ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

Reasoner Over Temporal Xml

Διδάσκον καθηγητής: Παπαδάκης Νικόλαος
Φοιτητής: Σαρτζετάκης Παναγιώτης
Αριθμ.Μητρ:1999

Περιεχόμενα

1. Εισαγωγή

Στόχος της παρακάτω εργασίας είναι να καταφέρουμε να πάρουμε κάποια δεδομένα τα οποία μας είναι χρήσιμα και να τα ξεχωρίσουμε από τα υπόλοιπα με την βοήθεια των μαθηματικών. Πιο συγκεκριμένα θα χρησιμοποιήσουμε τις **Πράξεις Συνόλων.** Θα θεωρήσουμε τα δεδομένα μας σύνολα και μέσα από διάφορες πράξεις συνόλων, χρησιμοποιώντας κυρίως την τομή, θα προσπαθήσουμε να βγάλουμε ένα αποτέλεσμα και κάποια συμπεράσματα. Δηλαδή ξέροντας την εξάρτηση κάποιων υποσυνόλων μεταξύ τους τότε μπορέσουμε να ξέρουμε αν από τα σύνολα τους θα προκύψει το επιθυμητό αποτέλεσμα.

Έχοντας λοιπόν σαν βάση τις **Πράξεις Συνόλων** θα προσπαθήσουμε με την βοήθεια της **XML** και της **Xquery** να υλοποιήσουμε την παραπάνω ιδέα.
Με την βοήθεια της **XML** θα καταφέρουμε να δημιουργήσουμε αυτά τα υποσύνολα και έπειτα χρησιμοποιώντας την **Xquery** (με την βοήθεια της **Exist**) θα κάνουμε στην ουσία τις πράξεις συνόλων τα οποία επί της ουσίας ορίζονται από τα υποσύνολα τους.

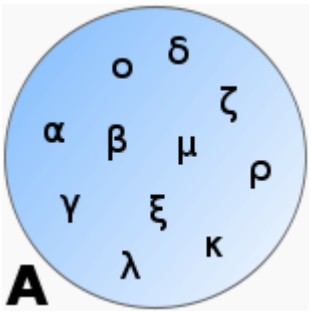
Σε μια προσπάθεια να γίνονται όλες οι παραπάνω διαδικασίες όσο πιο αυτοματοποιημένα γίνετε θα χρησιμοποιήσουμε αρκετά και την **Java**.Μέσα από την **Java** θα δημιουργήσουμε το **XML** αρχείο αλλά και πολλά **XML** αρχεία χρησιμοποιώντας κάθε φόρα αυτό που χρειαζόμαστε για να πάρουμε τις πληροφορίες που χρειαζόμαστε. Έπειτα το αρχείο που χρειαζόμαστε θα ανέβει στην **Exist** προκείμενου να κάνουμε τα ερωτήματα τα οποία θέλουμε και να πάρουμε και το ανάλογο αποτέλεσμα.

Στην συνέχεια της Εισαγωγής θα προσπαθήσω να σας εξηγήσω τις μαθηματικές ιδέες που θα χρησιμοποιησουμε αλλα και πως λειτουργουν τα εργαλεια που θα μας βοηθησουν να επιτύχουμε το στόχο μας.

1.1 Εισαγωγή στα Σύνολα και στα Υποσύνολα

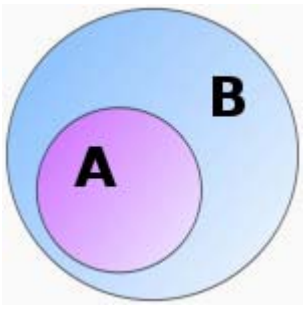
1.1.1 Τι είναι σύνολο και υποσύνολο

Σύνολο ονομάζεται μια καλώς ορισμένη συλλογή από διακεκριμένα αντικείμενα, τα οποία καλούνται **στοιχεία** ή **μέλη** του συνόλου. Όταν λέμε ότι είναι *καλώς ορισμένη* συλλογή, εννοούμε ότι θα πρέπει να γνωρίζουμε με ακρίβεια αν ένα αντικείμενο ανήκει στο σύνολο ή όχι. Επίσης, να σημειώσουμε ότι τα στοιχεία ενός συνόλου πρέπει να είναι *διακεκριμένα* αντικείμενα. Τα στοιχεία δεν μετρούν πολλαπλές φορές, για παράδειγμα το σύνολο των γραμμάτων της λέξης "Επ" δεν περιέχει 3 στοιχεία, περιέχει 2 διακεκριμένα γράμματα: π, ε



Το σύνολο
Α={α,ο,γ,β,κ,ρ,λ,ξ,ζ,μ,δ}

Έστω σύνολα Α και Β.
Ονομάζουμε το Α **υποσύνολο** του Β αν κάθε στοιχείο του Α είναι και στοιχείο του Β, δηλαδή το Α είναι υποσύνολο του Β αν και μόνο αν
$x \in A \Rightarrow x \in B$



Το Α είναι
υποσύνολο του Β

1.1.2 Πράξεις Συνόλων

Ένωση των συνόλων Α και Β είναι ένα νέο σύνολο, που συμβολίζεται με $A \cup B$και αποτελείται από όλα τα στοιχεία τα οποία ανήκουν σε ένα τουλάχιστον από τα σύνολα Α και Β, δηλαδή

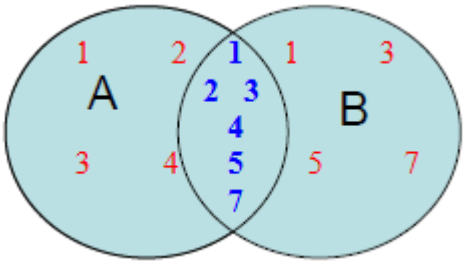
$$A \cup B = \{ x \mid x \in A \vee x \in B \}$$

Παράδειγμα

Έστω το σύνολο Α = {1,2,3,4} και το σύνολο Β={1,3,5,7}.

Τότε η ένωση τους θα είναι:

$$A \cup B = \{ 1,\ 2,\ 3,\ 4,\ 5,\ 7 \}$$



Τομή των συνόλων Α και Β είναι ένα νέο σύνολο, που συμβολίζεται με $A \cap B$και αποτελείται από όλα τα στοιχεία τα οποία ανήκουν ταυτόχρονα στο σύνολο Α και στο σύνολο Β, δηλαδή

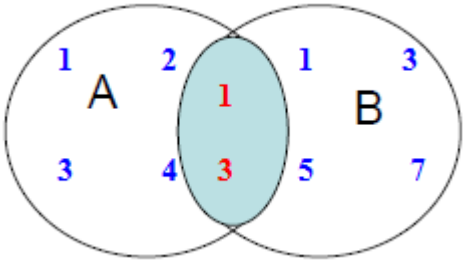
$$A \cap B = \{ x \mid x \in A \wedge x \in B \}$$

Παράδειγμα

Έστω το σύνολο Α = {1,2,3,4} και το σύνολο Β={1,3,5,7}.

Τότε η τομή τους θα είναι:

$$A \cap B = \{ 1,\ 3 \}$$



(*4)

1.2 Εισαγωγή στην XML

1.2.1 Χαρακτηριστικά της XML

Με την τεράστια εξάπλωση του παγκόσμιου ιστού, η ανάγκη για ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων και πλατφορμών έγινε επιτακτική
Το κύριο πρόβλημα είναι ότι η μορφή και ο τύπος των δεδομένων ποικίλλει. Μπορεί να είναι αρχεία κειμένου, δεδομένα βάσεων δεδομένων, μεταδεδομένα κτλ.
Επιτακτική ανάγκη για ένα κοινό πρότυπο αναπαράστασης και ανταλλαγής δεδομένων, κοινό για όλες τις πλατφόρμες

1.2.1.1 Η ιστορία της XML

Οι ρίζες της XML μπορούν ν' αναζητηθούν στην εκρηκτική ανάπτυξη του Παγκόσμιου Ιστού στα μέσα της δεκαετίας του 1990 και στους πολέμους των browser που έλαβαν χώρα μεταξύ της Microsoft Corporation και της Netscape Corporation, όπου καθεμιά από αυτές πάλευε για την απόλυτη κυριαρχία.

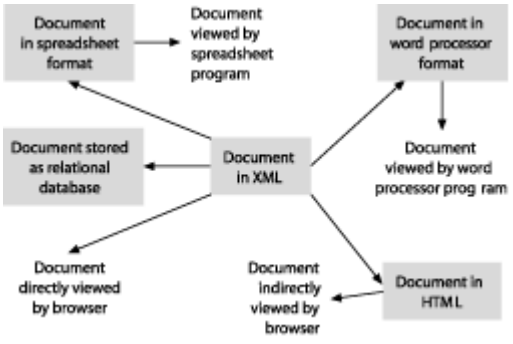
Καθώς το Web γινόταν όλο και πιο μεγάλο, και όλο και περισσότεροι χρήστες το χρησιμοποιούσαν, άρχισαν ν' ανακαλύπτονται από τους προγραμματιστές, που χρησιμοποιούσαν HTML, διάφορα προβλήματα:

- Το γεγονός ότι ο ίδιος πόρος HTML εμφανιζόταν με διάφορες μορφές, ανάλογα με τον browser που χρησιμοποιούνταν. Αυτό σήμαινε ότι, οι σχεδιαστές των ιστοσελίδων έπρεπε το λιγότερο να διπλασιάσουν τις προσπάθειές τους.
- Ορισμένοι κατασκευαστές browser ανέπτυξαν εργαλεία HTML που δεν ήταν αναγνωρίσιμα από άλλους browser.
- Ήταν σχεδόν αδύνατον να διακριθεί οποιαδήποτε σημαντική μέσα σε μια ιστοσελίδα: πουθενά αλλού αυτό δεν ήταν πιο προφανές από ότι στην χρήση των μηχανών αναζήτησης. Ακόμη και στα μέσα της δεκαετίας του 1990 πολλοί χρήστες εξέφραζαν την απογοήτευσή τους όσον αφορούσε αυτά τα προγράμματα εξαιτίας του όγκου των ανακτημένων αρχείων που επέστρεφαν οι μηχανές, τα οποία, στην καλύτερη περίπτωση, σχετίζονταν οριακά με την αναζήτηση. Η αιτία αυτής της "φτωχής" απόδοσης δεν ήταν η τεχνολογία των μηχανών αναζήτησης - στην πραγματικότητα επρόκειτο για εξεζητημένα προγράμματα που αποτελούν αποδεικτικό της εξυπνάδας των προγραμματιστών - αλλά το ότι τα δεδομένα που επεξεργάζονταν, η σελίδες HTML, δεν έδιναν πολλά στοιχεία για το περιεχόμενό τους.

Εξαιτίας αυτών των προβλημάτων η Κοινοπραξία Παγκόσμιου Ιστού, η ομάδα που ελέγχει την διαδικασία τυποποίησης του Ιστού, αποφάσισε το 1996 ν' αναπτύξει μία σημειακή γλώσσα που μελλοντικά θα υποσκέλιζε την HTML. Τα στόχοι αυτής της γλώσσας ήταν:

- Να χρησιμοποιείται εύκολα στο Internet.
- Να μπορεί να υποστηρίζει πολλές εφαρμογές οι οποίες θα κυμαίνονται από browser μέχρι βάσεις δεδομένων μηχανών αναζήτησης.
- Να είναι συμβατή με την SGML(Standard Generalized Markup Language), την γλώσσα επεξεργασίας κειμένου που αποτέλεσε την έμπνευση για την HTML.
- Να μην αποτελεί πολύπλοκη διαδικασία η ανάπτυξη επεξεργαστών κειμένων γραμμένων σε γλώσσες που θα βασίζονταν σε XML, για παράδειγμα θα έπρεπε να είναι εύκολη η εγγραφή ενός προγράμματος για τον έλεγχο της σαφήνειας ενός κειμένου πόρου.
- Ο αριθμός των προαιρετικών εργαλείων της γλώσσας να είναι χαμηλός.
- Να είναι εύκολη η ανάγνωση και κατανόηση των αρχείων XML.
- Να είναι εύκολο ν' αναπτυχθούν, με την χρήση απλών συντακτών, αρχεία γραμμένων σε γλώσσα βασιζόμενη σε XML.

Το 1998 η γλώσσα, XML, παρουσιάστηκε παγκόσμια ως μία τελευταία υπόδειξη της Κοινοπραξίας Παγκόσμιου Ιστού. Σαν αποτέλεσμα, έγινε μία σταθερά για το Internet. Βασιζόταν στην γλώσσα επεξεργασίας κειμένων SGML, η οποία αποτέλεσε την έμπνευση για την HTML. Ο ρόλος της XML συνοψίζεται στο παρακάτω σχήμα, το οποίο παρουσιάζει την σχέση της με άλλες μορφές αποθήκευσης και παρουσίασης.



(*5)

1.2.1.2 Τι είναι XML

Η XML ονομάστηκε έτσι από τα αρχικά των eXtensive Markup Language. Είναι ένα σύνολο από κανόνες για τη δημιουργία ετικετών (tags) που περιγράφουν τα δεδομένα ενός εγγράφου καθώς και προσδιορίζουν και τα διάφορα μέρη από τα οποία αποτελείται ένα έγγραφο. Είναι μια μεταγλώσσα, με την οποία μπορούμε να ορίσουμε άλλες γλώσσες σήμανσης. Με την XML μπορεί κάποιος να δημιουργήσει ένα σύνολο από ετικέτες που επιθυμεί να χρησιμοποιήσει όπως αυτός επιθυμεί.
Παρακάτω ακολουθεί ένα πολύ απλό παράδειγμα ενός XML αρχείου

```
<STUDENTS>
<AM>1999</AM>
<GENDER>MAN</GENDER>
<SURNAME>SARTZETAKIS</SURNAME>
<NAME>PANAGIOTIS</NAME>
<BIRTHDATE>12-5-1987</BIRTHDATE>
</STUDENTS>
```

Πλεονεκτήματα της XML

- Κοινό πρότυπο μεταξύ διαφορετικών Πλατφορμών
- Αυτό-περιγραφική γλώσσα
- Αποθήκευση σε ASCII κείμενο
- Ευελιξία στη δομή καθώς ο καθένας δημιουργεί όσες και όποιες ετικέτες θέλει
- Ευανάγνωστα αρχεία
- Πληθώρα τεχνολογιών και εργαλείων
- Οι περισσότερες εφαρμογές υποστηρίζουν την εξαγωγή και εισαγωγή στοιχείων από έγγραφα XML

Στόχοι της XML

1. Απλή (όπως η HTML – αλλά όχι τόσο απλή)
    - Συγκεκριμένοι **συντακτικοί** κανόνες, για τον περιορισμό συντακτικών λαθών.
    - η σύνταξη **ορίζει** την δομή (ιεραρχική), και ονοματίζει δομημένα κομμάτια (element names) – είναι **αυτο-χαρακτηριζόμενα δεδομένα**
2. Επεκτάσιμη (σε αντίθεση με την HTML)
    - Μπορείς να δημιουργήσεις την δικιά σου γλώσσα από tags/elements
    - **Η αυστηρότητα του συντακτικού** εξασφαλίζει την εγκυρότητα της επεξεργασίας
3. Σχεδιασμένη για ένα **κατανεμημένο περιβάλλον** (όπως η HTML)
4. Μπορεί να '**αναμείξει'** διαφορετικούς τύπους δεδομένων (σε αντίθεση με την HTML)

Καινοτομίες της XML

1. Οι σημαντικότερες αλλαγές που επέφερε η XML είναι
   a. Δεδομένα
   b. Αρχιτεκτονική
   c. Λογισμικό

   a. Δεδομένα
   - Αποδέσμευση των δεδομένων από τις εφαρμογές
   - Δημιουργία επιχειρηματικών λεξικών
   - Ανάπτυξη της B2B επικοινωνίας

   b. Αρχιτεκτονική
   - Ανάπτυξη κατανεμημένων υπολογιστικών συστημάτων
   - Χρήση της XML και του διαδικτύου (TCP/IP) για επικοινωνία και ανταλλαγή
     δεδομένων και πληροφοριών

   c. Λογισμικό
   - Με τη χρήση της XML επεκτείνεται η δυνατότητα επικοινωνίας διαφόρων
     υποπρογραμμάτων και εφαρμογών μεταξύ τους
   - Προγράμματα βασισμένα σε modules για τη λύση συγκεκριμένων προβλημάτων
   - Απλότητα, Ευελιξία

Η διαφορά μεταξύ της XML και της HTML

Η XML δεν δημιουργήθηκε με σκοπό να αντικαταστήσει της HTML,έχουν σχεδιαστεί για να εξυπηρετεί η κάθε μια από αυτές διαφορετικό σκοπό:

- Η XML δημιουργήθηκε με σκοπό να μεταφέρει και να αποθηκεύει δεδομένα, δίνοντας ιδιαίτερη έμφαση στο τι είδους δεδομένα είναι αυτά.
- Η HTML δημιουργήθηκε με σκοπό να εμφανίζει τα δεδομένα, δίνοντας ιδιαίτερη έμφαση στο πως πρέπει τα δεδομένα αυτά να εμφανίζονται

Τι δεν είναι XML

- Δεν είναι γλώσσα προγραμματισμού
- Δεν είναι ένα λογισμικό
- Δεν είναι περιβάλλον ανάπτυξης λογισμικού
- Δεν είναι εργαλείο ανάπτυξης ιστοσελίδων

1.2.2 Δομή της XML

Δομή

- Ένα XML έγγραφο μοιάζει με ένα HTML έγγραφο
- Αποτελείται από tags, τα οποία είναι υποχρεωτικό να κλείνονται (σε αντίθεση με την HTML).
  Π.χ
  <person>John</person>
- Επιτρέπονται άπειρα επίπεδα εμφωλευμένων tags
- Απαγορεύεται οι ετικέτες να ξεκινούν με 'XML' είτε σε πεζά είτε σε κεφαλαία

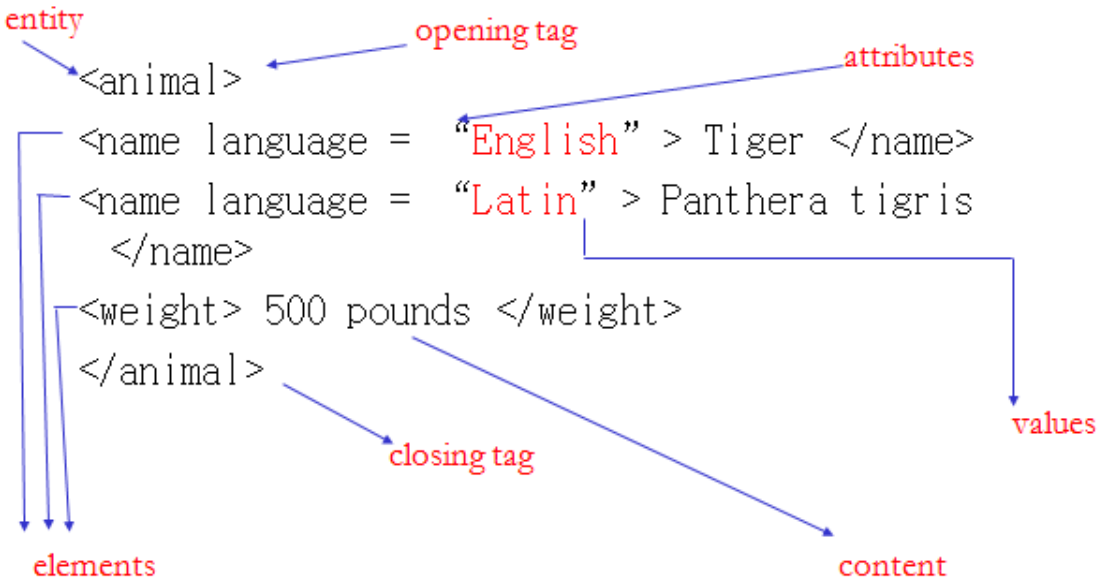
Παραδείγματα

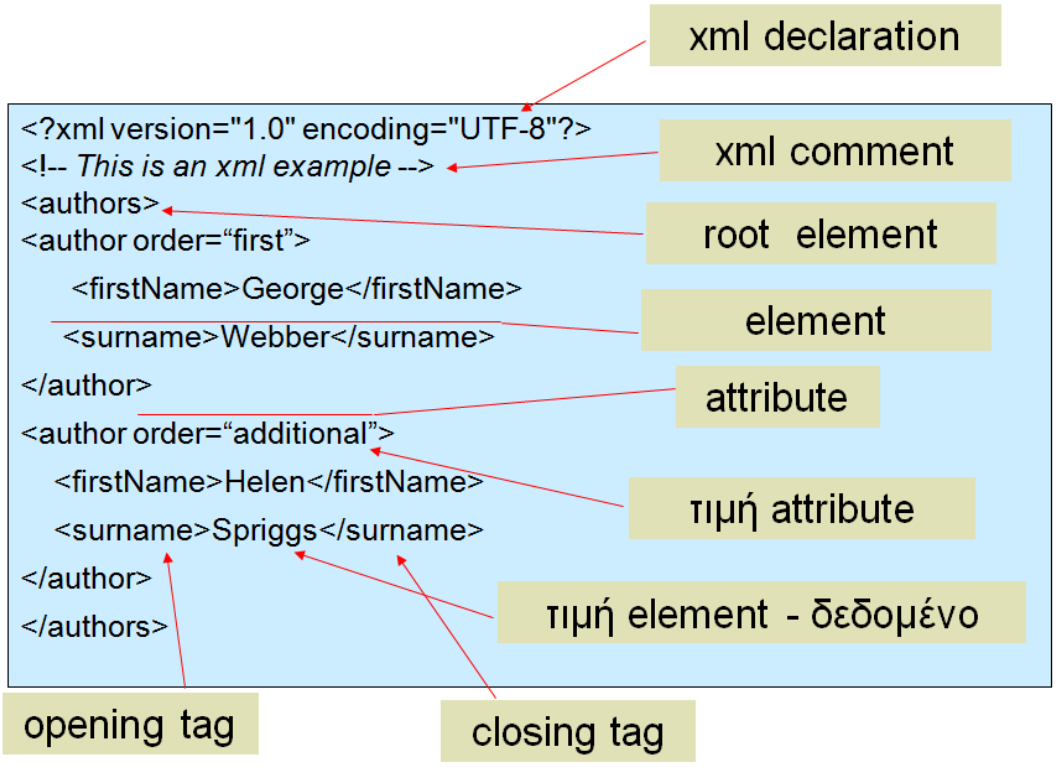
```
<root>
      <child>
            <subchild>.....</subchild>
      </child>
</root>
```

```
<school >
   <student>
         <name>Panos<name/>
   </student>
 </school>
```

```
<bookstore>
  <book category="CHILDREN">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Βασικά συνθετικά του συντακτικού της XML

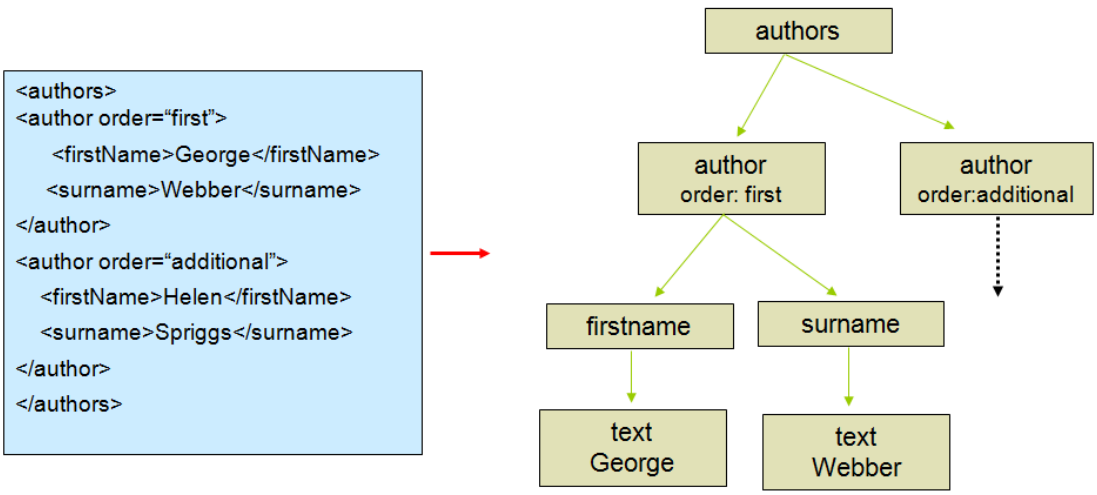




- **Element**: Το βασικό δομικό στοιχείο ενός XML εγγράφου
  <book> XML Language </book>
- **Attribute:** Ιδιότητα ενός element
  <book author="John">XML </book>
  Τα attributes μπορούν να γραφούν και με τη μορφή εμφωλευμένων elements. Π.χ
  <book>
  <author> John </author>
  <title> XML </title>
  </book>
  Η χρήση attributes κάνει το έγγραφο πιο δυσανάγνωστο και συνήθως αποφεύγεται.
- **Entity:** Οι οντότητες είναι αλφαριθμητικά που χρησιμοποιούνται ως συντομογραφίες άλλων αλφαριθμητικών.
  <!Entity message "Welcome">
  Με αυτόν τον τρόπο, όταν γράφουμε &message θα ισοδυναμεί με "Welcome"

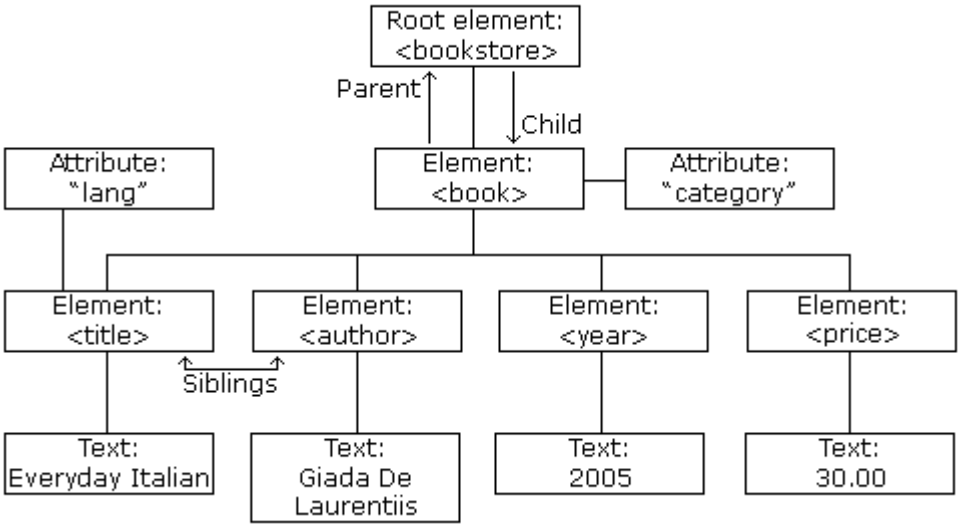
Η XML ως δενδρική δομή

- Κάθε xml έγγραφο (document) ενέχει την ιεραρχική σχέση πατέρα (parent node) παιδιών (child nodes), ξεκινώντας από το root element που είναι ο πατέρας όλων.
- Κάθε xml έγγραφο μπορεί να παρουσιαστεί ως μία δενδρική δομή, π.χ.



Παραδείγματα

```
<bookstore>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```



XML DOM (Document Object Model)

- Το DOM μοντέλο ορίζει έναν W3C (World Wide Web Consortium) στάνταρντ τρόπο προσπέλασης και δυναμικής διαχείρισης ενός xml εγγράφου (document).
- Το DOM μοντέλο παρουσιάζει ένα έγγραφο ως δενδρική δομή αποτελούμενη από κόμβους (nodes) όπου κάθε κόμβος μπορεί να είναι ένα element, ένα attribute, ένα σχόλιο ή απλό κείμενο.
- Ολόκληρο το xml έγγραφο αποτελεί έναν document κόμβο (node).
- Κάθε document έχει ένα root element από το οποίο αρχίζει και το οποίο είναι ουσιαστικά η κορυφή της δενδρικής (ιεραρχικής) δομής.
- Όλα τα nodes, εκτός από το root element, έχουν έναν πατέρα (parent node).
- Ένα node, εκτός από τα φύλλα του δένδρου, μπορεί να έχει έναν οποιονδήποτε αριθμό παιδιών (child nodes)
- Τα text nodes δεν έχουν παιδιά.
- Τα nodes που βρίσκονται στο ίδιο επίπεδο ονομάζονται siblings.

Παράδειγμα

```
<authors>
<author order="first">
    <firstName>George</firstName>
   <surname>Webber</surname>
</author>
<author order="additional">
   <firstName>Helen</firstName>
   <surname>Spriggs</surname>
</author>
</authors>
```

- Το element authors είναι το root element.
- Το element authors έχει δύο element author ως παιδιά τα οποία θεωρούνται sibling nodes.
- Το κάθε element author έχει δύο παιδιά, το element firstName και το element surname.
- Κάθε ένα από τα elements firstName και surname έχουν ως παιδιά text nodes.
- Κάθε element author σχετίζεται με το attribute order (τα attributes δεν έχουν πατέρα).
- Κάθε attribute order έχει ως παιδί ένα text node που είναι η τιμή του.

(*1,3,6)

1.2.3 Γραμματική της XML

1.2.3.1 Πως ορίζουμε τη γραμματική της XML

- Δύο τρόποι:
  - **XML Document Type Declaration (DTD)** – μέρος του XML specification.
  - **XML Schema** – XML specification, επιτρέπει περισσότερο 'δυνατούς' ορισμούς (τύπους δεδομένων)
- Δύο κατηγορίες XML δεδομένων:
  - **Well-formed** Όταν ένα XML document είναι συντακτικά σωστό
  - **Valid(Έγκυρο)** Ένα XML document το οποίο είναι και well-formed και συνεπές με ένα συγκεκριμένο DTD (or Schema)
- Τι ορίζουν τα DTDs και τα schema:
  - Ορίζουν elements και attributes ονόματα, ιεραρχικά φωλιασμένους κανόνες, περιεχόμενα των elements/περιορισμούς
- Τα XML Schemas είναι πιο ισχυρά από τα DTDs. Χρησιμοποιούνται συχνά για ***type validation*** ή για συσχέτιση ER σχημάτων με XML μοντέλα

XML Schema

- Ορίζει έναν τύπο xml εγγράφου περιγράφοντας τους περιορισμούς που πρέπει να υπακούουν όλα τα xml έγγραφα αυτού του τύπου αναφορικά με την δομή και το περιεχόμενό του.
- Ένα xml schema επεκτείνει τον έλεγχο εγκυρότητας ενός xml εγγράφου από τους βασικούς συντακτικούς κανόνες της XML (well-formedness) στην υπακοή των κανόνων και περιορισμών που θέτει το XML schema (validation).
- Ο έλεγχος εγκυρότητας μπορεί να πραγματοποιηθεί με έναν κατάλληλο validating XML parser.
- Ένα XML schema εκφράζεται μέσω μιας γλώσσας XML schema (XML schema language) Παραδείγματα:
  - DTD (Document Type Definition). Περιγράφει τους περιορισμούς στην δομή ενός τύπου εγγράφου όχι μόνο σε XML αλλά και στην γενικότερη γλώσσα SGML.
  - XML Schema (W3C). Αποτελεί επέκταση του DTD ορίζοντας τύπους δεδομένων και namespaces.
  - RELAX NG. Είναι σύγχρονη γλώσσα της XML Schema, παρέχοντας τις ίδιες δυνατότητες έκφρασης και επιπλέον μία σύντομη (compact) μη XML γραφή όμως δεν έχει επεκταθεί η χρήση της.

Παράδειγμα

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">
   <xs:element name="accountID" type="xs:string"/>
   <xs:element name="acknowledgeReceipt" type="xs:string"/>
   <xs:complexType name="amountType">
      <xs:simpleContent>
         <xs:restriction base="xs:string">
            <xs:attribute name="currency" use="required">
               <xs:simpleType>
                  <xs:restriction base="xs:NMTOKEN">
                     <xs:enumeration value="USD"/>
                       . . . (some stuff omitted) . . .
                  </xs:restriction>
               </xs:simpleType>
            </xs:attribute>
         </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="fromType">
      <xs:sequence>
         <xs:element name="amount" type="amountType"/>
         <xs:element ref="transitID" minOccurs="0"/>
         <xs:element ref="accountID"/>
         <xs:element ref="acknowledgeReceipt"/>
      </xs:sequence>
       . . .
```

DTD (Document Type Definition)

- Ο ορισμός των ετικετών που θα χρησιμοποιηθούν είναι προαιρετικός και γίνεται μέσω του σχήματος DTD (Document Type Definition)
- Ένα DTD αρχείο περιέχει ένα λεξικό για κάποια συγκεκριμένα XML έγγραφα
- Στο DTD αρχείο ορίζονται ποιες ετικέτες θα χρησιμοποιηθούν καθώς και οι σχέσεις μεταξύ ετικετών (π.χ ποιες περιέχουν ποιες κτλ).
- Τα DTD εισάγονται στον πρόλογο των XML αρχείων.
- Μπορεί να περιλαμβάνουν τις δηλώσεις των DTD εσωτερικά ή εξωτερικά

Τι Ορίζει

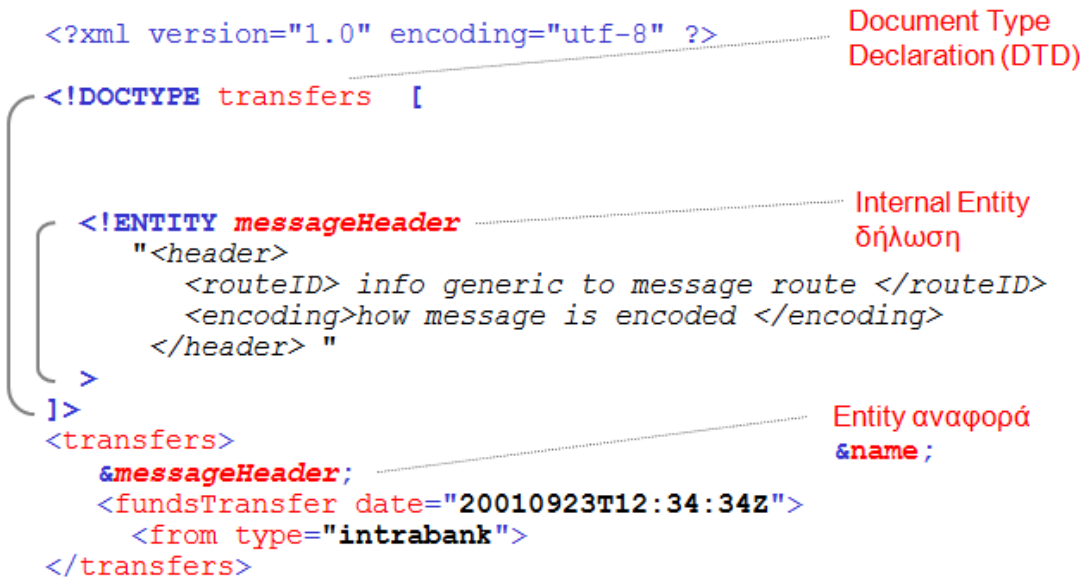
- Επιτρεπόμενη σειρά από tags και εμφωλιασμούς
- Τιμές attributes, τον τύπο τους και defaults
- Τα ονόματα εξωτερικών αρχείων που χρησιμοποιούνται (και περιέχουν ή μη XML περιεχόμενο)
- Το format των εξωτερικών ΜΗ XML Δεδομένων που μπορεί να περιέχουν
- Οντότητες που μπορεί να χρησιμοποιηθούν

Παράδειγμα DTD

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

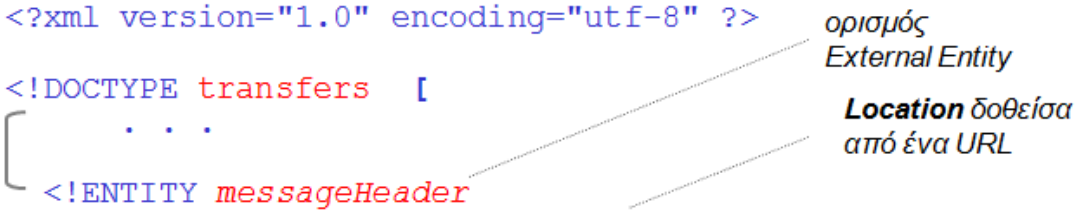
Παράδειγμα με εσωτερικό DTD




```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE transfers  [

  <!ENTITY messageHeader
     "<header>
        <routeID> info generic to message route </routeID>
        <encoding>how message is encoded </encoding>
      </header> "
  >
]>
<transfers>
   &messageHeader;
   <fundsTransfer date="20010923T12:34:34Z">
     <from type="intrabank">
</transfers>
```

Παράδειγμα με εξωτερικό DTD




```
<?xml version="1.0" encoding="utf-8" ?>

<!DOCTYPE transfers  [
      . . .

  <!ENTITY messageHeader
        SYSTEM "http://www.somewhere.org/dir/head.xml"
   >

]>
<transfers>
   &messageHeader;
   <fundsTransfer date="20010923T12:34:34Z">
     <from type="intrabank">

</transfers>
```

XML PARSERS

Για να μπορέσουμε να προσπελάσουμε και να διαχειριστούμε τα nodes ενός xml document θα πρέπει να χρησιμοποιήσουμε έναν parser.

Υπάρχουν δύο γενικές κατηγορίες xml parsers:

- SAX (Simple API for XML parser). Είναι ένας μονής κατεύθυνσης (one pass) σειριακός parser ο οποίος εκτελεί ενέργειες ανάλογα με τα γεγονότα (events) που συναντάει. Τέτοια events μπορεί να είναι ένα element node ή ένα text node.
  - http://www.megginson.com/SAX/index.html
  - Είναι ***event-based***
  - Ο Parser αναφέρει events κάθε φορά που 'βλέπει' ένα tag/attribute/text node/unresolved external entity/other
  - Ο προγραμματιστής βάζει "event handlers" για να δεσμεύσει αυτό το event report

Πλεονεκτήματα:

- Εύκολο στην χρήση
- Πολύ γρήγορο (μικρό χρόνος επεξεργασίας πριν το parsing)
- Χαμηλές απαιτήσεις μνήμης (δεν φορτώνεται ολόκληρο το έγγραφο στην μνήμη)
- Μπορεί να φορτώσει μέχρι και αρχεία μεγέθους gigabyte

Μειονεκτήματα:

- Πρέπει να κάνεις πολλά μόνος σου
- Δεν είναι πολύ χρήσιμο αν χρειάζεται να μεταβάλεις δυναμικά το κείμενο, οπότε χρειάζεται να είναι φορτωμένο όλο στην μνήμη

- DOM parser. Χτίζει στην μνήμη την αντίστοιχη δενδρική δομή όλου του xml document τους κόμβους του οποίου μπορεί να προσπελάσει προς κάθε δυνατή κατεύθυνση.

  - http://www.w3.org/DOM/
  - Parser δημιουργεί ένα in-memory δέντρο σύμφωνα με το κείμενο
  - Το DOM interface ορίζει μεθόδους για πρόσβαση και τροποποίηση του δέντρου

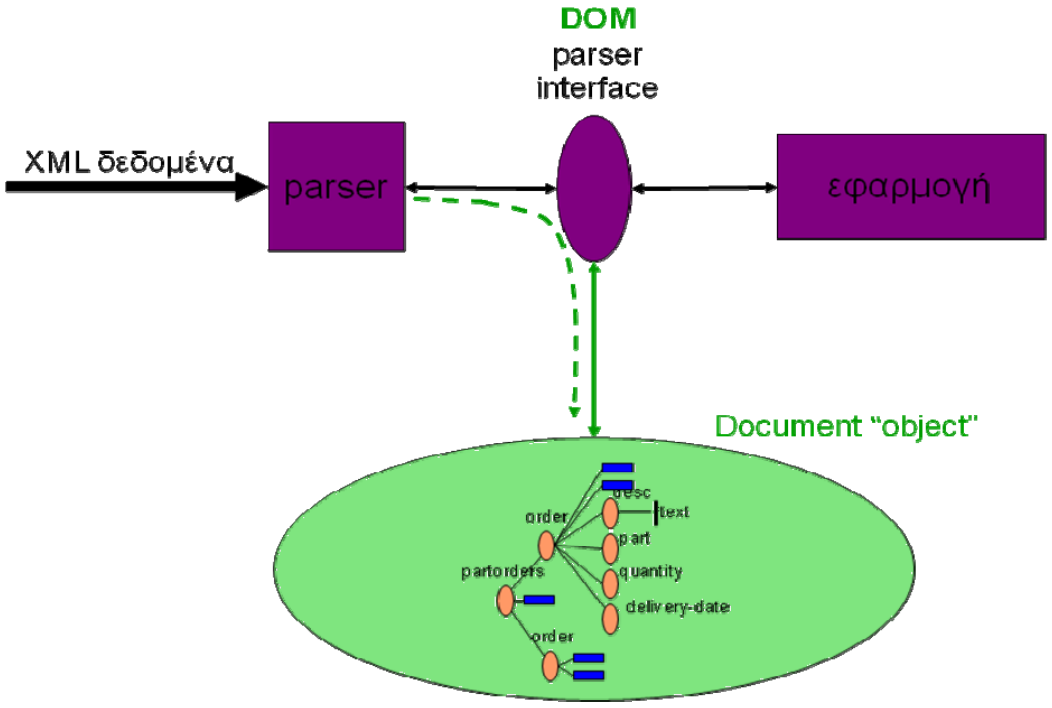
Πλεονεκτήματα:

- Χρήσιμο για δυναμική τροποποίηση και πρόσβαση του 'δέντρου' (κειμένου)
- Χρήσιμο σε ερωτήσεις (i.e. looking for data) τα οποία εξαρτώνται από την δομή του κειμένου [π.χ. element.childNode("2").getAttributeValue("bicycle")]
- Κοινό interface για πολλές γλώσσες προγραμματισμού (C++, Java, ...)

Μειονεκτήματα:

- Μπορεί να γίνει αργό (χρειάζεται να παράγει το δέντρο), και μπορεί να χρειάζεται πολύ μνήμη
- Το DOM interface στο προγραμματισμό είναι δύσχρηστο, δεν είναι 'πολύ' object oriented

DOM Parser Μοντέλο Επεξεργασίας



Διαφορές:

- Οι SAX parsers είναι γρήγοροι και με πολύ λιγότερες απαιτήσεις σε μνήμη επιτρέποντας την επεξεργασία πολύ μεγάλων xml documents.
- Οι DOM parsers είναι απλούστεροι απαιτώντας λιγότερες κλήσεις μεθόδων για την προσπέλαση και την διαχείριση των κόμβων του DOM tree και ιδανικοί όταν η δενδρική δομή πρέπει να διαπεραστεί πολλές φορές και σε διαφορετικές κατευθύνσεις.

(*1,3,6)

1.3 Εισαγωγή στην Xquery και Xpath

1.3.1 Εισαγωγή στην Xpath

Η XPath είναι γλώσσα ερωτήσεων σε XML τεκμήρια, η οποία οφείλει το όνομα της στην σύνταξη μονοπατιού που υιοθετεί η οποία προσομοιάζει με αυτήν των URLs. Η XPath, όπως έχουμε ήδη αναφέρει, αντιμετωπίζει το XML τεκμήριο σαν ένα δέντρο το οποίο περιέχει κόμβους. Σκοπός της είναι η δυνατότητα εύκολης μετακίνησης σε οποιοδήποτε κόμβο του εγγράφου, η δυνατότητα αναγνώρισης του τρέχοντα κόμβου και η δυνατότητα επιλογής μίας ομάδας κόμβων για περαιτέρω επεξεργασία. Διακρίνουμε επτά είδη κόμβων οι οποίοι είναι οι εξής:

- Κόμβοι ρίζας (root nodes)
- Κόμβοι στοιχείων (element nodes)
- Κόμβοι κειμένου (text nodes)
- Κόμβοι γνωρισμάτων (attribute nodes)
- Κόμβοι χώρων ονομάτων (namespace nodes)
- Κόμβοι οδηγιών επεξεργασίας (processing instruction nodes)
- Κόμβοι σχολίων (comment nodes)

1.3.2 Εισαγωγή στην Xquery

Η XQuery, η γλώσσα ερωτημάτων της XML, η οποία κατασκευάστηκε από την W3C είναι συμβατή, όπως έχει προαναφερθεί, με την XML, την XSLT, την XPath και την XML Schema. Παρά την μεγάλη χρηστικότητα της, δεν έχει αποτελέσει ακόμα καθιερωμένο πρότυπο. Μέχρι στιγμής, θα μπορούσαμε να πούμε, ότι είναι ένα πρόχειρο λειτουργικό εργαλείο. Αναμένεται, όμως, στο μέλλον να εξελιχθεί σε καθιερωμένο πρότυπο.

Η XQuery ακολουθώντας, την δομή της XPath αναπαριστά ένα δέντρο κόμβων. Τα είδη κόμβων είναι τα ήδη γνωστά μας από την XPath, τεκμηρίου (document nodes), στοιχείων (element nodes), γνωρισμάτων(attribute nodes), κειμένου (text nodes), χώρων ονομάτων (namespace nodes) , οδηγιών επεξεργασίας (processing instruction nodes) και σχολίων (comment nodes). Κάθε κόμβος έχει μία μοναδική ταυτότητα κόμβου που του δίνει τη δυνατότητα να ξεχωρίζει από άλλους κόμβους. Συγχρόνως, είναι επιτρεπτές οι ατομικές τιμές (atomic values), όπως οι συμβολοσειρές (string), οι ακέραιοι (integer), οι δεκαδικοί (decimal) και οι ημερομηνίες (date).

Ένα αντικείμενο (item) είναι μία μοναδική τιμή ή ένας μεμονωμένος κόμβος. Μία σειρά αντικειμένων αποτελεί μία ακολουθία. Στην XQuery κάθε τιμή είναι μία ακολουθία και γι' αυτό το λόγο δεν διακρίνεται διαφορά μεταξύ ενός αντικειμένου και μίας μακροσκελούς ακολουθίας.

Ο πρώτος κόμβος είναι σε ένα τεκμήριο είναι ο κόμβος του τεκμηρίου (document node), ο οποίος εμπεριέχει ολόκληρο το τεκμήριο, αναπαριστά, δηλαδή, το ίδιο το τεκμήριο. Οι κόμβοι στοιχείων, σχολίων και οδηγιών επεξεργασίας έχουν την ίδια σειρά με την οποία τους συναντάμε στην XML. Οι κόμβοι στοιχείων συναντιόνται πριν τα παιδιά τους (που είναι επίσης κόμβοι στοιχείων), τους κόμβους κειμένου, τους κόμβους σχολίων και τις οδηγίες επεξεργασίας. Τα γνωρίσματα δεν θεωρούνται παιδιά των στοιχείων, αλλά έχουν μία προκαθορισμένη θέση μέσα στο κείμενο, τα συναντάμε πριν τα παιδιά του στοιχείου στο οποίο βρίσκονται. Ιδιαίτερα σημαντικό, εδώ, είναι να σημειώσουμε, ότι κάθε κόμβος συναντάται μόνο μία φορά στο κείμενο.

(*2)

Παράδειγμα

Έχουμε το παρακάτω αρχείο catalog.xml

```
<catalog>
<product dept="WMN">
        <number>557</number>
        <name language="en">Fleece Pullover</name>
</product>
<product dept="ACC">
        <number>563</number>
        <name language="en">Floppy Sun Hat</name>
</product>
<product dept="ACC">
        <number>443</number>
        <name language="en">Deluxe Travel Bag</name>
</product>
<product dept="MEN">
        <number>784</number>
        <name language="en">Cotton Dress Shirt</name>
</product>
</catalog>
```

- Submit Query

doc("catalog.xml")/catalog/product[@dept = "ACC"]

- Result

```
<product dept="ACC">
        <number>563</number>
        <name language="en">Floppy Sun Hat</name>
</product>
<product dept="ACC">
        <number>443</number>
        <name language="en">Deluxe Travel Bag</name>
</product>
```

- Explain

doc("catalog.xml")→Φορτώνει το αρχείο **catalog.xml**

/catalog/product[@dept = "ACC"]→Ακολουθεί το path δίνοντας από ποιον κόμβο θα ξεκινήσει και σε ποιον θα καταλήξει. Έπειτα μόλις φτάσει στον επιθυμητό κόμβο θα ελέγξει αν η τιμή του attribute **dept** είναι αυτή που δίνουμε. Αν ναι τότε θα μας επιστρέψει όλα τα element αυτού του κόμβου.

Το Query θα μπορούσε να είναι και αυτό:

doc("catalog.xml")/ /product[@dept = "ACC"]

Με διπλό // πάει και βρίσκει κατευθείαν τον κόμβο που θέλουμε χωρίς να του δώσουμε όλο το path

(*7)

1.3.3. FLWOR εκφράσεις

Η FLWOR έκφραση είναι το ανάλογο SELECT-FROM-WHERE κατασκευή σε SQL και διαμορφώνει το σκελετό της XQuery έκφρασης. Μια FLWOR έκφραση αποτελείται από:

- FOR- όρος: δεσμεύει μια ή περισσότερες μεταβλητές σε μια ακολουθία τιμών που επιστρέφεται από μια άλλη έκφραση (συνήθως μια έκφραση μονοπατιών) και επαναλαμβάνει πέρα από τις τιμές.
- LET-όρος: επίσης δεσμεύει μια ή περισσότερες μεταβλητές αλλά χωρίς επανάληψη.
- WHERE-όρος: περιέχει ένα ή περισσότερα κατηγορήματα που φιλτράρει ή οριοθετεί το σύνολο κόμβων όπως παράγονται από τους FOR/LET όρους.
- ORDER BY: Καταφέρνουμε να ταξινομήσουμε τα αποτελέσματα μας με ότι εμείς κριτήριο θέλουμε
- RETURN-όρος: παράγει την εξωτερικά στοιχεία της FLWOR έκφρασης. Ο RETURN όρος συνήθως περιέχει έναν ή περισσότερους κατασκευαστές στοιχείων και/ή αναφορές στις μεταβλητές και εκτελείται μία φορά για κάθε κόμβο - αναφορά που επιστρέφεται από τους FOR/LET/WHERE όρους.

(*2)

Παράδειγμα

Δουλεύοντας το ίδιο αρχείο **catalog.xml**

- Submit Query

```
for $prod in doc("catalog.xml")/catalog/product
where $prod/@dept = "ACC"
order by $prod/name
return $prod/name
```

- Result

```
<name language="en">Deluxe Travel Bag</name>
<name language="en">Floppy Sun Hat</name>
```

- Explain

for $prod in doc("catalog.xml")/catalog/product→** Δηλώνουμε μια μεταβλητή **$prod την οποία την κάνουμε ίση με το παρακάτω path προκειμένου να μην το χρησιμοποιούμε συνέχεια
where $prod/@dept = "ACC"→Φιλτράρουμε το attribute **dept** και θέλουμε μόνο ότι είναι ισο με ACC
order by $prod/name→ Έπειτα κάνουμε μια ταξινόμηση κατά Όνομα
return $prod/name→ Και τέλος επιστρέφουμε τα Ονόματα(που τηρούν τα κριτήρια τα οποία θέτει το Where και είναι ταξινομημένα με το Order)

2.Ποιος είναι ο στόχος μας και πως θα τον επιτύχουμε

2.1 Ο στόχος μας

Θα προσπαθήσω να σας εξηγήσω τον στόχο μας με ένα παράδειγμα:

Ας υποθέσουμε ότι αν ένας δημόσιος υπάλληλος διαπράττει πλημμέλημα, τότε για τους επόμενους πέντε μήνες αυτός ο υπάλληλος θεωρείται «παράνομος». Όταν ένας δημόσιος υπάλληλος είναι παράνομος, τότε αυτός πρέπει να πάρει αναστολή και να μην μπορεί να πάρει προαγωγή για ολόκληρο το χρονικό διάστημα κατά το οποίο θεωρείται παράνομος . Επίσης, όταν ένας δημόσιος υπάλληλος έχει αναστολή, δεν μπορεί να λαμβάνει αποζημίωση μέχρι το τέλος της.
Κάθε δημόσιος υπάλληλος βαθμολογείται για το έργο του. Σε περίπτωση που λαμβάνει ένα κακό βαθμό, τότε θεωρείται κακός υπάλληλος. Σε περίπτωση που λαμβάνει ένα καλό βαθμό, τότε θεωρείται ένα καλός εργαζόμενος και μπορεί να λαμβάνει μπόνους αν δεν έχει κάποια αναστολή. Κάθε δημόσιος υπάλληλος λαμβάνει την αύξηση και την προαγωγή κάθε δύο και πέντε έτη, αντίστοιχα, αν δεν έχει διαπράξει κάποια παρανομία.
Μπορούμε να εντοπίσουμε πέντε πράξεις: misdemeanor,good grade, bad grade, take promotion ,take increase και επτά καταστάσεις good employee, illegal employee, take salary, take bonus, position suspended and salary.Η κατάσταση *position(p,l,t1)* σημαίνει πως ο δημόσιος υπάλληλος είναι σε μια θέση εργασίας *l* για *t1* μήνες ενώ το *salary(p,s,t1)* σημαίνει ότι ο δημόσιος υπάλληλος παίρνει έναν μισθό *s* για *t1* μήνες.
Οι άμεσες επιπτώσεις αυτών των πράξεων εκφράζονται με τους παρακάτω κανόνες:

1) ***Occur(misdemeanor(p),t)→illegal(p,[t,t+5])***
2) ***Occur(bad grade(p),t)→ -- good employee(p,[t, ∞))***
3) ***Occur(good grade(p),t)→ good employee(p,[t, ∞))***
4) ***Occur(take increase(p),t)^salary(p,s,24) →salary(p,s+1,0)***
5) ***Occur(take promotion (p),t)^position(p,l,60) →position(p,l+1,0)***

Επίσης έχουμε και το εξής περιορισμούς ακεραιότητας που οδηγούν σε έμμεσες επιπτώσεις των ακόλουθων πράξεων:

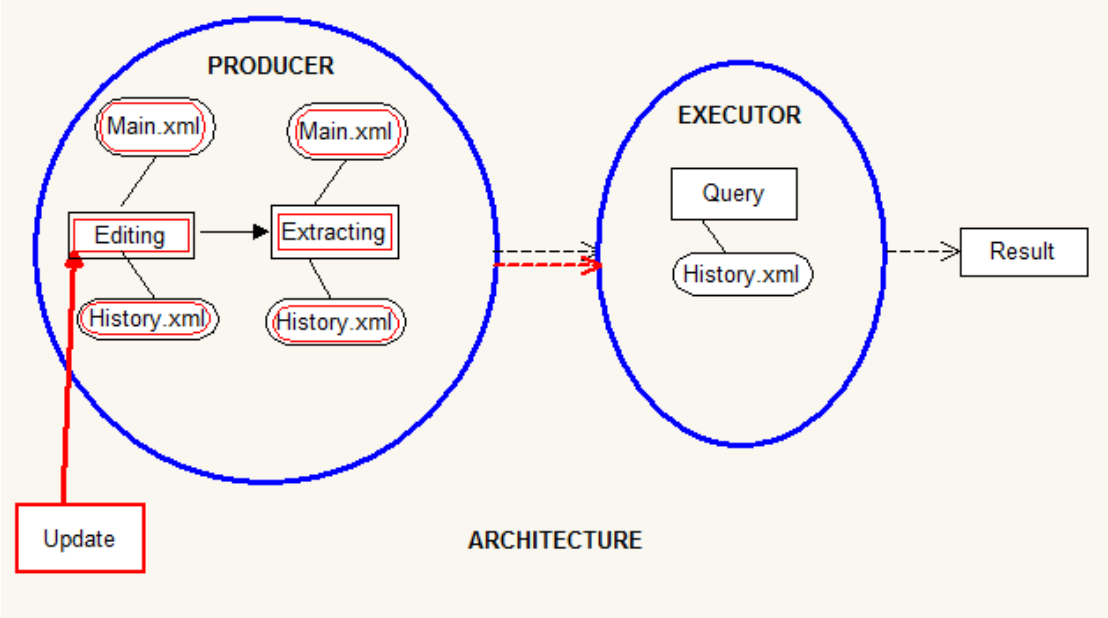
6) ***Illegal(p,t1) →Suspended(p,t1)***
7) ***Suspended(p,t1)→ -- take salary(p,t1)***
8) ***--Suspended(p,t)^good employee(p,t) →take bonus(p,t)***
9) ***--good employee(p,t1) →--take bonus(p,t1)***
10) ***--Suspended(p,t1)→ take salary(p,t1)***

Το θέμα το οποίο θα δουλέψουμε μοιάζει με τον κανόνα νούμερο 8. Ο κανόνας λέει ότι αν ο δημόσιος υπάλληλος δεν είναι σε αναστολή ΚΑΙ είναι καλός υπάλληλος μέσα σε κάποιο ορισμένο χρονικό διάστημα τότε θα πάρει μπόνους. Έτσι και εμείς θα φτιάξουμε μια XML βάση μέσα από την JAVA στην οποία θα υπάρχουν κάποια δεδομένα των οποίον οι τιμές τους θα αλλάζουν σε διάφορα χρονικά διαστήματα. Εμείς αυτό που θέλουμε είναι η τομή δυο διαφορετικών δεδομένων σε συγκεκριμένο χρονικό διάστημα να έχουν συγκεκριμένη τιμή. Τότε μόνο θα είναι το αποτέλεσμα της τομής αληθής, σε οποιαδήποτε άλλη περίπτωση το αποτέλεσμα θα είναι ψευδής.

(*8)

2.2 Πως θα επιτύχουμε το στόχο μας

Όλη η διαδικασία χωρίζεται αρχικά σε 2 μέρη. Το πρώτο μέρος είναι η δημιουργία της βάσης δεδομένων και το δεύτερο μέρος είναι η υποβολή ερωτημάτων σε αυτήν προκειμένου να δούμε τι αποτελέσματα θα πάρουμε.
Ας δούμε λίγο την αρχιτεκτονική του συστήματος μας και ας εξηγήσουμε λίγο το σχήμα που θα δούμε παρακάτω:



Όπως βλέπουμε στο σχήμα μας παραπάνω όλη η διαδικασία είναι στην ουσία χωρισμένη σε δυο μέρη. Ας δούμε λοιπόν πως λειτούργει το κάθε μέρος χωριστά.

PRODUCER

Αυτό είναι το στάδιο στο οποίο παράγουμε τα αρχεία που πρόκειται να χρησιμοποιήσουμε στην συνέχεια. Παράγουμε ότι ακριβώς χρειαζόμαστε για να το περάσουμε στο επόμενο στάδιο. Ένα κύριο σημείο αυτού του σταδίου είναι η **συνεχής ενημέρωση.** Οι τιμές αλλάζουν συνεχώς και καινούργια δεδομένα διαρρέουν πλέον στο σύστημα μας. Με λίγα λόγια κάθε φορά που κάτι αλλάζει ολόκληρη η διαδικασία ξεκινά από την αρχή.

EXECUTOR

Στο επόμενο στάδιο που είναι ο Executor γίνονται τα ερωτήματα από αρχεία που παράγει ο Producer. Γίνονται διάφορα ερωτήματα, αναλόγως τι ψάχνουμε να βρούμε. Με το να αλλάζουν συνεχώς τα δεδομένα κάθε φορά παίρνουμε διαφορετικά αποτελέσματα ή εκτελούμε και διαφορετικά ερωτήματα.
Όπως είναι λογικό μετά τον Executor παίρνουμε και το αποτέλεσμα μας. Πρέπει να γίνει όλη αυτή η διαδικασία προκειμένου να πάρουμε κάποια αποτελέσματα και να βγάλουμε τα ανάλογα συμπεράσματα.

Ας δούμε λοιπόν στην συνέχεια πως δημιουργείται η βάση και πως συνεχίζουμε βήμα βήμα.

2.2.1 Δημιουργία της βάσης

Η βάση θα είναι σε XML αλλά θα την δημιουργήσουμε με την βοήθεια της JAVA προκειμένου να αυτοματοποιήσουμε κάποιες διαδικασίες αλλά και για να έχουμε μεγαλύτερη ευελιξία στην βάση μας. Ας δούμε λοιπόν βήμα βήμα πως γίνεται η διαδικασία.
Σε πρώτη φάση θα βλέπουμε των κώδικα από την JAVA και θα τον εξηγήσουμε προκειμένου να δούμε πως λειτούργει και μετά θα δούμε το αποτέλεσμα το όποιο παράγει.

2.2.1.1 Δημιουργία του βασικού μέρους

Source

```
import java.io.*;

import org.dom4j.*;
import org.dom4j.io.OutputFormat;
import org.dom4j.io.XMLWriter;


public class makingXml {


   public static void main (String argv []) throws Exception{




        // Create the document
        Document document = DocumentHelper.createDocument();
        // Add the root                                    ΔΗΜΙΟΥΡΓΙΑ ΚΟΜΒΩΝ
        Element root =
document.addElement("test").addAttribute("id",(DateUtils.now("dd.MM.yy 'at'
hh:mm:ss")));

        Element tf1=root.addElement("tf").addAttribute("id", "A");
        tf1.addElement("name").addText("A");
        tf1.addElement("value").addText("TRUE");
        tf1.addElement("Starting_Date").addText((DateUtils.now("dd.MM.yy")));
        tf1.addElement("Starting_Time").addText((DateUtils.now("hh:mm:ss")));


        Element tf2=root.addElement("tf").addAttribute("id","B");
        tf2.addElement("name").addText("B");
        tf2.addElement("value").addText("FALSE");
        tf2.addElement("Starting_Date").addText((DateUtils.now("dd.MM.yy")));
        tf2.addElement("Starting_Time").addText((DateUtils.now("hh:mm:ss")));


        Element tf3=root.addElement("tf").addAttribute("id","C");
        tf3.addElement("name").addText("C");
        tf3.addElement("value").addText("FALSE");
        tf3.addElement("Starting_Date").addText((DateUtils.now("dd.MM.yy")));
        tf3.addElement("Starting_Time").addText((DateUtils.now("hh:mm:ss")));


        Element tf4=root.addElement("tf").addAttribute("id", "D");
```

```
        tf4.addElement("name").addText("D");
        tf4.addElement("value").addText("TRUE");
        tf4.addElement("Starting_Date").addText((DateUtils.now("dd.MM.yy")));
        tf4.addElement("Starting_Time").addText((DateUtils.now("hh:mm:ss")));


        Element tf5=root.addElement("tf").addAttribute("id", "E");
        tf5.addElement("name").addText("E");
        tf5.addElement("value").addText("FALSE");
        tf5.addElement("Starting_Date").addText((DateUtils.now("dd.MM.yy")));
        tf5.addElement("Starting_Time").addText((DateUtils.now("hh:mm:ss")));
```

ΕΠΙΛΟΓΗ ΜΟΡΦΗΣ ΕΞΟΔΟΥ

```
// Make a pretty output

                OutputFormat format = OutputFormat.createCompactFormat();
                format.setEncoding("UTF-8");
                format.setTrimText(false);
```

ΕΞΑΓΩΓΗ ΑΡΧΕΙΩΝ

```
// Save it

                XMLWriter writer = new XMLWriter(new
FileWriter("C:\\makingXml.xml"), format);
                writer.write(document);
                writer.close();


                XMLWriter writerA = new XMLWriter(new FileWriter("C:\\A.xml"),
format);
                writerA.write(tf1);
                writerA.close();

                XMLWriter writerB = new XMLWriter(new FileWriter("C:\\B.xml"),
format);
                writerB.write(tf2);
                writerB.close();


    }

    }
```

Explain

ΔΗΜΙΟΥΡΓΙΑ ΚΟΜΒΩΝ

Εδώ όπως βλέπουμε δημιουργούμε βήμα βήμα το αρχείο μας. Σε πρώτη φάση δημιουργούμε τον κάθε κόμβο έναν έναν χωριστά προσθέτοντας μέσα ότι στοιχεία θέλουμε και όπως τα θέλουμε χωρίς κανέναν περιορισμό. Αυτή την ευελιξία μας την προσφέρει το **dom4j jar** το οποίο με τον δεντροκεντρικό χαρακτήρα που έχει όπου στην ουσία σχεδιάζουμε το δένδρο μέσα από την JAVA.Για παράδειγμα ο πρώτος κόμβος θα πρέπει να είναι κάπως έτσι.

```
<tf id="A">
<name>A</name>
<value>TRUE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
</tf>
```

Ο κάθε κόμβος θα περιέχει το όνομα του, την τιμή που έχει και σε ποιο χρονικό διάστημα θα την έχει .Σε πρώτη φάση έχουμε μόνο της ημερομηνίες εκκίνησης αλλά όταν θα έρθει η στιγμή όπου η χρονική αυτή περίοδος θα τελειώσει θα περαστούν και ημερομηνίες τερματισμού.

ΕΠΙΛΟΓΗ ΜΟΡΦΗΣ ΕΞΟΔΟΥ

Εδώ επιλέγουμε πως θα είναι διατεταγμένη η βάση την οποία θα εξάγουμε. Δεν θα είναι της παραπάνω μορφής αλλά κάπως έτσι.

```
<tf
id="A"><name>A</name><value>TRUE</value><Starting_Da
te>28.10.09</Starting_Date><Starting_Time>10:34:03</Startin
g_Time></tf>
```

Δεν υπάρχει ούτε ένα κενό μεταξύ τους, αυτή η μορφή είναι πιο «συμπαγής». Επιλέγουμε αυτόν τον τρόπο γιατί το αρχείο το οποίο θα εξάγουμε τώρα θα το χρησιμοποιήσουμε στην συνέχεια για να το επανεισάγουμε και να το επεξεργαστούμε ξανά προκειμένου να τελειοποιηθεί.

ΕΞΑΓΩΓΗ ΑΡΧΕΙΩΝ

Σε αυτό το τελευταίο βήμα εξάγουμε τα αρχεία μας. Θα δημιουργηθεί είναι αρχείο με το όνομα makingXml.xml το οποίο θα περιέχει όλη την βάση μας, και θα δημιουργηθούν άλλα δυο αρχεία τα A.xml και B.xml τα οποία περιέχουν μόνο τους κόμβους A και B αντίστοιχα τα οποία θα εξηγήσουμε αργότερα που θα μας χρειαστούν.
Όλα αυτά τα αρχεία δεν είναι ακόμα έτοιμα θα τα επανεισάγουμε αργότερα προκειμένου να τα τελειοποιήσουμε

Result

<?xml version="1.0" encoding="UTF-8"?>

<test id="28.10.09 at 10:34:03"><tf id="A"><name>A</name><value>TRUE</value><Starting_Date>28.10.09</Starting_Date><Starting_Time>10:34:03</Starting_Time></tf><tf id="B"><name>B</name><value>FALSE</value><Starting_Date>28.10.09</Starting_Date><Starting_Time>10:34:03</Starting_Time></tf><tf id="C"><name>C</name><value>FALSE</value><Starting_Date>28.10.09</Starting_Date><Starting_Time>10:34:03</Starting_Time></tf><tf id="D"><name>D</name><value>TRUE</value><Starting_Date>28.10.09</Starting_Date><Starting_Time>10:34:03</Starting_Time></tf><tf id="E"><name>E</name><value>FALSE</value><Starting_Date>28.10.09</Starting_Date><Starting_Time>10:34:03</Starting_Time></tf></test>

2.2.1.2 Ολοκλήρωση του βασικού μέρους

Source

```
import java.io.*;
import org.w3c.dom.*;



import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;



import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;



public class addingEndTime {


    public static void main (String argv []) throws DOMException{
    try {

            DocumentBuilderFactory docBuilderFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder docBuilder =
docBuilderFactory.newDocumentBuilder();
            Document doc = docBuilder.parse (new
File("c:\\makingXml.xml"));

            ΠΡΟΣΘΕΣΗ ΗΜΕΝΟΜΗΝΙΩΝ ΤΕΡΜΑΤΙΣΜΟΥ

// AAAAAAAAAAAAAAAAAAAAAA
            Node endtimeA = doc.getFirstChild();

            Node DateA = doc.createElement("Ending_Date");
            DateA.setTextContent((DateUtils.now("dd.MM.yy")));
            endtimeA.getFirstChild().appendChild(DateA);
            Node timeA = doc.createElement("Ending_Time");
            timeA.setTextContent((DateUtils.now("hh:mm:ss")));
            endtimeA.getFirstChild().appendChild(timeA);


 //BBBBBBBBBBBBBBBBBBB



            Node endtimeB = doc.getFirstChild();

            Node DateB = doc.createElement("Ending_Date");
            DateB.setTextContent((DateUtils.now("dd.MM.yy")));
```

```
endtimeB.getFirstChild().getNextSibling().appendChild(DateB);
            Node timeB = doc.createElement("Ending_Time");
            timeB.setTextContent((DateUtils.now("hh:mm:ss")));

endtimeB.getFirstChild().getNextSibling().appendChild(timeB);


//CCCCCCCCCCCCCCCCCCCCCC
            Node endtimeC = doc.getFirstChild();

            Node DateC = doc.createElement("Ending_Date");
            DateC.setTextContent((DateUtils.now("dd.MM.yy")));

endtimeC.getFirstChild().getNextSibling().getNextSibling().appendChil
d(DateC);
            Node timeC = doc.createElement("Ending_Time");
            timeC.setTextContent((DateUtils.now("hh:mm:ss")));

endtimeC.getFirstChild().getNextSibling().getNextSibling().appendChil
d(timeC);


//DDDDDDDDDDDDDDDDDdd


            Node endtimeD = doc.getFirstChild();

            Node DateD = doc.createElement("Ending_Date");
            DateD.setTextContent((DateUtils.now("dd.MM.yy")));

endtimeD.getFirstChild().getNextSibling().getNextSibling().getNextSib
ling().appendChild(DateD);
            Node timeD = doc.createElement("Ending_Time");
            timeD.setTextContent((DateUtils.now("hh:mm:ss")));

endtimeD.getFirstChild().getNextSibling().getNextSibling().getNextSib
ling().appendChild(timeD);


 ///EEEEEEEEEEEEEEEEE

            Node endtimeE = doc.getFirstChild();

            Node DateE = doc.createElement("Ending_Date");
            DateE.setTextContent((DateUtils.now("dd.MM.yy")));
            endtimeE.getLastChild().appendChild(DateE);
            Node timeE = doc.createElement("Ending_Time");
            timeE.setTextContent((DateUtils.now("hh:mm:ss")));
            endtimeE.getLastChild().appendChild(timeE);
```

ΔΗΜΙΟΥΡΓΙΑ ΤΕΛΙΚΟΥ ΑΡΧΕΙΟΥ

```
            Transformer transformer =
TransformerFactory.newInstance().newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");

            //initialize StreamResult with File object to save to
file
            StreamResult result = new StreamResult(new
StringWriter());
```

```
            DOMSource source = new DOMSource(doc);
            transformer.transform(source, new StreamResult(new


            FileOutputStream("C:\\CompleteXml.xml")));

            String xmlString = result.getWriter().toString();
            System.out.println(xmlString);


    }catch (SAXParseException err) {
        System.out.println ("** Parsing error" + ", line " +
err.getLineNumber () + ", uri " + err.getSystemId ());
        System.out.println(" " + err.getMessage ());

        }catch (SAXException e) {
        Exception x = e.getException ();
        ((x == null) ? e : x).printStackTrace ();

        }catch (Throwable t) {
        t.printStackTrace ();
        }
        //System.exit (0);

    }//end of main


}
```

- **Explain**

ΠΡΟΣΘΕΣΗ ΗΜΕΝΟΜΗΝΙΩΝ ΤΕΡΜΑΤΙΣΜΟΥ

Εδώ πλέον παίρνοντας σαν είσοδο το προηγούμενο αρχείο που δουλέψαμε(**makingXml.xml**) αρχίζουμε και εισάγουμε τις ημερομηνίες τερματισμού σε όλους τους κόμβους. Η διαδικασία έχει ως εξής, όταν αποφασίσουμε ότι θέλουμε να αλλάξουμε την τιμή κάποιου κόμβου πρέπει να λήξει η χρονική η περίοδος πρώτα που κόμβος είχε την προηγούμενη τιμή προτού την αλλάξουμε. Έτσι όταν θέλουμε να το κάνουμε αυτό τρέχουμε την κλάση **addingEndtime.java** και εκείνη πάει και περνάει σε όλους τους κόμβους την τρέχουσα ημερομηνία και ώρα. Έτσι ώστε να έχουμε μια ολοκληρωμένη μορφή για ποια χρονικά διαστήματα οι κόμβοι είχαν συγκεκριμένες τιμές. Η μορφή τους θα είναι κάπως έτσι:

```
<tf id="A">
<name>A</name>
<value>TRUE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:14</Ending_Time>
</tf>
```

ΔΗΜΙΟΥΡΓΙΑ ΤΕΛΙΚΟΥ ΑΡΧΕΙΟΥ

Εδώ εξάγουμε τώρα το τελικό αρχείο **completeXml.xml** με τα αποτελέσματα και τα διαστήματα για τις τιμές των κόμβων. Λόγο της ανάγκης για τη συνεχή μεταβολή τιμών ανά τακτά χρονικά διαστήματα θεώρησα απαραίτητο την δημιουργία της βάσης μας σε αυτά τα δυο βήματα. Ας δούμε λοιπόν τώρα την τελική της μορφή.

- **Result**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<test id="28.10.09 at 10:34:03">
<tf id="A">
<name>A</name>
<value>TRUE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:14</Ending_Time>
</tf>
<tf id="B">
<name>B</name>
<value>FALSE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:14</Ending_Time>
</tf>
<tf id="C">
<name>C</name>
<value>FALSE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:14</Ending_Time>
</tf>
<tf id="D">
<name>D</name>
<value>TRUE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:14</Ending_Time>
</tf>
```

```
<tf id="E">
<name>E</name>
<value>FALSE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:14</Ending_Time>
</tf>
</test>
```

2.2.1.3 Δημιουργία History File

Source

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.StringWriter;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.DOMException;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;


public class MakingHistory {
	public static void main (String argv []) throws DOMException{
		try {

                        ΕΙΣΑΓΩΓΗ ΤΩΝ ΑΡΧΕΙΩΝ «Α» ΚΑΙ «Β»
                       DocumentBuilderFactory docBuilderFactory =
DocumentBuilderFactory.newInstance();
                       DocumentBuilder docBuilder =
docBuilderFactory.newDocumentBuilder();
Document doc = docBuilder.parse (new File("c:\\A.xml"));
Document docB = docBuilder.parse (new File("c:\\B.xml"));

                    ΠΡΟΣΘΕΣΗ ΗΜΕΝΟΜΗΝΙΩΝ ΤΕΡΜΑΤΙΣΜΟΥ
            // AAAAAAAAAAAAAAAAAAAAAA
Node endtimeA = doc.getFirstChild();

Node DateA = doc.createElement("Ending_Date");

DateA.setTextContent((DateUtils.now("dd.MM.yy")));
endtimeA.appendChild(DateA);
Node timeA = doc.createElement("Ending_Time");

timeA.setTextContent((DateUtils.now("hh:mm:ss")));
endtimeA.appendChild(timeA);

             ////BBBBBBBBBBBB

Node endtimeB = docB.getFirstChild();

Node DateB = docB.createElement("Ending_Date");

DateB.setTextContent((DateUtils.now("dd.MM.yy")));
endtimeB.appendChild(DateB);
Node timeB = docB.createElement("Ending_Time");
```

```
timeB.setTextContent((DateUtils.now("hh:mm:ss")));
endtimeB.appendChild(timeB);
```

ΔΗΜΙΟΥΡΓΙΑ ΤΩΝ HISTORY FILES

```
Transformer transformer =
TransformerFactory.newInstance().newTransformer();

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

//initialize StreamResult with File object to save to file
StreamResult result = new StreamResult(new StringWriter());
DOMSource source = new DOMSource(doc);
transformer.transform(source, new StreamResult(new

FileOutputStream("C:\\Ahis.xml",true)));

String xmlString = result.getWriter().toString();
                        System.out.println(xmlString);


//-----------------------------------B HISTORY

 StreamResult resultB = new StreamResult(new StringWriter());
DOMSource sourceB = new DOMSource(docB);
transformer.transform(sourceB, new StreamResult(new
FileOutputStream("C:\\Bhis.xml",true)));

String xmlStringB = resultB.getWriter().toString();
 System.out.println(xmlStringB);
 //---------------------------------- END B HISTORY


                }catch (SAXParseException err) {
                    System.out.println ("** Parsing error" + ", line
" + err.getLineNumber () + ", uri " + err.getSystemId ());
                    System.out.println(" " + err.getMessage ());

                    }catch (SAXException e) {
                    Exception x = e.getException ();
                    ((x == null) ? e : x).printStackTrace ();

                    }catch (Throwable t) {
                    t.printStackTrace ();
                    }
                    //System.exit (0);

                }//end of main



}
```

- **Explain**

ΕΙΣΑΓΩΓΗ ΤΩΝ ΑΡΧΕΙΩΝ «Α» ΚΑΙ «Β»

Εδώ παίρνουμε τα αρχεία Α και Β που είχαμε δημιουργήσει από την κλάση **makingXml.java.** Τα αρχεία αυτά περιέχουν μόνο τους κόμβους που έχουν όνομα Α και Β αντίστοιχα. Αυτό το κάνουμε σε μια προσπάθεια να δημιουργήσουμε ένα history file. Δηλαδή κάθε φορά που ο κόμβος με το όνομα Α (αλλά και οποιοσδήποτε άλλος κόμβος) αλλάζει τιμή και κατέπεκταση χρονικά διαστήματα θα αποθηκεύεται σε ένα ξεχωριστό αρχείο όπου θα περιέχει το ιστορικό του. Αυτό θα μας εξυπηρετήσει στην συνέχεια να κάνουμε τα ερωτήματα έχοντας όλες τις τιμές με όλα τα χρονικά διαστήματα σε ένα μόλις αρχείο. Σε αντιστοιχία με το παράδειγμα που είχα δώσει στην αρχή φανταστείτε κάθε εργαζόμενος να είχε το δικό του αρχείο με τη εξέλιξη της απόδοσης του και της θέσης εργασίας του.

ΠΡΟΣΘΕΣΗ ΗΜΕΝΟΜΗΝΙΩΝ ΤΕΡΜΑΤΙΣΜΟΥ

Σε αυτό βήμα περνιούνται αυτόματα οι ημερομηνίες τερματισμού σε κάθε αρχείο χωριστά όπως ακριβώς κάναμε και πριν.

ΔΗΜΙΟΥΡΓΙΑ ΤΩΝ HISTORY FILES

Τέλος εξάγουμε τα αρχεία **Ahis.xml** και **Bhis.xml** τα οποία εν τέλει είναι αυτά που θα χρησιμοποιήσουμε και παρακάτω. Περιέχουν το ιστορικό για κάθε στοιχειό σε ξεχωριστό αρχείο.

- **Result**

Ahis.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tf id="A">
<name>A</name>
<value>TRUE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:18</Ending_Time>
</tf>

<tf id="A">
<name>A</name>
<value>TRUE</value>
<Starting_Date>29.10.09</Starting_Date>
<Starting_Time>07:18:40</Starting_Time>
<Ending_Date>29.10.09</Ending_Date>
<Ending_Time>07:18:46</Ending_Time>
</tf>
```

```
<tf id="A">
<name>A</name>
<value>FALSE</value>
<Starting_Date>29.10.09</Starting_Date>
<Starting_Time>07:23:27</Starting_Time>
<Ending_Date>29.10.09</Ending_Date>
<Ending_Time>07:23:33</Ending_Time>
</tf>
```

Bhis.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<tf id="B">
<name>B</name>
<value>FALSE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:18</Ending_Time>
</tf>

<tf id="B">
<name>B</name>
<value>FALSE</value>
<Starting_Date>29.10.09</Starting_Date>
<Starting_Time>07:18:40</Starting_Time>
<Ending_Date>29.10.09</Ending_Date>
<Ending_Time>07:18:46</Ending_Time>
</tf>

<tf id="B">
<name>B</name>
<value>TRUE</value>
<Starting_Date>29.10.09</Starting_Date>
<Starting_Time>07:23:27</Starting_Time>
<Ending_Date>29.10.09</Ending_Date>
<Ending_Time>07:23:33</Ending_Time>
</tf>
```

3. Υποβολή ερωτημάτων (Xquery)

3.1 Ποιος είναι ο στόχος μας

Έχοντας πλέον δημιουργήσει τα διάφορα XML αρχεία μας είμαστε έτοιμοι πλέον να υποβάλουμε τα ερωτήματα μας στην βάση και να δούμε τι αποτελέσματα θα πάρουμε. Ποιο συγκεκριμένα θα εκτελέσουμε το ερώτημα που μας ενδιαφέρει.

--Suspended(p,t)^good employee(p,t)→take bonus(p,t)

Δηλαδή με τα δικά μας δεδομένα

A(true,t)^B(true,t)→C(true,t)

Το οποίο σημαίνει ότι θα πάρουμε κάθε ένα history file χωριστά (του Α και του Β) και θα ψάξουμε να δούμε αν για μια συγκεκριμένη χρονική στιγμή το Α και το Β είναι **ταυτόχρονα** αληθής.
Τότε και μόνο τότε θα είναι και το C αληθής για το συγκεκριμένο χρονικό διάστημα.
Το ερώτημα θα γίνει με την βοήθεια της **Exist.** Πάμε λοιπόν να δούμε πως θα το εκτελέσουμε.

3.2 Υποβολή ερωτημάτων

Έχουμε τα παρακάτω history files:

Ahis.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<test>
<tf id="A">
<name>A</name>
<value>TRUE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:18</Ending_Time>
</tf>

<tf id="A">
<name>A</name>
<value>TRUE</value>
<Starting_Date>29.10.09</Starting_Date>
<Starting_Time>07:18:40</Starting_Time>
<Ending_Date>29.10.09</Ending_Date>
<Ending_Time>07:18:46</Ending_Time>
```

```
</tf>

<tf id="A">
<name>A</name>
<value>FALSE</value>
<Starting_Date>29.10.09</Starting_Date>
<Starting_Time>07:23:27</Starting_Time>
<Ending_Date>29.10.09</Ending_Date>
<Ending_Time>07:23:33</Ending_Time>
</tf>
<tf id="A">
<name>A</name>
<value>TRUE</value>
<Starting_Date>02.11.09</Starting_Date>
<Starting_Time>08:27:47</Starting_Time>
<Ending_Date>02.11.09</Ending_Date>
<Ending_Time>08:27:53</Ending_Time>
</tf>
</test>
```

Bhis.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<test>
<tf id="B">
<name>B</name>
<value>FALSE</value>
<Starting_Date>28.10.09</Starting_Date>
<Starting_Time>10:34:03</Starting_Time>
<Ending_Date>28.10.09</Ending_Date>
<Ending_Time>10:34:18</Ending_Time>
</tf>

<tf id="B">
<name>B</name>
<value>FALSE</value>
<Starting_Date>29.10.09</Starting_Date>
<Starting_Time>07:18:40</Starting_Time>
<Ending_Date>29.10.09</Ending_Date>
<Ending_Time>07:18:46</Ending_Time>
</tf>

<tf id="B">
<name>B</name>
<value>TRUE</value>
<Starting_Date>29.10.09</Starting_Date>
<Starting_Time>07:23:27</Starting_Time>
<Ending_Date>29.10.09</Ending_Date>
<Ending_Time>07:23:33</Ending_Time>
</tf>
```

```
<tf id="B">
<name>B</name>
<value>TRUE</value>
<Starting_Date>02.11.09</Starting_Date>
<Starting_Time>08:27:47</Starting_Time>
<Ending_Date>02.11.09</Ending_Date>
<Ending_Time>08:27:53</Ending_Time>
</tf>
</test>
```

Εκτελούμε το παρακάτω ερώτημα:

```
for $Atf in (doc("Ahis.xml")/test/tf),
$Btf  in doc("Bhis.xml")/test/tf
where (($Atf/value="TRUE" and $Btf/value="TRUE")
and ($Atf/Starting_Time>="08:27" and $Btf/Starting_Time>="08:27")
and ($Atf/Starting_Date="02.11.09" and $Btf/Starting_Date="02.11.09") )
return "C is TRUE"
```

Εκτελώντας αυτό το ερώτημα το μόνο που έχουμε είναι κάθε φορά να βάζουμε τις ημερομηνίες που μας ενδιαφέρει και να βλέπουμε αν το αποτέλεσμα είναι αληθής. Στο παράδειγμα μας μόνο σε αυτές τι χρονικές στιγμές είναι το C αληθής. Σε οποιαδήποτε άλλη περίπτωση δεν ισχύει.

Βιβλιογραφία

Η πηγές για την ανάπτυξη της εισαγωγής είναι οι παρακάτω:
(Ο χαρακτηρισμός με * και με δείκτη έναν αριθμό δείχνει σε πια βιβλιογραφία αναφέρεται)

1) Ιατρική Πληροφορική Διδάσκων: Δ.Ι. Φωτιάδης
2) Louvari_ Xquery
3) Παρουσίαση xml και jdom Πανεπηστιμιο Αθηνών
4) http://el.wikiversity.org/wiki/
5) http://users.uom.gr/~kaklaman/book/Chapters/C8/What%20is%20XML%202.htm
6) http://www.w3schools.com/xml/default.asp
7) XQuery *Priscilla Walmsley*

8) The Ramification Problem in Temporal Databases: A Solution Implemented in SQL
   Nikos Papadakis, Dimitris Plexousakis, Grigoris Antoniou, Manolis Daskalakis, and Yannis Christodoulou

Η πηγές για την ανάπτυξη του κώδικα είναι οι παρακάτω:

http://www.javaworld.com/javaworld/jw-07-2000/jw-0728-jdom2.html?page=3
http://forums.datadirect.com/ddforums/thread.jspa?messageID=7643&tstart=0
http://www.ibm.com/developerworks/library/x-xjavaxquery/
http://www.petefreitag.com/item/445.cfm
http://labe.felk.cvut.cz/~xfaigl/mep/xml/java-xml.htm
http://www.daniweb.com/forums/thread115931.html#
http://www.kodejava.org/examples/21.html
http://www.developertutorials.com/tutorials/java/read-xml-file-in-java-050611/page1.html
http://stackoverflow.com/questions/33262/how-do-i-load-an-org-w3c-dom-document-from-xml-in-a-string
https://students.kiv.zcu.cz/doc/oracle/appdev.102/b14252/adx_j_parser.htm
http://www.oracle.com/technology/pub/articles/dev2arch/2006/02/load-save-dom.xml.html
http://www.javaworld.com/jw-12-2000/jw-1229-dates.html?page=2
http://onjava.com/pub/a/onjava/2005/01/12/xpath.html
http://www.netbeans.org/kb/docs/java/gui-functionality.html
http://java.sun.com/developer/JDCTechTips/2004/tt1201.html
http://answers.yahoo.com/question/index?qid=20081107235048AAK2A64
http://www.rgagnon.com/javadetails/java-0106.html
http://www.java2s.com/Code/Java/XML/JavaDOMeditLocateaNodeandChangeItsContent.htm
http://www.javapractices.com/topic/TopicAction.do?Id=42
http://nikojava.wordpress.com/tag/xml/
http://www.xml.com/pub/a/2004/08/18/xstream.html
http://www.objectdefinitions.com/odblog/2007/converting-java-dates-to-xml-schema-datetime-with-timezone/