = 1 -> L'element és un atribut de node. Una metadada en diu
```

```
      // l'estandar XML
      if (nodeTree.getType(i) == 1) {
        //Ens aseguram de no ficar un valor existent al vector d'atributs
        if (!atributos.contains(nodeTree.getAttributeName(i))) {
          atributos.add(nodeTree.getAttributeName(i));
        }
        //Obtenim el nom i el valor de l'atribut
        nombre = nodeTree.getAttributeName(i);
        valor = nodeTree.getAttributeText(i);
        //Cream un element del tipus "atributo" amb un nom, un valor i un codi
        //de node pare.
        Elemento elem = new Elemento(idNodo, "atributo", nombre, valor.trim());
        elementos.add(elem);
      }

      if (nodeTree.getType(i) == 0) {
        //Pare > 0 -> no és un node fill de l'arrel.
        //Sol tractar-se d'un camp. Pero podria ser un altre node.
        if (nodeTree.getParent(i) > 0) {
          //Dades de DEBUG
          //System.out.print(i + ":" + nodeTree.getElementName(i));
          nombre = nodeTree.getElementName(i); //Nom del camp

          //Si el camp no existeix al vector de camps l'afegim
          if (!campos.contains(nodeTree.getElementName(i))) {
            campos.add(nodeTree.getElementName(i));
          }

          //Per algun motiu d'implementacio interna els camps buids
          //tenen una longitud 2 i longitud > 2 implica un camp no buid.
          //NULL és el valor que donarem a un camp que apareix al node pero que
          //esta buid o sense valor.
          valor = "NULL";
          int j = 0; //Variable auxiliar sense més pretensions.

          //En general el valor d'un camp és el seguent element de la llista que
          //genera l'arbre del document.
          //Ara bé, si el camp te atributs, el valor del camp és posterior a aquest
atributs.
          //Per aixo fem el i+2 en lloc del i+1 si el camp te un atribut.
          if (i + 1 < nodeTree.getNodeCount()) {
            if ( (nodeTree.getType(i + 1) == 1) &&
                (i + 2 < nodeTree.getNodeCount())) {
              j = i + 1;
            }
            else {
              j = i;
            }
            if (nodeTree.getType(j + 1) == 2) { //Asegurar-se de que estem consultant un
TEXTE
              if (nodeTree.getElementText(j + 1).length() > 2) {
                //Dades de DEBUG
                //System.out.println(":" + nodeTree.getElementText(i + 1));

                //Fem getElementText(i+1) perque l'implementacio prepara que el
                //Texte d'un camp aparegui com al seguent element de la llista.
                valor = nodeTree.getElementText(j + 1);
              }
              else {
                //Dades de DEBUG
                //System.out.println("Esta buid");
                valor = "NULL"; //Faig que els camps buid prenguin per valor NULL
              }
              //Alguns documents contenen tabuladors dins dels nodes que són considerats
com
              //un valor de texte quan en el fons són etiquetes buides. Es tracta de
corregir
              //aquest problema amb el trim()
              if (valor.trim().length() < 1) {
                valor = "NULL"; //Documents que tenen camps plens de caracters de
                //control són tambe camps buids.
              }
              //Dono d'alta l'element "campo" amb els valors obtinguts anteriorment.
              Elemento elem = new Elemento(idNodo, "campo", nombre, valor.trim());
              //Dades de DEBUG
              //System.out.println(elem.toString());
              elementos.add(elem);
```

```
            }
            else {
              Elemento elem = new Elemento(idNodo, "campo", nombre, valor.trim());
              //Dades de DEBUG
              //System.out.println(elem.toString());
              elementos.add(elem);
            }
          }
        }
      }
    }

    //Dades de DEBUG
    //System.out.println("Estos són los atributos: " + atributos.toString());
    //System.out.println("Estos són los campos: " + campos.toString());
    //System.out.println("Estos son los elementos encontrados: " +
elementos.toString());

    //escribeCabecera(atributos, campos);
    //escribeFila(elementos, 1, atributos, campos);
    //escribeFila(elementos, 16, atributos, campos);

    //System.out.println("Numero de nodos:" + ids.size());

    //Escrivim la capçalera de la taula.
    escribeCabecera(atributos, campos);

    // Variable auxiliar per extreure els codis de NODE del Vector de nodes.
    // He creat un vector d'Integers perque Vector requeria un Object
    // com a parametre i no acceptava un simple int
    Integer id;
    //Ara fem un recorregut de la llista de nodes i de cada node en demanam
    //que se n'escriguin les seves dades en format de fila tabulada.
    for (ListIterator i = ids.listIterator(); i.hasNext(); ) {
      id = (Integer) i.next();
      escribeFila(elementos, id.intValue(), atributos, campos);
    //Dades de DEBUG
    //System.out.println("Fin de esta fila");
    }

    //Dades de DEBUG. Com les altres dades de DEBUG del document les
    //deixo perque consider que són utils per familiaritzar-se rapidament amb
    //la API d'XQEngine.

    //System.out.println(nodeTree.getAttributeName(0));
    //System.out.println(nodeTree.getAttributeName(1));
    //System.out.println(nodeTree.getAttributeText(2));
    //System.out.println(nodeTree.getAttributeText(11));

    //System.out.println(nodeTree.getElementName(1));

    //for (int i=0; i<nodeTree.getNodeCount();i++) {
    //  if (nodeTree.getType(i) == 2) {
    //    System.out.print(i);
    //    System.out.print(" lenght ->" + nodeTree.getElementText(i).length());
    //    if (nodeTree.getElementText(i).length() == 2) {
    //      System.out.println("BUID");
    //    } else {
    //      System.out.println(nodeTree.getElementText(i));
    //    }
    //   }
    //  }

    //System.out.println("Numero de atributos del segundo nodo (se numera desde 0)");
    //System.out.println(nodeTree.numAttributesOnElement(1));

  }
}
```

Classe auxiliar Elemento

```
/**
 * <p>Título: Classe Auxiliar Elemento</p>
 *
 * @author Daniel R.C.
 * @version 1.0
 */
public class Elemento {
  private int idNodo;
  private String tipo;
  private String nombre;
  private String valor;
  /**
   * Elemento
   * Constructor per defecte
   */
  public Elemento() {
  }

  /**
   * Elemento: Constructor parametritzat.
   *
   * @param idNodop int Identificador de node pare.
   * @param tipop String Cadena descriptora del tipus d'element campo/atributo
   * @param nombrep String Nom del campo/atributo
   * @param valorp String valor del campo/atributo
   */
  public Elemento(int idNodop, String tipop, String nombrep, String valorp) {
    tipo = tipop;
    nombre = nombrep;
    valor = valorp;
    idNodo = idNodop;
  }

  /**
   * toString
   * Mostra les dades de l'element en pantalla de forma bastant groera.
   * Metodo existent per motius de DEBUG
   * @return String
   */
  public String toString() {
    String resultado;
    resultado = tipo + "/" + nombre + "/" + valor + "/" + idNodo;
    return resultado;
  }

  /**
   * main
   * Metode main per les proves unitaries de la classe.
   *
   * @param args String[]
   */
  public static void main(String[] args) {
    Elemento elemento1 = new Elemento(12, "tipo", "nombre", "valor");
    System.out.println(elemento1.toString());
  }

  /**
   * getId
   *
   * @return int
   */
  public int getId() {
    return idNodo;
  }

  /**
   * getNombre
   *
   * @return String
   */
  public String getNombre() {
    return nombre;
```

```
  }

  /**
   * getTipo
   *
   * @return String
   */
  public String getTipo() {
    return tipo;
  }

  /**
   * getValor
   *
   * @return String
   */
  public String getValor() {
    return valor;
  }
}
```

Requisits de funcionament del codi d'exemple.

- jdk1.4 http://java.sun.com
- XQEngine 0.63 http://sourceforge.net/projects/xqengine

Les dues classes SampApp.java i Elemento.java així com el fitxer de dades LOM ejemplo.xml han de residir al mateix directori. El conjunt de llibreries d'XQEngine, així com les llibreries de la jdk1.4 de JAVA han de ser afegides al CLASSPATH de l'entorn JAVA del sistema.

La sortida del programa es fa a pantalla i pot ésser redirigida cap a un fitxer amb els operadors de redreçament. Per ex: **java SampApp > fitxer.txt.**

Eines lligades al funcionament el codi d'exemple.

Tot i que XQEngine pot indexar múltiples documents el resultat del procés d'anàlisi no és el desitjat a no ser que la totalitat dels documents LOM siguin fills del mateix node arrel. Això implica que s'ha de generar un nou document amb la concatenació de tots els documents LOM disponibles. Per generar aquest nou document unió de tots els documents disponibles s'han de llegir línia a línia tots els documents i bolcar-los a un nou fitxer de destí.
L'estudiant es troba molt familiaritzat amb Python com a llenguatge de guions per a procés de cadenes i fitxers de text. El resultat és un codi en Python per a realitzar aquesta tasca de fusió necessària per a obtindre els resultats desitjats.

Requisits de funcionament del guió:

- Python 2.2 o sup. http://www.python.org.

Python ha d'estar al PATH del sistema.

Execució del guió:
S'ha de crear un directori continguent la totalitat dels documents LOM que volem fusionar. Copiarem el guió al interior d'aquest directori.

Ara bastarà executar el guió amb un doble clic si ho fem des de un entorn de finestres o amb C:\directiriDeLoms\>python juntaloms.py si ho fem des de línia de comandes.

El fitxer resultant "ejemplo.xml" el trobarem al directori superior al directori sobre el qual hem llençat el guió.

juntaloms.py

```
import os,sys,getopt,re,fileinput

#Creo un fitxer ejemplo.xml en el directori superior
#a aquell en el que llenço l'script.
f = file("../ejemplo.xml","w+")

def procesaLinea(linea,fichero):
    #No processam les linies <? xml ?>
    if linea[:2]!="<?":
```

```
        #No volem atributs dins de l'etiqueta lom
        #Tenim problemes procesant les dades i tenen a veure amb
        #els atributs xlsns de l'etiqueta lom.
        #<lom xlsns:http://www.xx.org/> es converteix en <lom>
        if linea[1:4]=='lom':
                fin=linea.find(">")
                #Afegeixo in idFichero que no existeix per poder
                #detectar errors amb més precissió.
                linea="<lom idFichero='"+fichero+"'"+linea[fin:]
        f.write(linea+"\n")

def imprime(dummy, dir, files):
        #Afegim una arrel comu a tots els documents LOM que
        #estem fusionant.
        f.write("<raiz>\n")
        #Tinc un comptador de fitxers per feedback i per fer proves
        #amb menys fitxers quan fa falta.
        i=0
      for fichero in files:
            #print fichero
            i = i  + 1
            #if i==1000:
            #    break
            #No volem que s'afegeixi l'script als documents LOM!
            if fichero!="juntaloms.py":
                #El motiu principal d'emprar fileinput es que no espera
                #a llegir el fitxer sencer sinó que va buidant-se fila
                #a fila essent una solució ideal per manipular fitxers
                #grans.
                for linea in fileinput.input([fichero]):
                        procesaLinea(linea.strip(),fichero)
        f.write("</raiz>\n")
        print "procesados", i, "LOMS"

#Aquesta ordre visita cada fitxer del directori on s'ha llençat
#l'script i passa el nom del fitxer com parametre a la funcio que
#se li indiqui. En aquest cas "imprime"
os.path.walk("C:\LOMO", imprime, None)
```

Problemes detectats en l'execució de la solució

Problemes respecte del tamany de les dades a manipular.

El document resultant de fusionar la totalitat del repositori LOM de 7500 documents és d'un tamany de 30Mbytes. En voler processar aquest document l'aplicació respon amb un error OutOfMemoryError.
Consultades les fonts disponibles es procedeix a ampliar la memòria assignada a la maquina virtual de JAVA amb l'opció **-Xmx256** i posteriorment amb l'opció **–Xmx512** a una màquina dotada de 1Gbyte de RAM.

Cap de les dues opcions no solventa el problema. El procedeix aleshores a dividir el document en 4 parts d'aproximadament 8Mbyte cadascuna.

Problemes relatius a la lectura de les dades XML.

Errors 0xb5, &#x3c, final de fitxer incorrecte i falta node arrel del document.

Tots aquestos errors fan referència a un mateix problema. Els fitxers codificats amb UTF-16 empren 2 bytes per a cada caràcter. 0xb5 és un byte alt d'un caràcter UTF i &#x3c és el codi del caràcter "< en UTF.

La documentació consultada apunta cap a problemes bé dins el document, bé dins documents que són referenciats des de el document principal. Problemes, tots ells relacionats amb documents XML mal formats.

En un primer intent de solventar el problema s'eliminen les referències a documents externs –fulls d'estil, DTD's, etc.- des de dins els documents LOM. El resultat mostra una petita millora en el nombre d'errors, però no els resol.

En un següent intent s'identifiquen els documents causants dels problemes i se'n realitza un estudi separat dels altres per trobar-ne la causa del problema. Els documents causants dels problemes són perfectament vàlids examinats per separat.

En una lectura del document global es detecta un fet significatiu. Hi ha un gran nombre de documents LOM escrits sense cap retorn de línia. S'afegeixen retorns de línia als documents per fer-los més llegibles.

Curiosament és aquest últim fet el que soluciona tots els problemes. Possiblement l'eina examina els documents línia a línia i en trobar una línia d'un tamany superior als 8Mb havia de tindre problemes per processar-la.

Problemes relatius a la reconstrucció del fitxer d'anàlisi.

A causa de la problemàtica del tamany màxim de fitxer processat s'han de fer 4 execucions i generar 4 fitxers de resultats.

Els 4 fitxers per separat no són prou significatius i és adient dur a terme una fusió dels documents per obtindre la mateixa solució que si haguéssim aplicat els procediment damunt el fitxer "ejemplo.xml" de partida.

Un dels problemes de la fusió és que poden aparèixer camps a un segment dels documents que no existeixen en camp altre segment.

En aquest cas és aconsellable emprar alguna eina d'ajuda com ara MSAccess. La importació de text de MSAccess ens permet generar una taula amb el contingut dels nostres fitxers de text.

Importar el nostre primer document de text és senzill. Quan importem el segon document de text hem de dir que volem importar-ho a una taula existent i ficar les dades del segon document dins de la taula que ens va generar el primer document importat. En aquesta ocasió la aplicació ens informarà de quins camps d'origen no tenen cap imatge a la taula de destí. Anotem aquestos camps i els afegim manualment a la taula que tenim. En mode consulta fem un "UPDATE taula set campnous='-1'" per reflectir que aquestos camps no apareixen als documents LOM previs. Fet això podem continuar amb la importació de dades.

S'ha detectat que els camps amb capitalització són entesos com camps "desconeguts" per part de l'eina d'importació d'MSAccess. Aquestos camps s'han de revisar

manualment i se'ls hi ha de posar el nom de camp que tenien a altres documents que han estat correctament importats.

És important dur a terme aquestes tasques cuidadosament per poder reconstruir el document de forma correcta.

Un cop finalitzada la importació dels 4 segments es pot procedir a una exportació de la base de dades en el format de destí que considerem més adient.

Anàlisi de les dades generades

Per a realitzar un anàlisi de dades s'ha emprat una fulla de càlcul i s'ha importat el document que havíem obtingut als procediments anteriors.

S'ha dut a terme un recompte dels totals per columna. La fulla de càlcul no contabilitza text, de forma que bàsicament s'acumula la xifra -1 per columnes. La suma de valors -1 ens indicarà quants de documents no tenen aquell camp/atribut al seu cos.

Mitjançant l'ús de condicionals s'han creat camps que reflecteixin visualment quins camps són molt freqüents i quins són poc freqüents basant-se en percentatges d'aparició.

Mitjançant l'ús de filtres s'han detectat quantitats d'atributs/camps nuls i en general s'han posat en evidencia les dades que s'havien pogut preveure mitjançant inspecció visual dels documents.

S'han posat d'evidència carències d'aquesta forma d'anàlisi simplificat i descontextualitzat i s'han documentat per a ésser corregides o revisades en futures millores del projecte.

Futures millores del projecte.

Tot i no ésser significatiu de cara a un primer anàlisi la forma actual d'analitzar les dades realitza una simplificació del problema en no reflectir els camps atributs repetits o dades contextuals.

Podem dir que un cert camp apareix només al 50% dels documents. Però no podem parlar sobre el nombre d'aparicions d'aquest camp en aquestos documents.

Per exemple, un 43,7% dels 7462 documents estudiats empra un camp <string/>. Però quantes vegades apareix el camp <string/> a qualcun d'aquestos documents? És un ús puntual o un ús extens al llarg dels documents?

Una consulta d'XQuery *count(//string)* ens servirà per dir-nos que hi ha 95211 aparicions del camp <string/> als 3260 documents en que apareix. És a dir, aquest camp apareix en mitja 29 vegades a cada document.

Un altre problema vinculat a les simplificacions realitzades es deriva d'haver perdut la relació entre nodes, camps i atributs. És a dir, s'han descontextualitzat les dades.

Podem dir que 3256 documents empren l'atribut "language". Ara bé, no sabem en quin context l'empren. De quin camp és atribut?

Una consulta XQuery tal que *//string* pot posar d'evidència que totes les aparicions de l'atribut "language" estan vinculades directament al camp <string/>. Essent `<string language="">text</string>` el contexte en que apareixen tant aquest camp com aquest atribut.

Aquesta primera fase del projecte ens ha permès detectar irregularitats i definir una metodologia de treball que ens permeti aprofundir en l'estudi de les metadades presents als documents LOM. Fases posteriors d'estudi podrien aprofundir en els usos irregulars, la seva importància i els seus contexts.

Conclusions

Camps que apareixen almenys al 99% dels documents (13/70):

camp_general	camp_contribute	camp_minimumversion
camp_title	camp_technical	camp_educational
camp_language	camp_format	camp_datetime
camp_source	camp_size	camp_date
camp_value		

Camps que no apareixen almenys al 50% dels documents (13/70):

atrib_language	camp_tipicalagerange	camp_person
camp_lifeCycle	camp_difficulty	camp_otherPlatformRequeriment
camp_interactivityLevel	camp_intendeduserrole	camp_string
camp_semanticdensity	camp_entity	camp_metadataSchema
		camp_orComposite

En aquesta fase del projecte no hem d'extreure conclusions precipitades. Un fet significatiu és que tot i aparèixer als documents, la majoria de camps no tenen cap valor. Es pot donar el cas d'un camp que aparegui al 99% dels documents però que només tingui valors a un sol document. És el cas del camp "title" que aparentment només té un sol valor assignat.

Si la dada anterior resulta inesperada una consulta XQuery tal que *//title* ens mostraria tots els títols de tots els documents que tenim indexats i ens ajudaria a comprendre què està passant.

```
<title><langstring xml:lang="en">trees2</langstring></title>
```

En estar el títol entre etiquetes <langstring/> es considera que <title/> és un camp buit i que <langstring/> te per valor "trees2".

Les primeres conclusions venen a confirmar el que ens podíem esperar. En ésser LOM un estendard tan ampli, molt pocs usuaris n'arriben a fer un ús exhaustiu. D'altra banda els camps destinats a introduir dades més complexes són menys emprats que els camps destinats a dades generals.

Contribuïdor, títol, data, format o idioma són dades utilitzades a més del 99% dels documents. D'altra banda dificultat, rang d'edats, densitat semàntica o cicle de vida són alguns dels camps menys emprats ja que no apareixen ni al 50% dels documents.

Tot i així, si estudiem algunes dades numèriques respecte a d'utilització de LOM podrem veure dades interessants.

Consulta: count(//source)
106519 aparicions de l'etiqueta source damunt un total de 7500 documents LOM.
14 etiquetes <source/> per document aproximadament.

Això indica que hi ha 14 variacions damunt l'estàndard per a cada document, faltaria saber quines són però tot indica que tenen a veure amb decisions de LOM que no han estat acceptades per la comunitat. Com ara el cost amb valor "NO" per dir "Free" o "Gratuit".

Consulta:
let $lengua:=//language
for $lang in $lengua return
if ($lang='') then 'vacio' else ''

Sobre 10000 etiquetes language 5744 estan buides.
Això representa aproximadament un 57% d'etiquetes buides.

Consulta:
let $lengua:=//langstring
for $lang in $lengua return
if ($lang='') then 'vacio' else ''

Trobades 138439 etiquetes langstring. El 69% de elles estan buides.
En concret 95000 etiquetes langstring buides aproximadament.

Les etiquetes <language/> i <langstring/> han estat triades expressament perquè són potser les més significatives en quant a densitat d'etiquetes buides. Altres etiquetes com <title/> no apareixen buides a cap document, tot i que el primer anàlisi diu tot lo contrari, i d'altres etiquetes menys significatives com ara <cost/> apareixen buides en un 35% dels casos aproximadament.

Aquestes dades, juntament amb les dades obtingudes de l'anàlisi anterior dels camps posen en evidencia que LOM és l'estàndard adequat per al públic equivocat. Els usuaris de LOM es veuen obligats a afegir camps als seus documents malgrat no fer-ne cap ús, o potser a causa de no entendre'n l'ús. D'altra banda almenys un 20% (13/70) de les etiquetes presents als documents LOM tenen un públic molt reduït i no són enteses per contribuïdors menys tècnics. D'altra banda el gran nombre d'etiquetes diferents per representar la mateixa informació (<lanstring/>, <language/>, <string language=""/>) evidencia futurs problemes de portabilitat salvables únicament si les eines de consulta són prou versàtils per la manipulació dels documents XML com ho pot ésser XQuery en aquestos moments.

Imaginem per un moment un mestre d'educació primària que ha preparat un compte infantil amb els seus alumnes i vol publicar-lo a Internet, a un repositori LOM. És factible que aquesta persona pugui redactar un document LOM sense l'ajuda d'eines apropiades a aquest propòsit? no. És factible que aquesta persona pugui comprendre els continguts dels més de 60 camps que ha d'omplir per crear un document LOM? La resposta torna a ser no. Depèn l'èxit dels repositoris d'e-learning de l'aprovació i utilització dels estàndards per part del gran públic i no només de les grans associacions i facultats especialitzades? La resposta és que molt probablement sí.

Aquesta primera fase de l'anàlisi ens du a la conclusió que és necessària una versió reduïda de LOM o la creació d'algun estàndard creat per satisfer al públic en general i afavorir així la creació de més i millors repositoris d'objectes d'aprenentatge. En aquest

sentit hem detectat un 20% de camps que són emprats al 99% dels documents que podrien ser una base per la creació d'aquest estàndard reduït de descripció d'objectes del coneixement.

Pressa de contacte amb XQuery

D'ençà de l'aparició d'Internet els documents etiquetats han cobrat gran importància degut a la seva gran portabilitat entre plataformes.

XML és el llenguatge de metadades que ens permet definir llenguatges etiquetats. En aquest sentit HTML és XML, LOM és XML, SGML és XML i en general qualssevol llenguatge etiquetat d'estructura jeràrquica es pot definir en termes de XML.

Donat el gran nombre de documents que existeixen avui dia en format XML el problema d'accedir-los no ha passat desapercebut.

En un principi, les dades XML eren bolcades a bases de dades relacionals i convertides de nou a format XML en el moment de la seva presentació. Aquesta solució pot ésser interessant en el cas de comptar amb uns documents XML perfectament estructurats i poc extensibles. És freqüent trobar documents XML poc extensibles quan de fet XML és justament tot lo contrari, un llenguatge etiquetat extensible.

Traspassar dades entre una base de dades relacional i un document XML a més de ser ineficient ens obliga a limitar l'extensibilitat del document XML i adaptar-lo a les característiques de les nostres estructures relacionals que res tenen a veure amb l'estructura jeràrquica dels documents XML.

El consorci d'estàndards d'Internet W3C és conscient del problema i crea l'any 2002 una primera versió d'XQuery un llenguatge de consulta pensat especialment pels documents XML.

XQuery te punts en comú amb XSLT, llenguatge de transformació de documents XML que serveix per representar les dades XML en una gran varietat de formats diferents des de HTML fins a PDF, però resulta molt menys críptic i ofereix una certa similitud amb la sintaxi SQL de les consultes sobre bases de dades relacionals el que fa que sigui emprat amb facilitat per totes aquelles persones avesades a SQL.

Tot i ser un llenguatge nou i un estàndard recent tots els grans fabricants de bases de dades ja han donat suport explícit a XQuery. Microsoft SQL Server, Oracle i Informix entre altres suporten XQuery com a eina de consulta damunt els seus servidors de bases de dades natius per XML.

XQuery es basa en un estàndard anterior XPath, que defineix com posicionar-se dins un document XML. Un cop posicionats amb XPath, XQuery pot fer recorreguts, filtrats, comparacions i altres tasques amb relativa facilitat. Alguns tutorials mostren el codi equivalent a una mateixa transformació emprant XQuery o emprant XLST i llavors l'evidencia salta a la vista. Ningú tornarà a fer amb XLST el que ara pot fer amb XQuery.

Un llenguatge de programació específic per XQuery garanteix l'extensibilitat del llenguatge i dota al programador d'una eina completa en el mateix sentit que PL/SQL. L'única diferencia que trobarem entre XQuery i SQL és que a XQuery no trobem instruccions de gestió de la base de dades, únicament instruccions de consulta.

Descartats els productes comercials als que no tenim accés hem de cercar solucions al mon del programari lliure. En aquest sentit hi ha molts de projectes vinculats d'alguna manera a XQuery però es troben majoritàriament en una fase embrionària i no resulten útils**.**

Podem destacar com a projectes de programari lliure lligats directament a XQuery i llestos per ésser usats eXist i XQEngine.

eXist és un projecte molt més madur, més avançat, millor i sense cap mena de dubte és el punt de partida de qualssevol persona que vulgui familiaritzar-se amb XQuery i les bases de dades natives XML. El problema d'eXist és que en ésser un projecte avançat familiaritzar-se amb la seva API de programació JAVA resulta complexa.

XQEngine és un projecte menys avançat, amb menys pretensions, però totalment funcional. En ésser un projecte petit es pot llegir la totalitat del seu codi font i familiaritzar-se amb la seva API de programació JAVA és molt més senzill.

En el moment d'escriure aquestes línies cap dels dos projectes suportava el llenguatge de definició de funcions d'XQuery. Ambdós projectes suporten en tot cas l'extensió del llenguatge XQuery mitjançant classes JAVA.

Com es desprèn d'aquestes línies XQuery és un SQL per documents XML i s'hi poden dur a terme exactament les mateixes consultes. Ara bé, XQuery és molt més extensible i portable que SQL fet que el fa molt més desitjable que el seu predecessor. XQuery no requereix que les dades tinguin un format determinat, basta que siguin documents XML vàlids.

Les limitacions d'XQuery les trobem quan volem realitzar consultes respecte dels camps o els atributs del document i no simplement sobre les dades contingudes dins del document. És en aquestos casos quan s'ha d'emprar una solució mixta basada en XQuery i l'API d'XQEngine.

Valoració de les activitats

Metodologia prevista inicialment

Llenguatge de programació: JAVA 2 (jdk 1.4.x)
Base de dades: MySQL

Requisits inicials:

- Comptar amb un repositori de documents LOM vàlids.

Tasques que es duran a terme:

- Generar una base de dades relacional que reflecteixi l'estàndard LOM:

  La creació de la base de dades es dura a terme manualment. No existeixen actualment eines capaces d'automatitzar la creació de taules a partir de documents DTD amb el grau d'encert que ofereix la creació manual.

- Llegir les dades dels documents LOM del repositori i fer-les fàcilment accessibles a l'aplicació:

  L'extracció de dades XML es fa mitjançant un procediment anomenat *unmarshaling.* Existeixen API's especialitzades en la realització d'aquest procés. Es tracta d'estudiar aquestes API's per adaptar-les al procés d'extracció de dades de documents LOM i veure quina API ens ofereix més flexibilitat i facilitat d'ús.

  En el moment de la planificació no tinc prou dades per definir-me. Tot i que en principi l'API genèrica de JAVA JAXB sembla la més adient.

- Bolcar les dades extretes del document LOM a la base de dades:

  Es tractarà d'implementar funcions d'escriptura a base de dades damunt les classes generades en processos anteriors.

- Repetir el procés de lectura/bolcat de dades per cada document del repositori de forma automatitzada:

  Per dur a terme aquesta automatització emprarem les API's d'accés a directoris ofertes per JAVA.
  Un altre opció a contemplar és emprar l'aplicació JAVA des de línia de comandes i programar un guió de comandes per recórrer els directoris.

- Procedir a l'anàlisi dels documents LOM:

  Mitjançant consultes SQL s'obtindran dades sobre les que basar les estadístiques que hauran de determinar la qualitat de les metadades existents als documents LOM enregistrats.
  En el moment de redacció d'aquest document no he decidit com enfocar aquestes estadístiques.

Programa d'activitats previst inicialment

Planificació del projecte

Nota: Les dades indicades són aquelles en que previsiblement la tasca ha de quedar finalitzada.

Tasca principal: Generar estructura de la base de dades LOM (Inici: 4/10/04 Fi: 12/10/04)

- **Subtasca**: Deduir el diagrama entitat relació del model LOM a partir de la lectura del seu estàndard. (4/10/04)
- **Subtasca**: Traduir model entitat relació a relacional i normalitzar fins a 3ªFN mínim (5/10/04)
- **Subtasca**: Implementar el model relacional damunt un SGBD relacional. (8/10/04)
- Documentar aquesta tasca completada a la memòria. (12/10/04)

Tasca principal: Llegir les dades d'un document LOM i treballar amb elles des de JAVA (Inici 13/10/04 Fi: 01/01/04)

- **Subtasca**: Familiaritzar-se amb l'accés a dades XML des de JAVA (13/10704)
- **Subtasca**: Triar una eina per a dur a terme els procediments d'extracció de dades dels documents LOM(16/10/04)
- **Subtasca**: Definir una classe LOM en JAVA que representi els documents LOM(18/10/04)
- **Subtasca**: Aportar aquestes dades a la memòria(19/10/04)
- **Subtasca**: Crear una instancia de la classe LOM i omplir-la amb les dades d'un document LOM XML(24/10/04)
- **Subtasca**: Aportar aquestes dades a la memòria(25/10/04)
- **Subtasca**: Implementar mètodes de persistència damunt la classe LOM per tal de poder emmagatzemar les dades del document LOM dins una base de dades relacional.(30/10/04)
- **Subtasca**: Aportar aquestes noves dades a la memòria (01/11/04)

Presentar PAC2 amb els resultats obtinguts fins aquest punt. (02/11/04)

Tasca principal: Automatitzar el procediment de bolcat de dades (Inici: 03/11/04 Fi 28/11/04)

- **Subtasca**: Familiaritzar-se amb d'interfície d'accés a directoris de JAVA(04/11/04)
- **Subtasca**: Fer practiques de recorreguts de directoris sota JAVA(07/11/04)
- **Subtasca**: Aportar aquestos resultats a la memòria.(08/11/04)
- **Subtasca**: Fer proves d'automatització del bolcat de dades damunt un conjunt limitat de documents LOM i avaluar-ne els resultats.(15/11/04)
- **Subtasca**: Aportar aquestes dades a la memòria (16/11/04)

Tasca principal: Anàlisi de les dades(Inici 17/11/04 Fi 12/12/04)

- **Subtasca**: Dur a terme el bolcat de tots els documents LOM disponibles.(18/11/04)

- **Subtasca**: Dur a terme un manteniment de la base de dades per agilitzar-ne les consultes.(20/11/04)
- **Subtasca**: Realitzar consultes senzilles i tractar de descobrir possibles indexes que haguessin passat desapercebuts.(22/11/04)
- **Subtasca**: Aportar aquestes dades a la memòria.(23/11/04)
- **Subtasca**: Redactar un joc de consultes tractant de predir els resultats esperats.(27/11/04)
- **Subtasca**: Aportar aquestes dades a la memòria.(28/11/04)
- **Subtasca**: Comparar els resultats esperats amb els obtinguts i avaluar la correcció de les dades obtingudes.(29/11/04)
- **Subtasca**: Aportar aquestes dades a la memòria.(30/11/04)
- **Subtasca**: Amb els resultats del primer joc de proves, redactar un segon joc de proves(5/12/04)
- **Subtasca**: Aportar aquestes dades a la memòria.(6/12/04)
- **Subtasca**: Analitzar les dades de totes les proves realitzades i extreure'n conclusions(11/12/04)
- **Subtasca**: Aportar aquestes conclusions a la memòria.(12/12/04)

Lliurar la PAC3 (13/12/04)

Preparar el lliurament final (Inici 14/12/04 Fi:23/12/04)

Lliurament final (10/01/04)

Activitats, tasques, instruments i tècniques realment emprades

Metodologia

Llenguatge de programació: JAVA, Python
Bases de dades: eXist, XQEngine
Llenguatge de consulta sobre bases de dades: XQuery.
Altres eines: XQEngine API

Pla de treball

Fins al 14 d'Octubre el pla de treball era seguit de forma adequada. A partir del 14 d'Octubre l'estudiant va decidir canviar de metodologia i això va tindre diverses repercussions.

D'una banda totes les tasques citades aquí sota, que inicialment havia de fer l'estudiant de forma artesana, queden realitzades automàticament en haver canviat de metodologia. De fet, aquestes tasques les duen a terme tant eXist com XQEngine com a part de la seva funció de base de dades nativa XML.

- **Subtasca**: Triar una eina per a dur a terme els procediments d'extracció de dades dels documents LOM(16/10/04)
- **Subtasca**: Definir una classe LOM en JAVA que representi els documents LOM(18/10/04)
- **Subtasca**: Aportar aquestes dades a la memòria(19/10/04)

- **Subtasca**: Crear una instancia de la classe LOM i omplir-la amb les dades d'un document LOM XML(24/10/04)
- **Subtasca**: Aportar aquestes dades a la memòria(25/10/04)
- **Subtasca**: Implementar mètodes de persistència damunt la classe LOM per tal de poder emmagatzemar les dades del document LOM dins una base de dades relacional.(30/10/04)

Totes les tasques previstes fins dia 16 de Novembre van quedar anticipades en haver canviat d'eina.

La contrapartida del canvi és que adaptar-se a una nova eina i a una nova forma de treballar va consumir part del temps destinat al projecte. Consultat el tutor de projecte l'estudiant va decidir estudiar en profunditat les noves eines per treure-n'hi el màxim profit i veure com això pot beneficiar el resultat final del projecte.

A dia 13 de Desembre de 2004 s'havia acumulat un cert retràs damunt el pla previst. El canvi d'eina havia estat molt profitós i el coneixement d'XQuery i la seva utilització des de dins de JAVA obrien un camp de proves molt ampli i ple d'expectatives.

No hi ha cap resultat interessant damunt l'anàlisi de les metadades LOM. Però per altra banda, en aquestos moments, respondre qüestions damunt les metadades d'un document LOM o d'un repositori LOM és molt més senzill del que podria haver-ho estat en iniciar-se el projecte.

Aquestes són les tasques que hi havia pendents a dia 13 de Desembre i que suposaven un retard de dues setmanes damunt dels terminis previstos inicialment.

- **Subtasca**: Redactar un joc de consultes tractant de predir els resultats esperats.(27/11/04)
- **Subtasca**: Aportar aquestes dades a la memòria.(28/11/04)
- **Subtasca**: Comparar els resultats esperats amb els obtinguts i avaluar la correcció de les dades obtingudes.(29/11/04)
- **Subtasca**: Aportar aquestes dades a la memòria.(30/11/04)
- **Subtasca**: Amb els resultats del primer joc de proves, redactar un segon joc de proves(5/12/04)
- **Subtasca**: Aportar aquestes dades a la memòria.(6/12/04)
- **Subtasca**: Analitzar les dades de totes les proves realitzades i extreure'n conclusions(11/12/04)

Les previsions per part de l'estudiant a mitjans Desembre eren optimistes. Finalment a dia 5 de Gener l'estudiant aconsegueix generar la resposta esperada per a efectuar l'anàlisi de les dades i emetre les seves conclusions.

Fruit d'aquest anàlisi es detecten problemes derivats de la forma en que s'ha generat l'estudi essent necessari un aprofundiment en els resultats obtinguts i l'estudi de les dades mitjançant vies alternatives d'anàlisi.

Tutor de projecte i estudiant acorden la congelació del projecte en aquest punt. Essent les vies alternatives d'anàlisi reflectides a la memòria com a "Futures millores del projecte".

Valoració d'adequació entre lo previst i lo fet realment

La previsió inicial era dur a terme les tasques d'una forma "lenta però segura". Modelar les taules i omplir-les de dades de forma gairebé artesanal era una tasca llarga, mecànica i poc interessant que sense dubte hagués donat els seus fruits.

L'estudiant però decideix provar una solució alternativa amb esperances de donar una millor solució al problema plantejat. L'estudi d'XQuery resulta finalment més llarg i difícil del que es preveia en un principi. La presa de contacte amb una tecnologia totalment desconeguda ha implicat un esforç elevat i el pla del projecte se'n ha vist afectat.

L'estudi d'XQuery no és resultat d'un caprici per part de l'estudiant sinó més aviat un estudi en busca d'una solució millor a la prevista en primer terme. La solució relacional evidenciava grans problemes en el tractament de les dades i estudiar-ne una alternativa era gairebé un requisit per a l'èxit del projecte.

El problema final no ha estat el fet de no poder realitzar consultes damunt les dades disponibles sinó més aviat l'impossibilitat de dur a terme les consultes realment desitjades.

L'ús conjunt d'XQuery i JAVA resol en gran part les limitacions inicials en quant a consultes complexes, i permet a l'estudiant completar una primera fase d'anàlisi. Durant aquest anàlisi XQuery esdevé una eina fonamental en el descobriment de dades que l'anàlisi dut a terme no pot reflectir.

Finalment l'estudiant ha aconseguit dur a terme la tasca principal del projecte i a més ha treballat amb eines que el doten d'una gran capacitat d'anàlisi damunt les dades disponibles. La valoració per part de l'estudiant entre previst i fet és molt bona.

Resultats obtinguts en termes d'activitat professional.

És a dir, què ha produït l'activitat realitzada.

El resultat entregable consta d'un programa JAVA escrit amb l'API d'XQEngine que navega una arbre de nodes XML per generar-ne una sortida tabulada.

Es tracta d'una eina d'anàlisi que permet identificar atributs/camps dels documents i la seva densitat així com l'ús real que es fa d'ells, essent d'especial interès el fet que els camps buits queden reflectits.

D'altra banda s'ha dut a terme una tasca d'anàlisi damunt la tecnologia XQuery tant a nivell d'ús dins del projecte com a nivell d'anàlisi d'eines del mercat i les seves opcions. Aquesta feina ha estat gairebé de consultoria tècnica i és d'esperar que sigui aprofitada per altres projectes i professionals/estudiants relacionats amb els documents XML.

Resultats obtinguts per part de l'alumne.

És a dir, en quin grau s'ha assimilat la professionalitat

Es considera adquirit un alt grau de professionalitat en tant que l'estudiant ha aprofundit en els coneixements d'estàndards d'e-learning, s'ha relacionat amb el mon que envolta aquesta sector, ha estudiat noves tecnologies i eines per donar millors solucions als problemes que se li plantejaven i en general ha respectat les indicacions del tutor fent un correcte seguiment del projecte i abastant la majoria dels objectius fixats inicialment.

Valoració de les competències

Anàlisi i valoració del grau d'adquisició i pràctica al que s'ha arribat en les competències coneixements i habilitats previstes en el programa d'activitats.

El TFC havia de procurar una bona base de coneixements damunt l'estàndard LOM. En aquest sentit el grau de coneixement adquirit és molt satisfactori.

No només s'ha conegut i s'ha comprès LOM dins el seu entorn sinó que a més s'ha adquirit una visió global de la problemàtica de la descripció d'objectes de coneixement i les alternatives a LOM.

S'ha adquirit constància de la situació actual del mon de l'e-learning pel que fa a la compartició d'objectes de coneixement i s'ha constatat que no hi ha un únic i gran repositori d'objectes de coneixement sinó petits projectes no sempre oberts a la col·laboració externa.

S'ha notat també la presència de projectes poc actius. Tot i els avantatges de la web semàntica, orientada a continguts, sembla que aquesta no arriba a calar en la comunitat internacional que gira en torn a la web convencional, orientada a presentació, tot i les seves carències.

L'estudiant considera haver adquirit les competències i habilitats previstes en el programa d'activitats.

Possible adquisició d'altres competències i habilitats no previstes inicialment i el seu grau de relació amb el rol professional.

L'aprenentatge d'Xquery i la pressa de contacte amb les bases de dades natives per XML no estava prevista en un principi i resulten ser competències implicades molt directament amb el projecte que doten l'estudiant d'unes habilitats que no estaven previstes inicialment.

L'aplicació d'aquestos nous coneixements a qualssevol àmbit professional lligat al mon dels documents XML te un gran potencial.

En el temps d'estudi del projecte l'estudiant ha adquirit un "know how", coneixement pràctic, que el capacita per a dur a terme tasques per les que no estava preparat en iniciar el projecte.

Valoració de les pràctiques

En iniciar el projecte l'estudiant no tenia constància de la realitat de l'e-learning i la seva aplicació en diferents projectes oberts i de col·laboració a nivell internacional.

LOM era el nom d'un estàndard amb el que no hi tenia cap contacte i que semblava pertànyer a una altra dimensió totalment allunyada de la nostra realitat.

El fet d'haver de dur a terme una investigació oberta damunt LOM ha permès conèixer tot el que hi ha envoltant LOM i constatar que hi ha una gran comunitat de persones i entitats realment implicades en l'e-learning.

Abans de començar aquest projecte pensava que l'e-learning era la UOC. Pensava en l'e-learning com en un mitjà de comunicació, un campus virtual i poca cosa més.

LOM fa prendre consciència de la importància dels estàndards oberts en una societat que persegueix el coneixement global i on la difusió del coneixement es una peça critica d'aquest procés.

Respecte als coneixements tècnics adquirits gracies al projecte cal recalcar que mai havia aprofundit com fins ara en l'estudi i anàlisi de documents XML. És aquest projecte el que ha fet descobrir la importància de bases de dades natives per XML i la necessitat d'un llenguatge com XQuery que ofereix una eina de consulta nativa per a documents XML capaç de rivalitzar amb el clàssic SQL.

L'aprenentatge d'XQuery no només aporta coneixement damunt el mon de les bases de dades natives XML sinó que al mateix temps aporta una forta base de coneixements sobre l'estructura dels documents XML en general.

El projecte aporta al consumidor d'estàndards la possibilitat de veure'ls sota una altra perspectiva. La perspectiva de l'anàlisi i la discussió en termes científics.

Les aportacions del projecte són molt positives i abundants, l'estudiant es mostra molt satisfet amb el resultat de les pràctiques dutes a terme al llarg del projecte.

Comentaris

El projecte dota a l'estudiant d'una gran llibertat per a dur a terme el projecte. L'estudiant és convidat a estudiar, investigar, descobrir, dispersar-se i finalment concentrar-se en la resolució del tema del projecte.

Un dels perills d'aquesta llibertat és que l'estudiant es pot deixar portar per les ànsies de coneixement i oblidar l'objectiu final del projecte.

Al llarg d'aquest projecte s'ha posat de manifest que si bé són importants les bases de dades natives XML, aquestes no eren necessàries per resoldre el problema que hagués pogut ésser resolt des de bon principi emprant les eines habituals de procés de documents XML.

Ara bé, arribats a aquest punt l'estudiant hagués estat limitat pels seus coneixements i no hagués pogut imaginar noves formes d'anàlisi sobre les mateixes dades.

L'estudi de bases de dades natives XML, del llenguatge XQuery i de les eines de procés de documents XML clàssiques han dotat a l'estudiant d'un ample ventall d'eines amb les que poder abordar l'anàlisi de les dades des de noves perspectives.

És cert que l'estudi de les noves eines ha conduït a problemes amb la temporització prevista, però el resultat final és molt més interessant que el previst inicialment.

Bibliografia / Referències

Bibliografia

Llibres:

- [1] **Java & XML** – Brett Mclaughlin – O'reilly.
- [2] **Java & XML Data Binding** – Brett McLaughlin – O'reilly.
- [3] **JDO Java Data Objects** – Robin M. Roos – Ad. Weasley.
- [4] **Curso de Java** – Patrick Niemeyer, Jonathan Knudsen – O'reilly.
- [5] **XQuery Quick Start** - James McGovern, Per Bothner, Kurt Cagle, James Linn, Vaidyanathan Nagarajan – SAMS.
- [6] **XQuery**: **The XML Query language** - Michael Brundage – Addison Wesley.
- [7] **Xquery from the experts**: Howard Katz Editor, Don Chamberlin, Denise Draper, Mary Fernández, Michael Kay, Jonathan Robie, Michael Rys, Jérôme Siméon, Jim Tivy, Philip Wadler - Addison Wesley.
- [8] **XQuery Tutorial** - Peter Fankhauser, Fraunhofer http://ipsi.fhg.de .

Referències

[9] **LOM Final draft** (15 Juliol 2002).
http://ltsc.ieee.org/wg12/par1484-12-1.html

[10] **LOM tools** – API JAVA genèric per extreure dades LOM o generar documents LOM.
http://edusource.athabascau.ca/LOM-tools.html

[11] **XML/Database** – Recull d'enllaços tècnics en referència als documents XML i la seva relació amb les bases de dades.
http://www.rpbourret.com/xml/XMLDBLinks.htm

A destacar dins d'aquestos enllaços:

- Mapping Objects To Relational Databases (PDF) by Scott W. Ambler.
- Storing XML in Relational Databases by Igor Dayen.
- A General Technique for Querying XML Documents using a Relational Database System (PDF) by Shanmugasundaram, Shekita, Kiernan, Krishnamurthy, Viglas, Naughton, and Tatarinov.
- Berkeley DB XML: An Embedded XML Database by Paul Ford
- What is XQuery? by Per Bothner.
- Introduction to XQuery by Weiqi Gao.
- Persistent DOM: An architecture for XML repositories in relational databases (PDF) by Richard Edwards and Sian Hope.
- XTABLES: Bridging Relational Technology and XML (PDF) by J. Funderburk, G. Kiernan, J. Shanmugasundaram, E. Shekita, C. Wei.
- SQL/XML Working draft (ISO September, 2003).

[12] **Generating SQL statements from DTDs and XML documents –** Técniques i eines per migrar documents XML a bases de dades relacionals.
http://www-106.ibm.com/developerworks/xml/library/x-matters12.html

[13] **LOM-Editor, Technische Universität Darmstadt (2001) –** Editor de documents LOM implementat en JAVA amb documentacio i codi font.
http://www.multibook.de/lom/en/index.html

[14] **JAVA** – Documentació en línia damunt tots els detalls referents a l'API de JAVA.
http://java.sun.com

[15] **D'Arcy Norman @ The learning commons –** Bloc de Norman d'Arcy on entre d'altres coses hi vaig trobar informació sobre LOM i les seves "aparents" carències i sobretot és un enllaç important perquè alla vaig contactar amb en Norman d'Arcy que amb 2 missatges molt curts i directes m'ha donat l'ajuda que havia de menester en aquell moment.
http://commons.ucalgary.ca/weblogs/dnorman/

[16] **eXist: Open Source Native XML database –** Web del projecte eXist. Aquest projecte implementa una base de dades nativa per documents XML amb totes les seves eines. Aporta a més extensions al llenguatge i una perfecta integració amb JAVA.
http://exist.sourceforge.net

[17] **Qexo The GNU kawa implementation of XQuery**. – Kawa és un compilador d'altres llenguatges cap a JAVA. Qexo és una implementació que permet generar classes java i aplicacions JAVA a partir de consultes XQuery.
http://www.**qexo**.org

[18] **IMS Best practice guide** – Guia per a transformar documents IMS en documents LOM de forma efectiva. Donen indicacions sobre com interpretar LOM en alguns casos.
http://www.imsglobal.org/metadata/mdv1p3pd/imsmd_bestv1p3pd.html

[19] **Editor on-line de documents LOM** – És un repositori en linea on els contribuïdors poden crear el seu propi document LOM en fer una contribució. Els seus formularis resulten interessants per quan representen un document LOM genèric i tancat.
http://orworld.upb.de:8081/Lomweb/

[20] **Learning about Learning objects** – Portal dedicat als objectes de coneixement amb informació recent sobre el que esta passant al mon dels learning objects.
http://www.learning-objects.net/modules.php?name=News&new_topic=4

[21] **Learning technology newsletter – Issue 5**. – És una revista tècnica amb continguts on-line. La qualitat dels seus articles i articulistes és excel·lent i és per mi una font d'inspiració.
http://lttf.ieee.org/learn_tech/issues/january2003/index.html#6

[22] **Transformation of ARIADNE XML instances into instances to LOM –** És la font del repositori de dades LOM d'ARIADNE que emprem en aquest anàlisi.
http://rubens.cs.kuleuven.ac.be:8989/ariadne/

[23] **XQuery Reference –** Pagina de tutorials, referències i exemples sobre XQuery, XPath i altres tecnologies lligades a la manipulació de dades XML.
http://www.w3schools.com/xquery/xquery_reference.asp
[24] **XQEngine –** Pagina oficial del projecte XQEngine, implementació JAVA de bases de dades natives XML inclòs suport per XQuery, amb documentació i exemples.
http://xqengine.sourceforge.net/apiTour.html

[25] **An Introduction to XML:DB API –** Article que explica en que consisteix XML:DB en dona referències i en proposa algun exemple.
http://www.xml.com/pub/a/2002/01/09/xmldb_api.html

Referències per a la redacció de la memòria

[26] Sachetto, V. *Las Comunicacions en la Ingeniería: Guía para la redacción de informes.* 2002. Universidad Tecnológica Nacional Facultad Regional La Plata.

[27] Cardona, Salvador, Jordi, Lluïsa. *Presentació d'informes científics i tècnics.* Departament Enginyeria mecànica. 2000. Universitat Politècnica de Catalunya.

[28] Varios. Reglamento para los Proyectos Fin de Carrera de la Titulación de Ingeniero Informático. 2002. UNED.

[29] Varios. Indicacions per a la redacció de la memòria. 2002. UOC.

ANEXES I

Estructura relacional d'un document LOM

LOM: idLom
General.description: fk_idLom, language, description, source
General.language: fk_idLom, language, source
General.keyword: fk_idLom, language, keyword, source
General.structure: fk_idLom, structure, source
General.agregationLevel: fk_idLom, agregationLevel, source
General.coverage: fk_idLom, language, coverage, source
General.catalogIdentifier: fk_idLom, language, idCatalog, source
General.entryIdentifier: fk_idLom, idCatalog, value, source
General.title: fk_idLom, language, title, source

LifeCycle.version: fk_idLom, version, language, source
LifeCycle.status: fk_idLom, status, language, source
LifeCycle.Contribute.date: fk_idLom, date, source
LifeCycle.Contribute.role: fk_idLom, role, language, source
LifeCycle.Contribute.entity: fk_idLom, entity, order, language, source

MetaData.identifier: fk_idLom, catalog, entry, source
MetaData.language: fk_idLom, language, source
MetaData.Contribute.role: fk_idLom, role, language, source
MetaData.Contribute.entity: fk_idLom, entity, order, language, source
MetaData.Contribute.date: fk_idLom, date, source
MetaData.schema: fk_idLom, schema, source

Technical.format: fk_idLom, format, language, source
Techinical.size: fk_idLom, size, language, source
Technical.location: fk_idLom, location, language, source
Technical.installationRemarks: fk_idLom, remarks, language, source
Technical.Requeriment.OrComposite.type: fk_idLom, idComposite, type, language, source
Technical.Requeriment.OrComposite.name: fk_idLom, idComposite, name, language, source
Technical.Requeriment.OrComposite.minVersion: fk_idLom, idComposite, minVersion, language, source
Technical.Requeriment.OrComposite.maxVersion: fk_idLom, idComposite, maxVersion, language, source
Technical.otherPlatformRequeriments: fk_idLom, requeriments, language, source
Technical.duration: fk_idLom, duration, language, source

Educational.interactivityLevel: fk_idLom, interactivityLevel, language, source
Educational.interactivityType: fk_idLom, interactivityType, language, source
Educational.learningResourceType: fk_idLom, learningResourceType, order, language, source
Educational.semanticDensity: fk_idLom, semanticDensity, language, source
Educational.intendedEndUserRole: fk_idLom, intendedEndUser, language, source
Educational.context: fk_idLom, context, language, source

Educational.typicalAgeRange: fk_idLom, typicalAgeRange, language, source
Educational.difficulty: fk_idLom, difficulty, language, source
Educational.typicalLearningTime: fk_idLom, typicalLearningTime, language, source
Educational.description: fk_idLom, description, language, source
Educational.language: fk_idLom, languageValue, language, source

Right.cost: fk_idLom, cost, language, source
Right.copyright: fk_idLom, copyright, language, source
Right.description: fk_idLom, description, language, source

Relation.kind: fk_idLom, kind, language, source
Relation.Resource.identifier: fk_idLom, idIdentifier
Relation.Resource.Identifier.catalog: fk_idLom, fk_idIdentifier, catalog, language, source
Relation.Resource.Identifier.description: fk_idLom, fk_idIdentifier, description, language, source
Relation.Resource.Identifier.entry: fk_idLom, fk_idIdentifier, entry, language, source

Annotation.entity: fk_idLom, entity, language, source
Annotation.date: fk_idLom, date, language, source
Annotation.description: fk_idLom, description, language, source

Classification.purpose: fk_idLom, purpose, language, source
Classification.description: fk_idLom, description, language, source
Classification.keyword: fk_idLom, keyword, order, language, source
Classification.TaxonPath.source: fk_idLom, idPath, sourceValue, language, source
Classification.TaxonPath.taxon: fk_idLom, fk_idPath, idTaxon, taxon, order, language, source
Classification.TaxonPath.Taxon.id: fk_idLom, fk_idPath, fk_idTaxon, id, language, source
Classification.TaxonPath.Taxon.entry: fk_idLom, fk_idPath, fk_idTaxon, entry, language, source

ANEXES II

S'aporta copia dels missatges intercanviats amb Norman d'Arcy per que quedi constància dels termes d'ús als que va sotmetre les dades que va enviar.

```
On Oct 14, 2004, at 4:41 PM, Daniel R.C. wrote:

> Hi,
>
> My name is Daniel Rodriguez from Spain.
>
> I'm a last year student in Computer Science and I do my project Yes,
> you are rigth ;-) my project is about LOM To do my work I need to
get
> samples from people using LOM. I try to do statistiques about LOM
> usage.
> In fact, I writing to you because I need your help. I need LOM/XML
> documents and I can't find them in the Net.
> Please, can you give me some of your LOM/XML documents?

Daniel, I can't just distribute the raw source of our metadata - they
were contributed by many users, and they haven't approved
distribution.

If it is just for research purposes, I could provide you with a copy
of all of our records, under the condition that they not be added to
another repository.

Let me know if that's ok...

> PS: My work consist in bind LOM's into relational database -is only
to
> use SQL I think it is better than XQuery/XPath- to do an statistic
about
> quality of lom's.

I'm sure an XML database like eXist or XStreamDB would be much better
suited, and give you the full XQuery language to play with...

- D'Arcy Norman
      Software Developer
      Learning Commons, The University of Calgary
      dlnorman@ucalgary.ca
      http://careo.ucalgary.ca
      http://commons.ucalgary.ca/weblogs/dnorman
      (403) 220-2504
      AIM/iChatAV: dnorman@mac.com
```

Apèndix: Sintaxi bàsica d'Xquery.

A continuació s'exposen alguns detalls de la sintaxi d'Xquery que poden ésser útils per interpretar correctament les consultes d'exemple que s'inclouen en el següent Apèndix.

Sigui el següent document xml.

```
<root>
        <element>
                <atribut metadata="metadata1">atrib1</atribut>
        </element>
        <element>
                <atribut metadata="metadata2">atrib2</atribut>
        </element>
</root>
```

Assignació de valors a variables:

```
1. let $a:=3
2. let $a:=/root/element/atribut
3. let $a:=/root/element/atribut/text()
4. let $a:=/root/element/atribut/@metadata
```

1. Assigna 3 a la variable $a
2. Assigna la llista d'atributs "atribut" a la variable $a. En no especificar cap atribut concret el resultat són tots els nodes atribut del document.
3. Assigna la llista de valors dels atributs "atribut" a la variable $a. text() és una funció que retorna el valor dels camps, el valor per defecte és la funció value() que retorna el nom del camp però no el seu valor.
4. Assigna la llista de metadatas "metadata" a la variable $a. La @ s'empra per fer referència als camps de metadades dins de l'estructura.

Iteració damunt els valors d'una llista.

```
for $item in $llista return $item
```

Iteració de valors filtrats per una condició.

```
for $item in $llista where $item='condicio' return $item
```

Estructura condicional.

```
if (condicio) then 'si cert' else 'si fals'
```

Concatenació de text i variables en una comanda de retorn.

```
for $item in $llista where $item='condicio'
     return Aquest item compleix la condicio: {$item}
```

Accedir a elements o nodes de forma individual.

Podem accedir als nodes de forma individual amb `$item[posicio]`

Iteració damunt un rang de valors determinat.

```
for $i in (1 to 10) return $i
```

On ampliar coneixements:

És convenient ampliar coneixements damunt la manera en que Xquery accedeix i navega pels nodes del document (XQuery empra la navegació definida per l'estàndard XPath) i ampliar coneixements damunt les funcions integrades a Xquery.

Per aprofundir coneixements tot quedant en una introducció rapida la millor recomanació és el tutorial d'Xquery que podrem trobar a la bibliografia. En concret: http://www.w3schools.com/xquery/xquery_reference.asp

Apèndix: Conèixer XQuery amb exemples.

- Algunes de les següents consultes han estat realitzades damunt documents LOM, d'altres s'han realitzat damunt documents XML de prova d'una complexitat molt reduïda. Es tractava d'adquirir control damunt l'eina de consulta no d'iniciar cap anàlisi de dades.
- Les consultes han estat provades amb la base de dades nativa XML eXist i el seu client d'accés a dades. Els resultats damunt XQEngine són similars.

Objectiu de la consulta

Mostrar etiquetes language d'un document que tinguin per valor "nl".

Consulta XQuery

```
let $lenguas:=//language

for $lang in $lenguas where $lang='nl'

    return <lengua>{$lang}</lengua>
```

Explicació de la consulta

La doble barra // és un comodí que significa "qualssevol node per sota de l'arrel".

Estem cercant qualssevol node language per sota de l'arrel i guardem aquesta llista de nodes a la variable $lenguas

Fem un recorregut damunt els nodes continguts a $lenguas que tinguin per valor 'nl'

Fiquem el resultat entre les nostres pròpies etiquetes <lengua>

Resultat

```
<lengua>

   <language>

      <exist:match xmlns:exist="http://exist.sourceforge.net/NS/exist">nl</exist:match>

   </language>

</lengua>

<lengua>

   <language>

      <exist:match xmlns:exist="http://exist.sourceforge.net/NS/exist">nl</exist:match>

   </language>

</lengua>
```

Notar que **<exist:match xmlns:exist="http://exist.sourceforge.net/NS/exist">** no és part del document sinó que és una resposta introduïda per motor de consulta.

Objectiu de la consulta

Mostrar etiquetes language d'un document que continguin una jota 'j'

Consulta XQuery

```
let $lenguas:=//language

for $lang in $lenguas[contains(.,'j')] return <lengua>{$lang}</lengua>
```

Explicació de la consulta

La consulta es similar a l'anterior. Prestam atencio a dos novetats:

1. Utilització de la funció XQuery contains(node,valor) –on el punt fa referència al node actual

2. Enlloc de where s'han emprat [ ] Els corchets permeten especificar condicions damunt un node sense necessitat de la clàusula where.

Resultat

El resultat és buit per que no hi ha cap llengua amb una 'j' al document.

Objectiu de la consulta

Obtindre el node <general> d'un document LOM al que l'etiqueta language tingui per valor 'nl'

Consulta XQuery

```
let $lenguas:=//general[language='nl']

for $lang in $lenguas return $lang
```

Explicació de la consulta

Cercar dins el document XML LOM un node <general> que contengui una etiqueta language amb valor igual a 'nl'.

Resultat

```
<general>

    <identifier>

        <catalog>ARIADNE</catalog>

        <entry>BLKLKP61</entry>

    </identifier>

    <description>

        <string language="nl">Lees aandachtig de opgave, zodat de oplossing die je indient overeenkomt met het gevraagde!</string>

    </description>

    <language>

        <exist:match xmlns:exist="http://exist.sourceforge.net/NS/exist">nl</exist:match>

    </language>

    <aggregationLevel/>

    <aggregationLevel>

        <source>LOMv1.0</source>
```

```
        <value>2</value>
      </aggregationLevel>
      <title>
        <string language="nl">Opgave Practicum 2 Informatie- en ProgrammaStructuren 1997-98 (KULAK)</string>
      </title>
    </general>
```

Notar de nou que el terme string no existeix al document original sinó que és un afegit del motor de consulta **<string language="nl"> .**

Objectiu de la consulta

Omplir etiquetes buides amb el texte “vacio” y les plenes amb el texte “lleno”

Consulta XQuery

```
let $lenguas:=if (empty(//language)) then 'vacio' else 'lleno'

for $lang in $lenguas return <lengua>{$lang}</lengua>
```

Explicació de la consulta

Aquesta consulta inicial no fa el que s’espera d’ella.

La idea inicial era examinar cada node language del document i canviar el text a ‘vacio’/’lleno’ segons si el node era o no buit.

El que fa realment la consulta es dir-nos que la consulta //language no retorna buit sinó que hi ha nodes language i per tant el resultat és un simple “lleno”.

Resultat

```
<lengua>lleno</lengua>
```

Resultat d’executar //language

```
<language/>
<language/>
<language/>
<language>nl</language>
<language>nl</language>
```

Hom podria creure que la solució passa per la següent consulta, en la que el test empty() es fa sobre cada node individual. Però vorem que no s’obté el resultat esperat.

```
let $lengua:=//language

for $lang in $lengua return

if (empty($lang)) then 'vacio' else 'lleno'
```

Pero el resultat, contra pronòstic, és el següent:

lleno

lleno

lleno

lleno

lleno

Curiosament la seguent consulta si que dona la resposta esperada. No s'utilitza empty() sino una comparacio amb l'string buid.

```
let $lengua:=//language

for $lang in $lengua return

if ($lang='') then 'vacio' else 'lleno'
```

Resultat:

vacio

vacio

vacio

lleno

lleno

Objectiu de la consulta

Aprofundir en el coneixement de la funció empty() i respondre la pregunta: Títol dels documents que tinguin el camp language de la secció general no buit.

Consulta XQuery

```
for $b in /lom

where not(empty($b/general/language))

return $b/general/title
```

Explicació de la consulta

La novetat és not() que actua com un negador corrent.

El resultat de la consulta de nou segueix sense ésser l'esperat. Apareix el títol Beach que no te assignat cap valor a l'etiqueta language de la secció general.

Resultat

```
<title>

   <langstring xml:lang="">Beach</langstring>

</title>

<title>

   <string language="nl">Opgave Practicum 2 Informatie- en ProgrammaStructuren 1997-98 (KULAK)</string>

</title>
```

Objectiu de la consulta

Aprofundir en d'utilització de la funció count() i tractar de contar els elements buits d'un document LOM.

Consulta XQuery

```
for $b in /lom

where $b/general/language=''

return count($b/general/title)
```

Explicació de la consulta

Aquesta versió modificada de la consulta anterior troba els elements buits de forma adequada. Ara bé, a l'hora de comptar-los count() no respon com voldríem.

De fet la resposta del count() és correcta. Estem aplicant count() damunt un node individual que canvia a mesura que canvia l'iterador. Però com que aquesta és l'única forma que hem trobat de distingir els elements buits no se'ns acut cap forma de comptar elements buits.

Resultat

```
1

1

1

1

1

1
```

Objectiu de la consulta

En aquesta ocasió tractem d'accedir al contingut d'una etiqueta i a les seves metadades.

En aquest exemple emprem un document XML books.xml de prova.

Consulta XQuery

```
for $book in /BOOKS/BOOK

return <libro>{ $book/@YEAR, $book/TITLE/text()}</libro>
```

Explicació de la consulta

La @ ens permet adreçar-nos a les metadades, a les dades que són atributs d'una etiqueta.

La funció **text()** ens permet referir-nos al valor contingut en l'etiqueta enlloc de referir-nos al nom de la etiqueta.

Resultat

```
<libro YEAR="1999 2003">Data on the Web</libro>

<libro YEAR="2002">XML in Scotland</libro>
```

Objectiu de la consulta

Accedir de forma concreta al valor d'una metadada. Utilització de la funció data(), position() i last().

Consulta XQuery

```
for $elemento in /ejemplo/elemento return

<em>{data($elemento/@id)[position()=last()]}</em>
```

Explicació de la consulta

Aquesta consulta forma part d'una sèrie de consultes que tenien per propòsit resoldre el problema de convertir un document XML en un document tabulat amb unes marques especials pels camps buits o no descrits.

Per a cada node "elemento" del document "ejemplo" es tracta d'accedir al valor de la seva metadada "id" i mostrar-ne el valor.

Position()=last() és una forma de posicionar-se damunt un node del document XML.

Resultat

```
<em>elem1</em>

<em>elem2</em>
```

Objectiu de la consulta

L'objectiu d'aquest exemple és mostrar el contingut de tots els fills segons del node arrel.

Consulta XQuery

```
for $x in /ejemplo//* return $x/text()
```

Explicació de la consulta

Es fa un recorregut damunt els nodes dos nivells per sota de "ejemplo" i s'en treu el valor amb text().

Resultat

```
valor1_1

valor2

valor1_2

valor3
```

Objectiu de la consulta

Aquest exemple mostra com accedir als elements de forma indexada.

Consulta XQuery

```
let $p:=/BOOKS/BOOK

let $n:=count($p)

return $p[$n]
```

Explicació de la consulta

Es treuen tots els nodes BOOK. Contem quants de nodes hi ha hagut i mostrar el darrer dels nodes.

Resultat

```
<BOOK YEAR="2002">

   <AUTHOR>Buneman</AUTHOR>

   <TITLE>XML in Scotland</TITLE>

   <REVIEW>

      <EM>The <EM>best</EM> ever!</EM>

   </REVIEW>

</BOOK>
```

Objectiu de la consulta

Aquest exemple aprofita l'exemple anterior i va un pas endavant enumerant els llibres.

Consulta XQuery

```
let $p:=/BOOKS/BOOK

let $n:=count($p)

for $i in (1 to $n)

return <libro n="{$i}">{$p[$i]}</libro>
```

Explicació de la consulta

Obtenim els nodes BOOK. Comptem el numero de nodes. Creem una tupla de valors per recórrer-la. Recorrem la tupla del nombre de nodes i afegim a cada node nova informació sobre el numero de node.

Resultat

```
<libro n="1">

   <BOOK YEAR="1999 2003">

      <AUTHOR>Abiteboul</AUTHOR>

      <AUTHOR>Buneman</AUTHOR>

      <AUTHOR>Suciu</AUTHOR>

      <TITLE>Data on the Web</TITLE>

      <REVIEW>A <EM>fine</EM> book.</REVIEW>

   </BOOK>
```

```
</libro>

<libro n="2">

   <BOOK YEAR="2002">

      <AUTHOR>Buneman</AUTHOR>

      <TITLE>XML in Scotland</TITLE>

      <REVIEW>

         <EM>The <EM>best</EM> ever!</EM>

      </REVIEW>

   </BOOK>

</libro>
```

Objectiu de la consulta

Amb text() podíem veure el contingut dels nodes. Ara emprarem name() per veure el nom dels nodes.

Consulta XQuery

```
for $x in /ejemplo//* return $x/name()
```

Explicació de la consulta

Per a cada node que sigui net de “ejemplo” mostrar el nom de l’etiqueta.

Resultat

```
elemento

camp1

camp2

elemento

camp1

camp3
```

Objectiu de la consulta

Extreure valors del atributs dels nodes “elemento”.

Consulta XQuery

```
for $x in /ejemplo/elemento

return $x/@id
```

Explicació de la consulta

Per a cada node elemento accedim al valor de la seva metadada.

Resultat

```
elem1

elem2
```

Objectiu de la consulta

Extreure el nom dels atributs dels nodes “elemento”.

Consulta XQuery

```
for $x in /ejemplo/elemento

return $x/@*/name()
```

Explicació de la consulta

Per a cada node “elemento” mostrem el nom de les seves metadades.

L’asterix fa de comodí per representar qualssevol metadada del node.

Resultat

```
id

id
```

Objectiu de la consulta

Provar una funcio d’Xquery que faci la feina d’un select distinct de SQL.

Consulta XQuery

```
for $p in distinct-values(//elemento) return $p
```

Explicació de la consulta

El millor per comprendre com funciona distinct-values és comparar la consulta amb distinct amb la mateixa consulta sense el distinct-values.

Aquesta seria la consulta sense el distinct i el seu resultat:

```
for $p in //elemento return $p
```

```
<elemento id="elem1">

   <camp1>valor1_1</camp1>

   <camp2>valor2</camp2>

</elemento>

<elemento id="elem2">

   <camp1>valor1_2</camp1>

   <camp3>valor3</camp3>

</elemento>.
```

Resultat

```
valor1_1valor2

valor1_2valor3
```