Ημιδομημένες ΒΔ - XML

- Εισαγωγή
- Η δομή των XML δεδομένων
- Οργάνωση / διαχείριση XML δεδομένων
- Ερωτήσεις σε XML δεδομένα
- Αποθήκευση XML δεδομένων σε Σχεσιακές ΒΔ

Βασική πηγή διαφανειών: Silberschatz et al., “Database System Concepts”, 4/e
Εργαστήριο Πληροφοριακών Συστημάτων, Παν/μιο Πειραιώς (http://infolab.cs.unipi.gr/)
έκδοση: Δεκέμβριος 2007

- Εισαγωγή
- Η δομή των XML δεδομένων
- Οργάνωση / διαχείριση XML δεδομένων
- Ερωτήσεις σε XML δεδομένα
- Αποθήκευση XML δεδομένων σε Σχεσιακές ΒΔ

Εισαγωγή

- XML: Extensible Markup Language
- Ορίστηκε από το WWW Consortium (W3C) ως συμπλήρωμα της HTML.
  - Δεν σχεδιάστηκε για βάσεις δεδομένων αλλά για διαχείριση εγγράφων
  - Ενώ η HTML (Hyper-Text Markup Language) περιγράφει την αναπαράσταση των δεδομένων, η XML περιγράφει το περιεχόμενο, π .χ ., tag: <people> </people>
    - Για τον ορισμό της αναπαράστασης μπορεί να χρησιμοποιηθεί ένα ξεχωριστό stylesheet σε μια ειδική γλώσσα την XSL.
  - Σε αντίθεση με την HTML, η XML είναι επεκτάσιμη
    - Οι χρήστες μπορούν να προσθέτουν νέα tags και να καθορίζουν πως πρέπει να αντιμετωπίζονται τα tags κατά την εμφάνισή τους.
  - Ο στόχος ήταν η XML να αντικαταστήσει την HTML ως γλώσσα για την δημοσίευση ιστοσελίδων στο Internet.

Εισαγωγή (συν.)

- **Markup:** οποιοδήποτε τμήμα ενός document το οποίο δεν αποτελεί τμήμα της εκτυπωμένης εξόδου.
- **Markup language:** μια αυστηρά τυπική περιγραφή σχετικά με το ποιο είναι το περιεχόμενο του κειμένου, ποιο το markup και τι σημαίνει.
- Σε σύγκριση με τις βάσεις δεδομένων η XML μπορεί να μην είναι τόσο αποδοτική αφού έχουμε επανάληψη των tags, ωστόσο η XML είναι ιδανική για την ανταλλαγή δεδομένων μεταξύ εφαρμογών
  - Η κυρίως χρήση της XML αφορά την ανταλλαγή δεδομένων μεταξύ εφαρμογών
  - Τα tags είναι αυτοπεριγραφικά.

Εισαγωγή (συν.)

- Παράδειγμα

```
<bank>
    <account>
      <account-number>  A-101     </account-number>
      <branch-name>      Downtown </branch-name>
      <balance>           500        </balance>
    </account>
    <depositor>
      <account-number>  A-101    </account-number>
      <customer-name>   Johnson </customer-name>
    </depositor>
  </bank>
```


Γιατί χρειάζεται η XML;

- Η ανταλλαγή δεδομένων είναι εξαιρετικά σημαντική σήμερα
  - Π.χ.: Τραπεζικά δεδομένα: μεταφορές κεφαλαίου ...
  - Η έντυπη διακίνηση πληροφορίας μεταξύ των εταιριών αντικαθίσταται από την ηλεκτρονική διακίνηση.
- Κάθε τομέας εφαρμογών έχει το δικό του σύνολο από standards για την αναπαράσταση της πληροφορίας.
- Η XML έχει γίνει η βάση όλων των νέας γενιάς standards για την ανταλλαγή των δεδομένων.
- Ένα μεγάλο σύνολο εργαλείων διατίθεται για την ανάλυση (parsing), την δυνατότητα πλοήγησης (browsing) και τον καθορισμό ερωτήσεων (querying) πάνω στα XML δεδομένα.

Θέματα σχετικά με τη χρήση της XML

- Οργάνωση των XML δεδομένων
- Διαχείριση των XML δεδομένων
- Ερωτήσεις πάνω στα XML δεδομένα

Η δομή των XML δεδομένων

- Το βασικό στοιχείο σε ένα XML document είναι το element
- **Element (στοιχείο)**: ένα τμήμα δεδομένων ανάμεσα σε δύο matching tags (<*tagname*> </*tagname*>).
- Τα Elements πρέπει να είναι κατάλληλα εμφωλευμένα.
  - Σωστή εμφώλευση
    - <account> … <balance> …. </balance> </account>
  - Λανθασμένη εμφώλευση
    - <account> … <balance> …. </account> </balance>
  - Τυπικά: κάθε tag αρχής πρέπει να έχει ένα μοναδικό και "ταιριαστό" tag τέλους.

Παράδειγμα εμφωλευμένων Elements

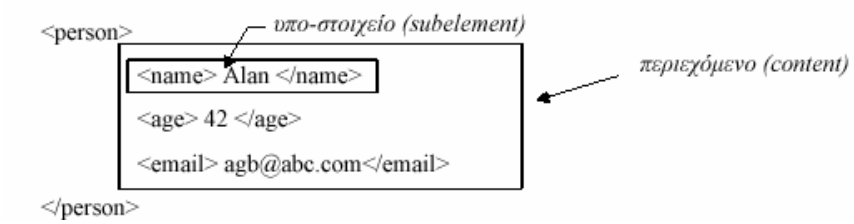
```
<bank-1>
    <customer>
      <customer-name> Hayes </customer-name>
      <customer-street> Main </customer-street>
      <customer-city>    Harrison </customer-city>
      <account>
        <account-number> A-102 </account-number>
        <branch-name>     Perryridge </branch-name>
        <balance>          400 </balance>
      </account>
      <account>
          …
      </account>
    </customer>
      .
      .
  </bank-1>
```


Γιατί χρειάζεται η εμφώλευση;

- Δεν υποστηρίζεται αλλά ούτε και απαγορεύεται η χρήση της στις σχεσιακές βάσεις δεδομένων
  - Η κανονικοποίηση αντικαθιστά τις εμφωλευμένες δομές σε κάθε σειρά με ένα ξένο κλειδί σε ένα πίνακα που περιέχει το customer name και πληροφορίες για την customer address.
  - Η εμφώλευση υποστηρίζεται στις αντικειμενοστραφείς – σχεσιακές βάσεις δεδομένων.
- Κατάλληλο για τη μεταφορά δεδομένων
  - Οι εξωτερικές εφαρμογές δεν έχουν άμεση πρόσβαση στα δεδομένα που αναφέρονται από κάποιο ξένο κλειδί.

Η δομή των XML δεδομένων (Συν.)

- Στην XML επιτρέπεται ο συνδυασμός κειμένου με υπο-στοιχεία (subelements).



Γνωρίσματα *(attributes)*

- Τα elements μπορούν να έχουν **γνωρίσματα**, τα οποία ορίζονται ως ζεύγη (όνομα, τιμή) μέσα στο tag αρχής του element.
  - <account **acct-type** = "checking" >
    <account-number> A-102 </account-number>
    <branch-name> Perryridge </branch-name>
    <balance> 400 </balance>
    </account>
- Τα γνωρίσματα είναι strings, και δεν περιέχουν markup πληροφορία
- Τα γνωρίσματα θα πρέπει να εμφανίζονται μία μόνο φορά σε ένα tag, σε αντίθεση με τα sub-elements που μπορούν να επαναλαμβάνονται.
  - <account **acct-type** = "checking" **monthly-fee**="5">

Γνωρίσματα vs. subelements

- Το όνομα ενός γνωρίσματος είναι μοναδικό σε κάθε tag, ενώ τα ονόματα των subelements δεν είναι.
  - Τα γνωρίσματα αποτελούν μέρος του markup (άρα δεν εμφανίζονται στο τελικό αποτέλεσμα), ενώ τα sub-elements αποτελούν μέρος των περιεχομένων του document.
  - Προσφέρουν εναλλακτικούς τρόπους αναπαράστασης δεδομένων.
    - <person>
      <name> Alan </name>
      <age> 42 </age>
      <email> agb@abc.com</email>
      </person>
    - <person name="Alan" age="42" email= "agb@abc.com" />
  - Υπόδειξη: χρησιμοποιήστε τα γνωρίσματα για την αναγνώριση των elements (as identifiers), και χρησιμοποιήστε τα sub-elements για τα περιεχόμενα.

Namespaces

- Εφόσον η XML χρησιμοποιείται για την ανταλλαγή δεδομένων μεταξύ επιχειρήσεων θα πρέπει να υπάρχει ένας μηχανισμός για τον καθορισμό καθολικά μοναδικών ονομάτων για τα element tags των document.
  - Καθορίζοντας ένα μοναδικό string ως το όνομα κάποιου element αποφεύγεται η σύγχυση.
- Λύση: συσχέτισε κάθε tag ή element με ένα μοναδικό καθολικό identifier, π.χ. μια web σελίδα
  - **unique-name:element-name**
- Π.χ. Αν η First Bank ήθελε να εξασφαλίσει ότι τα XML documents της δεν θα είχαν ίδια tags με τα XML documents κάποιας άλλης επιχείρησης θα μπορούσε να χρησιμοποιήσει ένα url ως εξής:
- <bank Xmlns:FB='http://www.FirstBank.com'>
  …
  <FB:branch>
  <FB:branchname>Downtown</FB:branchname>
  <FB:branchcity> Brooklyn </FB:branchcity>
  </FB:branch> … </bank>

XML Document Schema

- Οι βάσεις δεδομένων έχουν σχήματα που καθορίζουν ποια δεδομένα μπορούν να αποθηκευτούν στη βάση και τους τύπους δεδομένων των αποθηκευμένων τιμών.
- Τα XML documents μπορούν να δημιουργηθούν χωρίς να σχετίζονται με κάποιο σχήμα.
  - ελευθερία (ένα element μπορεί να έχει οποιοδήποτε subelement ή γνώρισμα)
  - τι γίνεται στην περίπτωση που τα XML documents πρέπει να επεξεργαστούν αυτόματα ως τμήμα μιας εφαρμογής;
- Τα σχήματα είναι σημαντικά για την ανταλλαγή δεδομένων στην XML.
- Υπάρχουν δύο μηχανισμοί για τον καθορισμό ενός XML σχήματος
  - **Document Type Definition (DTD)**
    - Ήδη καθιερωμένο, χρησιμοποιείται ευρέως.
  - **XML Schema**
    - Νεότερο, με αυξανόμενη χρήση.

Document Type Definition (DTD)

- Περιγράφει μια κλάση από XML documents χρησιμοποιώντας μια γλώσσα που είναι ουσιαστικά μια γραμματική χωρίς συμφραζόμενα με αρκετούς περιορισμούς.
- Το DTD περιορίζει τη δομή των XML δεδομένων
  - Ποια elements μπορούν να υπάρχουν
  - Ποια γνωρίσματα μπορεί/ πρέπει να έχει ένα element
  - Ποια subelements μπορεί/ πρέπει να υπάρχουν μέσα σε ένα element, και πόσες φορές.
- Το DTD δεν περιορίζει τους τύπους δεδομένων π.χ. integer
  - Όλες οι τιμές αναπαρίστανται ως strings.
- Το DTD είναι μια λίστα από κανόνες σχετικά με τα subelements και τα γνωρίσματα που μπορούν να υπάρχουν μέσα σε ένα element.
- DTD syntax
  - <!ELEMENT element (subelements-specification) >
  - <!ATTLIST element (attributes) >

Παράδειγμα: Bank DTD

```
<!DOCTYPE bank [
    <!ELEMENT bank ( ( account | customer | depositor)+)>
    <!ELEMENT account (account-number branch-name balance)>
    <! ELEMENT customer(customer-name customer-street
                                                customer-city)>
    <! ELEMENT depositor (customer-name account-number)>
    <! ELEMENT account-number (#PCDATA)>
    <! ELEMENT branch-name (#PCDATA)>
    <! ELEMENT balance(#PCDATA)>
    <! ELEMENT customer-name(#PCDATA)>
    <! ELEMENT customer-street(#PCDATA)>
    <! ELEMENT customer-city(#PCDATA)>
]>
```


Καθορισμός elements στο DTD

- Τα subelements μπορούν να καθοριστούν ως
  - ονόματα elements, ή
  - #PCDATA, π.χ., character strings
    - PCDATA (parsed character data) υποδηλώνει κείμενο
  - EMPTY (το element δεν έχει περιεχόμενο)
  - ANY (δεν υπάρχει περιορισμός ως προς τα subelements που μπορεί να έχει το element)
    - Αν δεν υπάρχει δήλωση για κάποιο element υποδηλώνεται το any.

Καθορισμός elements στο DTD (συν.)

- Παράδειγμα
  - <! ELEMENT depositor (customer-name account-number)>
  - <! ELEMENT customer-name (#PCDATA)>
  - <! ELEMENT account-number (#PCDATA)>
- Ένα subelement μπορεί να περιέχει κανονικές εκφράσεις (regular expressions)
  - <!ELEMENT bank ((account | customer | depositor)+)>
    - Σημειογραφία:
      - "|" - εναλλακτικές επιλογές
      - "+" - >=1 εμφανίσεις
      - "*" - >=0 εμφανίσεις

IDs και IDREFs

- Το **ID** γνώρισμα αποτελεί μοναδικό identifier για ένα element (object identifier).
- Η τιμή του ID ενός element θα πρέπει να είναι μοναδική στο XML document.
- Σε ένα element το πολύ ένα γνώρισμα επιτρέπεται να είναι τύπου ID.
- Το **IDREF** γνώρισμα είναι μια αναφορά σε ένα element και πρέπει να περιέχει μια τιμή που εμφανίζεται σε κάποιο ID γνώρισμα των elements του document.
- Ένα IDREF γνώρισμα περιέχει ένα σύνολο από (>=0) τιμές ID γνωρισμάτων.

Bank DTD με γνωρίσματα

- Το DTD της τράπεζας με γνωρίσματα τύπου ID και IDREF.

```
<!DOCTYPE bank-2[
  <!ELEMENT account (branch, balance)>
  <!ATTLIST account
        account-number ID          # REQUIRED
        owners             IDREFS # REQUIRED>
  <!ELEMENT customer(customer-name, customer-street,
                                                customer-city)>
  <!ATTLIST customer
        customer-id        ID          # REQUIRED
        accounts           IDREFS # REQUIRED>
  … declarations for branch, balance, customer-name,
                        customer-street and customer-city
]>
```


XML δεδομένα με γνωρίσματα ID και IDREF

```
<bank-2>
    <account account-number="A-401" owners="C100 C102">
        <branch-name> Downtown </branch-name>
        <balance>        500 </balance>
    </account>
    <customer customer-id="C100" accounts="A-401">
        <customer-name>Joe        </customer-name>
        <customer-street> Monroe  </customer-street>
        <customer-city>    Madison</customer-city>
    </customer>
    <customer customer-id="C102" accounts="A-401 A-402">
        <customer-name> Mary     </customer-name>
        <customer-street> Erin       </customer-street>
        <customer-city>    Newark </customer-city>
    </customer>
</bank-2>
```

Περιορισμοί των DTD's

- Δεν υπάρχουν τύποι στα elements κειμένου και τα γνωρίσματα
  - Όλες οι τιμές είναι strings, δεν υπάρχουν integers, reals, κ.α.
- Είναι δύσκολο να καθοριστούν μη ταξινομημένα σύνολα από subelements.
  - Η ταξινόμηση δεν υπάρχει συνήθως στις βάσεις δεδομένων
  - Η έκφραση (A | B)* επιτρέπει τον καθορισμό ενός μη ταξινομημένου συνόλου, αλλά
    - Δεν εγγυάται ότι κάθε ένα από τα A και B εμφανίζεται μία μόνο φορά
- Τα IDs και IDREFs δεν έχουν τύπους
  - Στο παράδειγμα δεν υπάρχει τρόπος να καθοριστεί ο τύπος του element στο οποίο αναφέρεται ένα γνώρισμα τύπου ID ή IDREFS.
    - Το γνώρισμα *owners* θα πρέπει (ιδανικά) να αναφέρεται στα elements του customer.

XML Schema

- Πρόκειται για μια σύνθετη **schema language** που αντιμετωπίζει τα μειονεκτήματα των DTDs. Υποστηρίζει
  - "Κλασσικούς" τύπους δεδομένων
    - Π.χ. integer, string, κλπ.
    - περιορισμούς στις min/max τιμές
  - Τύπους δεδομένων ορισμένους από το χρήστη
  - Καθορίζεται και η ίδια στην σύνταξη της XML, σε αντίθεση με τα DTDs
  - Είναι ενσωματωμένη με τα namespaces
  - Έχει πολλά ακόμη χαρακτηριστικά
    - List types, περιορισμούς μοναδικότητας και ξένου κλειδιού, κληρονομικότητα ...
- ΟΜΩΣ είναι σημαντικά πιο πολύπλοκη από τα DTDs, και δεν χρησιμοποιείται ακόμη ευρέως.

XML Schema του Bank DTD

```
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema>
<xsd:element name="bank" type="BankType"/>
<xsd:element name="account">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="account-number" type="xsd:string"/>
         <xsd:element name="branch-name"    type="xsd:string"/>
         <xsd:element name="balance"        type="xsd:decimal"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
..... definitions of customer and depositor ....
<xsd:complexType name="BankType">
   <xsd:sequence>
      <xsd:element ref="account"   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="customer" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="depositor" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```


Ερωτήσεις και μετατροπές σε XML δεδομένα

- Αναφερόμαστε σε:
  - Μετατροπή πληροφορίας από ένα XML schema σε ένα άλλο.
  - Ερωτήσεις πάνω σε XML δεδομένα
- Τα δύο παραπάνω σχετίζονται άμεσα, και αντιμετωπίζονται με τα ίδια εργαλεία
- Standard XML querying/ translation languages
  - **XPath** - Απλή γλώσσα που χρησιμοποιεί εκφράσεις μονοπατιού.
  - **XSLT** - Απλή γλώσσα που σχεδιάστηκε για μετάφραση από XML σε XML και από XML σε HTML
  - **XQuery** - Μια XML query language με πολλά χαρακτηριστικά που προτάθηκε ως standard για ερωτήσεις πάνω σε XML δεδομένα.

Το δεντρικό μοντέλο των XML δεδομένων

- Οι γλώσσες ερωτήσεων και μετατροπών στηρίζονται σε ένα **δεντρικό μοντέλο** από XML δεδομένα.
- Ένα XML document μοντελοποιείται ως **δέντρο**, οι **κόμβοι** του οποίου αντιστοιχούν στα elements και τα γνωρίσματα.
  - Οι κόμβοι των elements έχουν παιδιά, τα οποία μπορεί να είναι γνωρίσματα ή subelements.
  - Το κείμενο ενός element μοντελοποιείται ως ένα κόμβος κειμένου που είναι παιδί του element.
  - Τα παιδιά ενός κόμβου διατάσσονται με βάση τη σειρά τους στο XML document
  - Οι κόμβοι των elements και των γνωρισμάτων (εκτός της ρίζας) έχουν ένα πατέρα, ο οποίος είναι element.
  - Η ρίζα έχει ένα παιδί, το οποίο είναι το βασικό element του document.
- Χρησιμοποιούνται όροι όπως: κόμβος, παιδί, πατέρας, αδερφός κόμβος, πρόγονος, απόγονος, κ.α., οι οποίοι θα πρέπει να διερμηνευτούν με βάση το παραπάνω δεντρικό μοντέλο των XML δεδομένων.

XPath

- Χρησιμοποιείται για την επιλογή τμημάτων ενός XML document χρησιμοποιώντας **εκφράσεις μονοπατιού**.
- Μία έκφραση μονοπατιού (path expression) είναι μια ακολουθία βημάτων που διαχωρίζονται μέσω του "/"
  - Κάτι αντίστοιχο με τα όνομα αρχείου στην ιεραρχία του μονοπατιού
- Το αποτέλεσμα μιας έκφρασης μονοπατιού: ένα σύνολο τιμών.
- Π.χ. η έκφραση μονοπατιού (για το παράδειγμα bank-2 data)
  - /bank-2/customer/customer-name
  - επιστρέφει τα εξής:
  - <customer-name>Joe</customer-name>
  - <customer-name>Mary</customer-name>
- Ενώ η έκφραση μονοπατιού
  - /bank-2/customer/customer-name/text()
- επιστρέφει τα ίδια ονόματα χωρίς όμως τα περιβάλλοντα tags.

XPath (συν.)

- Το αρχικό "/" συμβολίζει τη ρίζα του document
- Οι εκφράσεις μονοπατιού **αποτιμούνται** από αριστερά προς τα δεξιά.
  - Κάθε βήμα ενεργεί στο σύνολο των στιγμιοτύπων που παράχθηκε από το προηγούμενο βήμα.
- Οι συνθήκες του selection μπορούν να μπουν σε οποιοδήποτε βήμα σε ένα μονοπάτι, μέσα σε [ ].
  - Π.χ. /bank-2/account[balance > 400]
    - Επιστρέφει τα elements account με balance>400.
  - /bank-2/account[balance]
    - επιστρέφει τα elements account με που περιέχουν υπόλοιπο (balance) ως subelement.
- Οι τιμές των γνωρισμάτων προσπελαύνονται μέσω του "@".
  - Π.χ. /bank-2/account[balance > 400]/@account-number
    - Επιστρέφει τους αριθμούς λογαριασμού με balance>400.

XPath συναρτήσεις

- Υποστηρίζει αρκετές συναρτήσεις
  - Η συνάρτηση **count()** στο τέλος του μονοπατιού μετράει το πλήθος των elements του συνόλου που παράχθηκε από το μονοπάτι.
    - Π.χ. /bank-2/account[customer/count() > 2]
      - Επιστρέφει τα accounts με > 2 πελάτες
  - Επίσης, υποστηρίζει συναρτήσεις που ελέγχουν τη θέση των κόμβων (1, 2, ..).
- Οι λογικές εκφράσεις **and** και **or** και η συνάρτηση **not()** μπορούν να χρησιμοποιηθούν στις συνθήκες.
- Τα IDREFs μπορούν να βρεθούν μέσω της συνάρτησης **id()**
  - Η id() μπορεί να εφαρμοστεί επίσης και σε σύνολα όπως τα IDREFS
  - Π.χ. /bank-2/account/id(@owners)
    - Επιστρέφει όλους τους πελάτες που αναφέρονται από το γνώρισμα owners του account element.

XPath συναρτήσεις (συν.)

- Ο τελεστής **"|"** υλοποιεί την ένωση *(union)*
  - Π.χ. /bank-2/account/id(@owners) **|** /bank-2/loan/id(@borrower)
    - Επιστρέφει τους πελάτες που έχουν είτε accounts είτε loans.
    - Ωστόσο, το "|" δεν μπορεί να εμφωλευτεί μέσα σε άλλους τελεστές.
- Το **"//"** χρησιμοποιείται για να παρακάμψει τα πολλαπλά επίπεδα κόμβων
  - Π.χ. /bank-2//customer-name
    - Βρίσκει κάθε customer-name element *οπουδήποτε* κάτω από το /bank-2 element, και ανεξάρτητα από το element στο οποίο περιέχεται.
- Σε ένα βήμα στο μονοπάτι μπορούμε να μεταβούμε στους: γονείς, αδέρφια, προγόνους και απογόνους των κόμβων που προκύπτουν από το προηγούμενο βήμα και όχι απλά στα παιδιά.
  - **"//"**: αντιστοιχεί στο "όλους τους απογόνους"
  - **".."**: καθορίζει τον πατέρα.

XSLT

- Ένα **stylesheet** αποθηκεύει τον τρόπο μορφοποίησης ενός document - συνήθως αποτελεί ξεχωριστό document.
  - Π.χ. Ένα HTML stylesheet μπορεί να καθορίσει τη γραμματοσειρά, τα χρώματα κ.α.
- Η **XML Stylesheet Language (XSL)** σχεδιάστηκε αρχικά για την αυτόματα παραγωγή HTML από XML.
- Η XML περιλαμβάνει ένα γενικού σκοπού μηχανισμό μετατροπής, που ονομάζεται XSL Transformations (XSLT).
  - Η XSLT μπορεί να μεταφράζει XML σε XML, και XML σε HTML.
- Οι μετατροπές της XSLT εκφράζονται μέσω κανόνων που ονομάζονται **templates**.
  - Τα templates συνδυάζουν τα selections μέσω των XPath με την "κατασκευή" των αποτελεσμάτων.

XSLT Templates

- Π.χ. ενός XSLT template με **match** και **select**

```
<xsl:template match="/bank-2/customer">
    <xsl:value-of select="customer-name"/>
</xsl:template>
<xsl:template match="*"/>
```

- Το γνώρισμα match του xsl:template καθορίζει ένα pattern στο XPath
- Τα elements του XML document που ταιριάζουν με το pattern επεξεργάζονται μέσω των ενεργειών που υπάρχουν μέσα στο xsl:template element.
  - xsl:value-of επιστρέφει καθορισμένες τιμές (εδώ, το customer-name)
- Για τα elements που δεν ταιριάζουν σε κανένα template
  - Τα γνωρίσματα και τα περιεχόμενα κειμένου εξάγονται ως έχουν
  - Τα templates εφαρμόζονται αναδρομικά στα subelements
- Το <xsl:template match="*"/> template περιλαμβάνει όλα τα elements που δεν ταιριάζουν σε κανένα άλλο template

XSLT Templates (συν.)

- Αν κάποιο element ταιριάζει σε παραπάνω από ένα templates, μόνο ένα από αυτά χρησιμοποιείται.
  - Το ποιο, εξαρτάται από σύνθετα σχήματα προτεραιότητας / προτεραιότητες του χρήστη.
  - Υποθέτουμε πως μόνο ένα template ταιριάζει σε κάθε element.

XML έξοδος

- Οποιοδήποτε κείμενο ή tag του XSL stylesheet που δεν ανήκει στο xsl namespace εξάγεται ως έχει.
- Π.χ. για το παράδειγμα:

```
<xsl:template match="/bank-2/customer">
  <customer>
   <xsl:value-of select="customer-name"/>
   </customer>
</xsl:template>
<xsl:template match="*"/>
```

  - η έξοδος είναι:

```
<customer> Joe   </customer>
<customer> Mary </customer>
```


XSLT: ταξινόμηση

- Η χρήση μιας xsl:sort εντολής μέσα σε ένα template έχει ως αποτέλεσμα να ταξινομηθούν όλα τα elements που πληρούν τους κανόνες του template.
- Π.χ.

```
<xsl:template match="/bank">
        <xsl:apply-templates select="customer">
        <xsl:sort select="customer-name"/>
        </xsl:apply-templates>
</xsl:template>
<xsl:template match="customer">
        <customer>
                <xsl:value-of select="customer-name"/>
                <xsl:value-of select="customer-street"/>
                <xsl:value-of select="customer-city"/>
        </customer>
<xsl:template>
<xsl:template match="*"/>
```

XQuery

- Η XQuery είναι μια γενικού σκοπού γλώσσα ερωτήσεων για XML δεδομένα που αναπτύχθηκε από το W3C (World Wide Web Consortium).
- Η XQuery προέρχεται από τη γλώσσα ερωτήσεων Quilt, η οποία με τη σειρά της προέρχεται από τις SQL, XQL και XML-QL.
- Η XQuery χρησιμοποιεί ένα συντακτικό της μορφής FLWR ("flower"):
  for ... let ... where .. result ...
  όπου
  for ⇔ SQL from
  where ⇔ SQL where
  result ⇔ SQL select
  let επιτρέπει προσωρινές μεταβλητές, δεν υπάρχει κάτι αντίστοιχο στην SQL.

Εκφράσεις FLWR στην XQuery

- Απλή FLWR έκφραση στην XQuery
  - Βρες όλους τους λογαριασμούς με balance>400, και κάθε αποτέλεσμα να περικλείεται από ένα <account-number> .. </account-number> tag.

```
for     $x in /bank-2/account
let     $acctno := $x/@account-number
where   $x/balance > 400
return  <account-number> $acctno </account-number>
```

- Η πρόταση Let δεν είναι απαραίτητη στο παραπάνω ερώτημα, η επιλογή μπορεί να γίνει στο XPath. Η ερώτηση μπορεί να γραφτεί ως:

```
for $x in /bank-2/account[balance>400]
return <account-number> $x/@account-number
                        </account-number>
```

Joins

- Καθορίζονται με τρόπο παρόμοιο με αυτό της SQL

```
for $a  in  /bank/account,
    $c  in  /bank/customer,
    $d  in  /bank/depositor
 where   $a/account-number = $d/account-number
   and $c/customer-name = $d/customer-name
 return <cust-acct> $c $a </cust-acct>
```

- Εναλλακτικά με XPath selections:

```
for $a in /bank/account
     $c in /bank/customer
      $d in /bank/depositor[
                account-number = $a/account-number and
                customer-name  = $c/customer-name]
 return <cust-acct> $c $a</cust-acct>
```


XQuery: εκφράσεις μονοπατιού

- **$c/text():** επιστρέφει το κείμενο ενός element χωρίς subelements ή tags.
- Οι εκφράσεις μονοπατιού στην XQUERY υποστηρίζουν τον τελεστή: "–>" για τα IDREFs
  - Ισοδύναμος με τη συνάρτηση id( ) της XPath, αλλά πιο εύχρηστος
  - Μπορεί να εφαρμοστεί σε ένα σύνολο από IDREFs και να επιστρέψει ένα σύνολο αποτελεσμάτων
  - Στην έκδοση June 2001 έγινε αλλαγή από "–>" το "=>"

Αποθήκευση XML Δεδομένων

- Τα XML δεδομένα μπορούν να αποθηκευτούν σε μη-σχεσιακές αποθήκες δεδομένων
  - Απλά αρχεία
    - απλά
    - Αλλά, έχουν όλα τα προβλήματα της αποθήκευσης σε αρχείο (no concurrency, no recovery, …)
  - XML βάσεις δεδομένων
    - Σχεδιασμένες ειδικά για την αποθήκευση XML δεδομένων, να υποστηρίζουν το DOM μοντέλο και ερωτήσεις
    - Υπάρχουν περιορισμένα εμπορικά συστήματα προς το παρόν
- …

Αποθήκευση XML Δεδομένων (συν.)

- … ή μπορούν να αποθηκευτούν σε σχεσιακές βάσεις δεδομένων
  - Τα δεδομένα πρέπει να μετατραπούν σε σχεσιακή μορφή
  - πλεονέκτημα: ώριμα συστήματα βάσεων δεδομένων
  - μειονέκτημα: επιπλέον κόστος για τη μετατροπή των δεδομένων και των queries
  - Εναλλακτικοί τρόποι αποθήκευσης:
    - Αναπαράσταση μέσω string
    - Αναπαράσταση μέσω δέντρων
    - Αντιστοιχία σε σχέσεις

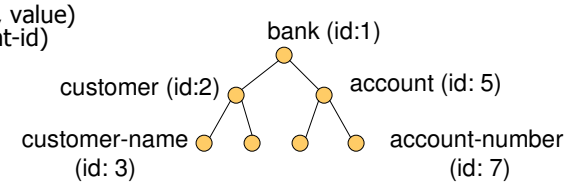
Αναπαράσταση μέσω strings

- Αποθήκευση κάθε element του υψηλότερου επιπέδου σαν ένα string πεδίο μιας πλειάδας της βάσης
  - Χρήση μιας απλής σχέσης για την αποθήκευση όλων των elements, ή
  - Χρήση μιας ξεχωριστής σχέσης για κάθε element type του υψηλότερου επιπέδου
- Πλεονεκτήματα
  - Μπορεί να αποθηκεύσει οποιαδήποτε XML δεδομένα ακόμη και χωρίς DTD.
  - Αν υπάρχουν πολλά elements στο υψηλότερο επίπεδο του document, τα strings είναι μικρά σε σχέση με το πλήρες document
    - Επιτρέπει γρήγορη πρόσβαση στα επιμέρους elements.
- Μειονέκτημα: χρειάζεται να αναλύσουμε συντακτικά (parse) strings για να προσπελάσουμε τις τιμές μέσα στα elements
  - Η ανάλυση αυτή είναι αργή

Αναπαράσταση μέσω δέντρων

- Μοντελοποίηση XML δεδομένων ως δέντρο και αποθήκευσή τους χρησιμοποιώντας σχέσεις

  nodes(id, type, label, value)
  child (child-id, parent-id)



- Σε κάθε element / γνώρισμα δίνεται ένα μοναδικό αναγνωριστικό.
- Η ετικέτα καθορίζει το όνομα του tag του element / το όνομα του γνωρίσματος.
- Η τιμή είναι το κείμενο του element / γνωρίσματος.
- Η σχέση child συμβολίζει τις σχέσεις parent-child στο δέντρο
  - Μπορούμε να προσθέσουμε ένα επιπλέον γνώρισμα στο child για να κρατάμε τη διάταξη των παιδιών.

Αναπαράσταση μέσω δέντρων (συν.)

- Πλεονεκτήματα:
  - Μπορεί να αποθηκεύσει οποιαδήποτε XML δεδομένα ακόμη και χωρίς DTD.
- Μειονεκτήματα:
  - Τα δεδομένα σπάνε σε πάρα πολλά τμήματα, αυξάνοντας την πολυπλοκότητα χώρου.
  - Ακόμα και τα απλά queries απαιτούν ένα μεγάλο αριθμό joins, το οποίο μπορεί να απαιτεί πολύ χρόνο.

Αντιστοιχία σε σχέσεις

- Αν το DTD του document είναι γνωστό, μπορεί να γίνει η αντιστοίχιση των δεδομένων σε σχέσεις.
- Μια σχέση δημιουργείται για κάθε τύπο element.
  - Τα elements (τύπου #PCDATA), και τα γνωρίσματα αντιστοιχίζονται στα γνωρίσματα των σχέσεων.
- Πλεονεκτήματα:
  - Αποδοτική αποθήκευση
  - Δυνατότητες: μετάφρασης XML queries σε SQL, αποδοτικής εκτέλεσης, μετάφρασης των SQL αποτελεσμάτων πίσω σε XML.
- Μειονεκτήματα:
  - Απαιτείται η γνώση του DTD
  - Υπάρχει κόστος μετάφρασης

Αντιστοιχία σε σχέσεις (συν.)

- Η σχέση που δημιουργείται για κάθε τύπο element περιέχει
  - Ένα id γνώρισμα για την αποθήκευση ενός μοναδικού id για κάθε element.
  - Ένα γνώρισμα που αντιστοιχεί σε κάθε γνώρισμα element
  - Ένα parent-id γνώρισμα για την αποθήκευση του element πατέρα
    - Όπως στην αναπαράσταση μέσω δέντρων
    - Μπορεί επίσης να αποθηκευτεί και η θέση (ith child)
- Όλα τα subelements που εμφανίζονται μία μόνο φορά μπορούν να γίνουν γνωρίσματα της σχέσης
  - Για text-valued subelements, αποθήκευση του κειμένου ως τιμή του γνωρίσματος.
  - Για πολύπλοκα subelements, αποθήκευση του id του subelement.
- Τα subelements που εμφανίζονται πολλές φορές αναπαρίστανται σε ξεχωριστό πίνακα
  - Παρόμοια αντιμετώπιση με τη μετατροπή του ER σε σχεσιακό.