ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων

Πανεπιστημιακές Σημειώσεις

Η Γλώσσα Σήμανσης XML

Δημήτριος Σάμψων

Επίκουρος Καθηγητής

- Δεκέμβριος 2003 -

Περιεχόμενα

Πρόλογος

Το παρόν έγγραφο αποτελεί τις Πανεπιστημιακές Σημειώσεις που αναπτύχθηκαν για το μάθημα ΤΕ256 «Αποθήκες και Εξόρυξη Δεδομένων», του Τμήματος Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς. Το μάθημα αυτό διδάσκεται στο 5^{ο} Εξάμηνο του 3^{ου} Έτους στην κατεύθυνση Ψηφιακών Συστημάτων και είναι υποχρεωτικό. Η παρούσα έκδοση των Πανεπιστημιακών Σημειώσεων αναπτύχθηκαν για τις ανάγκες του μαθήματος κατά την διεξαγωγή του στο χειμερινό εξάμηνο του ακαδημαϊκού έτους 2003-2004. Στη συγγραφή των σημειώσεων συμμετείχε ο υποψήφιος διδάκτορας του Τμήματος Κώστας Καστραντάς.

1 Εισαγωγή

1.1 Γλώσσες Σήμανσης

Σήμερα, μεγάλος όγκος πληροφορίας είναι διαθέσιμος στο διαδίκτυο. Είναι αναγκαίο οι πηγές πληροφορίας να είναι εύκολα προσβάσιμες και να υπάρχει δυνατότητα μεταφοράς τους και να είναι ευέλικτες. Επιπλέον, η πληθώρα διαφορετικών συστημάτων υπαγορεύει την ανάγκη τα έγγραφα πληροφορίας να είναι ανεξάρτητα οποιουδήποτε συστήματος και περιεχομένου. Για την επίλυση αυτών των προβλημάτων αναπτύχθηκαν οι γλώσσες σήμανσης.

Μία γλώσσα σήμανσης (markup language) είναι ένα σύνολο εντολών που επιτέπουν την προσθήκη πληροφορίας στο περιεχόμενο μιας πηγής δεδομένων. Το περιεχόμενο μπορεί να είναι κείμενο, εικόνα ή οποιαδήποτε άλλη μορφή ηλεκτρονικής πληροφορίας. Οι εντολές των γλωσσών σήμανσης είναι απλές και κατανοητές. Οι εντολές αυτές ονομάζονται ετικέτες (tags).

Η SGML (Standard Generalized Markup Language), αποτελεί το διεθνές πρότυπο (ISO 8879) γενικευμένης γλώσσας σήμανσης. Καθορίζει τις μεθόδους αναπαράστασης πληροφορίας, οι οποίες είναι ανεξάρτητες από οποιοδήποτε σύστημα ή μηχάνημα. Οι μέθοδοι αυτοί είναι καθορισμένοι ώστε να είναι κατανοητοί από ανθρώπους και μηχανές. Όλες οι γλώσσες σήμανσης οι οποίες συμμορφώνονται με τους κανόνες που καθορίζει η SGML είναι εφαρμογές της. Επιπλέον, η SGML είναι μία μετα-γλώσσα: παρέχει τη δυνατότητα καθορισμού νέων γλωσσών σήμανσης που είναι υποσύνολά της. Αν και η SGML έχει ουσιαστικό ρόλο στον καθορισμό εγγράφων σύμφωνα με τις προδιαγραφές του προτύπου επικοινωνίας, είναι πολύπλοκη για τους σκοπούς του διαδικτύου.

Μια εφαρμογή της SGML είναι η HTML (HyperText Markup Language). Η HTML αποτελεί χαρακτηριστικό παράδειγμα γλώσσας σήμανσης. Τα HTML έγγραφα είναι ανεξάρτητα αρχεία, τα οποία περιέχουν μια σειρά από εντολές σήμανσης, τις ετικέτες. Η σύνταξη με βάση μια γλώσσα σήμανσης αποτελείται από στοιχεία (elements) τα οποία περιέχουν άλλα στοιχεία, ιδιότητες (attributes) και περιεχόμενο. Για παράδειγμα <HTML> είναι η ετικέτα του βασικού στοιχείου ενός HTML εγγράφου. Όλο το περιεχόμενο ενός HTML εγγράφου περιέχεται ανάμεσα στις ετικέτες αρχής (<HTML>) και τέλους (</HTML>). Οι ιδιότητες τροποποιούν τα HTML στοιχεία. Αποτελούνται από ένα ζευγάρι όνομα-τιμή: το όνομα της ιδιότητας, ένα σύμβολο ισότητας και την τιμή της ιδιότητας. Το σύνολο των εντολών (ετικέτες) της HTML είναι αυστηρά καθορισμένο και σε πολλές περιπτώσεις δεν είναι αρκετό για τις αυξανόμενες απαιτήσεις του διαδικτύου.

Η XML (eXtensible Markup Language) είναι μια γλώσσα σήμανσης που σχεδιάστηκε για να ικανοποιήσει πολλαπλές ανάγκες στις οποίες η HTML αδυνατεί να δώσει λύση. Δίνει στα έγγραφα μεγαλύτερο επίπεδο προσαρμοστικότητας στο στυλ και τη δομή από αυτό που παρέχει η HTML. Η XML αποτελεί υποσύνολο της SGML και απευθύνεται στη σήμανση εγγράφων που περιέχουν δομημένες πληροφορίες. Έχει καθορισμένη δομή σύνταξης αλλά δεν αποτελείται από προκαθορισμένες ετικέτες. Επομένως η XML δεν είναι μόνο μια γλώσσα σήμανσης αλλά είναι και αυτή μια μετα-γλώσσα που χρησιμοποιείται για να καθορίσει νέες γλώσσες σήμανσης.

1.2 Η γλώσσα σήμανσης XML

Το 1990, δημιουργήθηκε η HTML από τους Tim Berners-Lee και Anders Berglund, οι οποίοι καθόρισαν έναν τύπο SGML εγγράφου, για έγγραφα υπερκειμένου, το οποίο ήταν συναφές και αποτελεσματικό. Είναι πολύ εύκολο να υλοποιηθεί λογισμικό που να υποστηρίζει HTML, ενώ η δημιουργία ενός HTML εγγράφου είναι ακόμα πιο εύκολη. Για το λόγο αυτό η HTML χρησιμοποιείται ευρέως.

Για να επιτευχθεί η απαραίτητη απλότητα από την HTML, μερικές από τις βασικές αρχές γενικού προγραμματισμού θυσιάστηκαν. Για παράδειγμα ένας τύπος εγγράφου χρησιμοποιείται για όλους τους σκοπούς με αποτέλεσμα να είναι απαραίτητη η υπερβολική χρήση ετικετών αντί για τον καθορισμό νέων με συγκεκριμένο σκοπό. Επιπλέον οι περισσότερες από τις ετικέτες σκοπεύουν μόνο στη διαμόρφωση της εμφάνισης του εγγράφου. Για την επίλυση αυτών των προβλημάτων έγινε προσπάθεια υιοθέτησης της SGML για χρήση στο διαδίκτυο, αλλά κάτι τέτοιο αποδείχτηκε πολύπλοκο και πολύ δύσκολο. Επομένως ήταν αναγκαίος ο καθορισμός μιας νέας γλώσσας, η οποία να είναι υποσύνολο της SGML αλλά να διατηρεί τα βασικά χαρακτηριστικά της. Η γλώσσα αυτή είναι η XML. Οι σχεδιαστικοί στόχοι της XML είναι οι εξής:

- να είναι εύχρηστη στο Internet
- να υποστηρίζει πλήθος εφαρμογών
- να είναι συμβατή με την SGML
- ευκολία ανάπτυξης προγραμμάτων που επεξεργάζονται XML έγγραφα
- ο αριθμός των προαιρετικών χαρακτηριστικών στην XML να είναι όσο το δυνατόν πιο μικρός, ιδανικό επίπεδο το μηδέν.
- τα XML έγγραφα θα πρέπει να είναι ευανάγνωστα
- ο σχεδιασμός XML θα πρέπει να προετοιμάζεται γρήγορα
- ο σχεδιασμός XML θα πρέπει να είναι τυπικός και περιεκτικός
- τα XML έγγραφα θα πρέπει να δημιουργούνται εύκολα
- η περιεκτικότητα στον XML συμβολισμό είναι μικρής σημασίας

Τα κύρια χαρακτηριστικά της XML είναι:

- παρέχει ακριβή καθορισμό του περιεχομένου, ώστε τα αποτελέσματα μιας αναζήτησης να είναι ικανοποιητικά.
- παρέχει δυνατότητα καθορισμού των στοιχείων του εγγράφου (π.χ. Τίτλος, Συγγραφέας, Εκδότης …)
- επιτρέπει τη μεταφορά μέσω διαδικτύου, δομημένης πληροφορίας ανεξάρτητα από πλατφόρμες και συστήματα

Η XML συμπληρώνει και δεν αντικαθιστά την HTML. Ενώ η HTML χρησιμοποιείται για διατύπωση και εμφάνιση των δεδομένων η XML αναπαριστά τη συναφή έννοια των δεδομένων. Στην HTML οι ετικέτες είναι προκαθορισμένες ενώ η XML παρέχει τη δυνατότητα να καθορίζουν οι χρήστες τις ετικέτες και τις δομημένες μεταξύ τους σχέσεις. Επομένως, ο χρήστης με τη βοήθεια της XML μπορεί να δημιουργήσει μια νέα γλώσσα σήμανσης καθορίζοντας ένα νέο σύνολο από ετικέτες ή να χρησιμοποιήσει ένα σύνολο από ετικέτες το οποίο είναι καθορισμένο από κάποιον άλλο. Αυτό έχει ως αποτέλεσμα να υπάρχει απεριόριστος αριθμός γλωσσών σήμανσης που προέρχονται από την XML, και για το λόγο αυτό η XML είναι μια μετα-γλώσσα. Τα πλεονεκτήματα της γλώσσας σήμανσης XML είναι τα εξής:

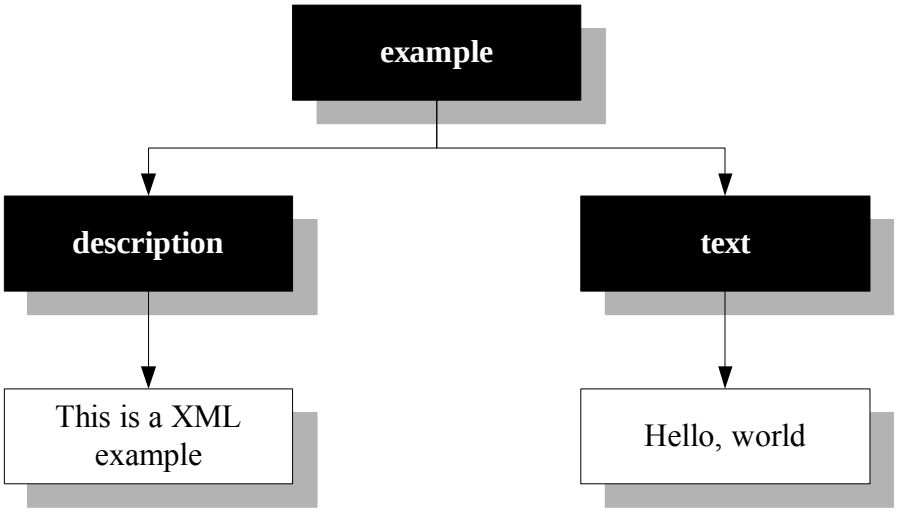
- Πιο ικανοποιητικές αναζητήσεις
- Ανάπτυξη ευέλικτων εφαρμογών
- Ενσωμάτωση δεδομένων από ανόμοιες πηγές πληροφορίας
- Δεδομένα από πολλαπλές εφαρμογές
- Υπολογισμός και διαχείριση δεδομένων τοπικά
- Πολλαπλοί τρόποι εμφάνισης των δεδομένων
- Ενημέρωση πεδίων

- Διαθέσιμα Πρότυπα
- Μορφοποίηση εμφάνισης για το διαδίκτυο

Η γλώσσα προγραμματισμού XML περιγράφει μια κατηγορία πληροφοριών που καλούνται *XML έγγραφα* (*documents*) καθώς επίσης περιγράφει τμηματικά τη συμπεριφορά των προγραμμάτων που τα επεξεργάζονται. Τα XML έγγραφα είναι αυστηρά δομημένα και μπορούν να αναπαρασταθούν με δενδρική δομή. Το πρώτο στοιχείο του εγγράφου, το οποίο περικλείει και όλα τα άλλα, ονομάζεται «ρίζα» (root). Κάθε στοιχείο που περιέχει στοιχεία ή δεδομένα ονομάζεται «πατέρας» (parent), ενώ κάθε στοιχείο ή δεδομένο το οποίο περιέχεται σε κάποιο άλλο στοιχείο ονομάζεται «παιδί» (child). Κάθε στοιχείο ή δεδομένο πρέπει να ανήκει σε κάποιο άλλο ή να είναι παιδί της ρίζας. Ένα παράδειγμα XML εγγράφου είναι το παρακάτω:

```
<?xml version="1.0"?>
<example>
      <text> Hello, world!</text>
      <description>This is a XML example</description>
</example>
```

Το παράδειγμα αυτό μπορεί να αναπαρασταθεί με την εξής δενδρική μορφή:



Εικόνα 1-1 Η δενδρική μορφή ενός XML εγγράφου

Για τη δημιουργία XML εγγράφων μπορεί να χρησιμοποιηθεί οποιαδήποτε εργαλείο συγγραφής ή επεξεργασίας κειμένου (π.χ. Microsoft Notepad, Unix vi, Apple SimpleText κλπ). Επιπλέον, υπάρχουν διαθέσιμα πολλά εργαλεία επεξεργασίας XML εγγράφων τα οποία παρέχουν πλεονεκτήματα όπως διαφορετικά χρώματα εμφάνισης των ετικετών, απόκρυψη των ετικετών και εμφάνιση μόνο του περιεχομένου τους κλπ. Επίσης πολλά από αυτά τα εργαλεία παρέχουν μηχανισμούς ελέγχου εγκυρότητας και ορθής μορφοποίησης των εγγράφων, δύο χαρακτηριστικά που αναφέρονται παρακάτω.

1.3 Μοντελοποίηση εγγράφων

Ένα από τα σημαντικότερα πλεονεκτήματα της XML είναι η δυνατότητα που παρέχει για καθορισμό νέων γλωσσών σήμανσης, καθορίζοντας στοιχεία και ιδιότητες τα οποία περιγράφουν με τον καλύτερο τρόπο συγκεκριμένο είδος πληροφορίας. Για τον πλήρη και τυπικό καθορισμό μιας νέας γλώσσας σήμανσης, για τον περιορισμό του συνόλου των στοιχείων και των ιδιοτήτων της και για τον έλεγχο της γραμματικής και της σύνταξης των ετικετών χρησιμοποιούνται τα *μοντέλα εγγράφων* (*document models*).

Τα μοντέλα εγγράφων καθορίζουν ποια XML έγγραφα συμμορφώνονται με τη γλώσσα σήμανσης XML. Τα μοντέλα εγγράφων απαντούν σε ερωτήματα του τύπου: «αυτό το στοιχείο έχει τίτλο;» ή «το στοιχείο πρέπει να έχει τιμή;» κλπ. Το μοντέλο είναι ένα έγγραφο γραμμένο με βάση ένα συντακτικό που έχει καθοριστεί για να περιγράφει XML γλώσσες, και ορίζει μονοσήμαντα τη γραμματική και το σύνολο ετικετών της γλώσσας σήμανσης. Η γλώσσα σήμανσης που καθορίζεται από το μοντέλο ονομάζεται τύπος εγγράφου ή εφαρμογή XML.

Η χρήση ενός μοντέλου εγγράφου είναι προαιρετική καθώς η XML έχει σχεδιαστεί έτσι ώστε να μην είναι απαραίτητο ένα τέτοιο έγγραφο και η χρήση του εξαρτάται αποκλειστικά από την εφαρμογή και τον δημιουργό της. Η χρήση όμως των μοντέλων εγγράφων έχει σημαντικά πλεονεκτήματα:

- Οι προκαθορισμένοι κανόνες καθιστούν πιο εύκολη τη δημιουργία λογισμικού για τα έγγραφα και ελαχιστοποιούν την πιθανότητα λαθών.
- Όταν το έγγραφο απαιτεί συγκεκριμένα στοιχεία αυτά μπορούν να καθοριστούν με τη βοήθεια του μοντέλου
- Το μοντέλο καθοδηγεί τον αδαή χρήστη σε ένα περιβάλλον συγγραφής XML εγγράφων, παρέχοντας στο χρήστη μια λίστα από τα απαραίτητα στοιχεία.

Τα μειονεκτήματα της χρήσης των μοντέλων εγγράφων είναι η δυσκολία διαχείρισης, η καθυστέρηση κατά την επεξεργασία και η παρεμπόδιση δημιουργίας νέων στοιχείων καθώς το μοντέλο καθορίζει ποια στοιχεία μπορούν ή όχι να χρησιμοποιηθούν.

Οι πιο δημοφιλείς τύποι μοντέλων εγγράφων είναι:

(i) ορισμός τύπου εγγράφων (Document Type Definition - DTD)

(ii) XML Σχήματα (XML Schemas)

Ένα DTD καθορίζει ένα τύπο εγγράφου ως εξής:

Καθορίζει το σύνολο των επιτρεπτών στοιχείων. Δεν επιτρέπεται η χρήση άλλων ονομάτων για τα στοιχεία από αυτά που καθορίζονται από το παραπάνω σύνολο. Το σύνολο στοιχείων ονομάζεται και «λεξικό» της γλώσσας σήμανσης.

Καθορίζει το μοντέλο περιεχομένου κάθε εγγράφου. Το μοντέλο περιεχομένου είναι ένα υπόδειγμα το οποίο καθορίζει ποιο στοιχείο ή δεδομένο μπορεί να είναι μέρος ενός στοιχείου, με ποια σειρά, πόσες φορές και αν είναι υποχρεωτικό ή προαιρετικό. Το μοντέλο περιεχομένου ονομάζεται και «γραμματική» της γλώσσας σήμανσης.

Καθορίζει ένα σύνολο επιτρεπτών ιδιοτήτων για κάθε στοιχείο. Κάθε ορισμός ιδιότητας καθορίζει το όνομα, τον τύπο δεδομένων, τις προκαθορισμένες τιμές (εάν υπάρχουν) και την συμπεριφορά (υποχρεωτική ή προαιρετική) της ιδιότητας.

Παρέχει ένα πλήθος μηχανισμών για την εύκολη διαχείριση του μοντέλου, για παράδειγμα χρήση οντοτήτων (entities), εισαγωγή τμημάτων από εξωτερικά αρχεία κλπ.

Εναλλακτικά μπορεί να χρησιμοποιηθεί το μοντέλο εγγράφου XML Σχήμα. Τα DTD παρουσιάζουν μειονεκτήματα όπως η περιορισμένη δυνατότητα καθορισμού τύπων δεδομένων και η διαφορετική σύνταξη του DTD από τα XML έγγραφα. Τα XML Σχήματα δίνουν λύση στους παραπάνω περιορισμούς. Τα σημαντικότερα πλεονεκτήματα των XML σχημάτων σε σχέση με τα DTD είναι:

- Υποστηρίζουν περισσότερους από 44 τύπους δεδομένων και επιτρέπουν τη δημιουργία καινούριων
- Έχουν την ίδια σύνταξη με ένα XML έγγραφο
- Είναι αντικειμενοστραφείς (για παράδειγμα επιτρέπουν την επέκταση ή τον περιορισμό ενός τύπου δεδομένων)

- Επιτρέπουν την δημιουργία συνόλων (δημιουργία στοιχείων χωρίς να είναι σημαντική η σειρά εμφάνισής τους στο έγγραφο)
- Επιτρέπουν τον καθορισμό περιεχομένου ενός στοιχείου ως μοναδικό (key)
- Επιτρέπουν τον καθορισμό πολλαπλών στοιχείων με το ίδιο όνομα και διαφορετικό περιεχόμενο, χωρίς να δημιουργείται σύγχυση κατά την επεξεργασία

Η επιλογή ανάμεσα στα δύο μοντέλα εγγράφων είναι θέμα χρήστη και εφαρμογής. Τα XML Σχήματα είναι μεταγενέστερα και παρουσιάζουν σημαντικά πλεονεκτήματα έναντι των DTD, αλλά δεν τα αντικαθιστούν. Τα DTD πλεονεκτούν σε μέγεθος, έχουν γνώριμη και απλή σύνταξη στην οποία πολλοί είναι ήδη εξοικειωμένοι. Μερικές εφαρμογές της XML, καθορισμένες με DTD ή με XML Σχήμα είναι:

- **CML** (**C**hemical **M**arkup **L**anguage): Γλώσσα σήμανσης για την περιγραφή πολύπλοκων μορίων που χρησιμοποιείται από χημικούς
- **MathML** (**M**athematical **M**arkup **L**anguage): Γλώσσα σήμανσης για την περιγραφή εξισώσεων στο διαδίκτυο
- **VML** (**V**ector **M**arkup **L**anguage): Γλώσσα σήμανσης για τον σχεδιασμό γραφικών που περιγράφονται με διανύσματα
- **RDF** (**R**esource **D**escription **F**ramework): Γλώσσα σήμανσης για την αναπαράσταση μεταδεδομένων.

1.4 XML έγγραφα

Υπάρχουν δυο τύποι XML εγγράφων : τα ορθά μορφοποιημένα ή καλά ορισμένα (well-formed) και τα έγκυρα (valid). Ένα καλά ορισμένο XML έγγραφο ακολουθεί τους γενικούς κανόνες σύνταξης της XML, οι οποίοι είναι πιο αυστηροί από αυτούς της HTML και της SGML. Μερικοί από τους κανόνες αυτούς είναι:

- Οι χαρακτήρες δεδομένων της XML έχουν πάντα μία ετικέτα τέλους οποιουδήποτε είδους: είτε όπως το ζεύγος <ετικέτα></ετικέτα>, είτε μια ετικέτα κενού στοιχείου όπως <ετικέτα/>.
- Η σήμανση της XML ξεκινάει πάντοτε με το σύμβολο < ή με το σύμβολο &.
- Οι τύποι των στοιχείων και τα ονόματα των εισαγωγικών αναγνωρίζουν πεζά και κεφαλαία.
- Οι τιμές των ιδιοτήτων απαιτούν εισαγωγικά

Τα έγκυρα XML έγγραφα ακολουθούν ένα μοντέλο εγγράφου, δηλαδή ένα σετ κανόνων που ένα έγγραφο ακολουθεί, και τους οποίους το λογισμικό έχει την ικανότητα να διαβάσει πριν την επεξεργασία και την εμφάνιση του εγγράφου. Αυτοί οι κανόνες γενικά καθορίζουν το όνομα και το περιεχόμενο κάθε στοιχείου (element). Τα έγκυρα XML έγγραφα προσφέρουν πολύ περισσότερα στην επεξεργασία του εγγράφου από τα αντίστοιχα ορθά μορφοποιημένα. Η συγγραφή, επεξεργασία, αποθήκευση, και εμφάνιση του εγγράφου έγιναν ευκολότερες γιατί το έγγραφο υπάρχει σε ένα δομικό περιβάλλον.

Η βασική διαφορά μεταξύ των ορθά μορφοποιημένων και των έγκυρων XML εγγράφων είναι η σχέση τους με το μοντέλο εγγράφου. Οι δημιουργοί των XML εγγράφων πρέπει να συντάσσουν έγκυρα έγγραφα, ενώ ένα ικανό λογισμικό πλοήγησης XML (XML browsers) χρειάζεται μόνο να ελέγχει το έγγραφο για ορθή μορφοποίηση, πριν το εμφανίσει.

1.5 Επεξεργασία XML εγγράφων

Ένα λογισμικό μοντέλο που καλείται *επεξεργαστής XML* (*XML processor*) χρησιμοποιείται για να διαβάζει XML έγγραφα και παρέχει πρόσβαση στο περιεχόμενο και τη δομή τους. Ο επεξεργαστής XML λειτουργεί εκ μέρους ενός άλλου μοντέλου που καλείται *εφαρμογή*

(*application*). Αυτή η προδιαγραφή περιγράφει την απαιτούμενη συμπεριφορά του επεξεργαστή και συγκεκριμένα πως θα πρέπει να διαβάζει τα XML δεδομένα και ποιες πληροφορίες πρέπει να παρέχει στην εφαρμογή. Παραδείγματα XML επεξεργαστών συμπεριλαμβάνουν εφαρμογές ελέγχου εγκυρότητας, περιηγητές (web browsers), εργαλεία συγγραφής και επεξεργασίας XML εγγράφων (XML Editors, XML management tools) κλπ.

Ένας *αναλυτής XML* (*XML parser*) είναι ένα λογισμικό μοντέλο το οποίο διαβάζει XML αρχεία, ελέγχει την ορθή μορφοποίησή τους και την εγκυρότητά τους και τα προωθεί για περαιτέρω επεξεργασία. Ο αναλυτής XML είναι η πιο απλή μορφή ενός XML επεξεργαστή και είναι απαραίτητο τμήμα κάθε XML εφαρμογής. Ο αναλυτής μετατρέπει την χωρίς νόημα ακολουθία χαρακτήρων του XML εγγράφου σε τμήματα (tokens) που έχουν νόημα. Τα κουπόνια είτε ερμηνεύονται ως γεγονότα (events) τα οποία καθοδηγούν ένα πρόγραμμα είτε δημιουργούν μία προσωρινή δομή στη μνήμη (δενδρική αναπαράσταση) πάνω στην οποία το πρόγραμμα μπορεί να επέμβει.

1.6 Δομή των σημειώσεων

Στο παρών έγγραφο εξετάζουμε τη Γλώσσα Σήμανσης XML. Στα επόμενα κεφάλαια διαπραγματευόμαστε τα παρακάτω θέματα:

Στο κεφάλαιο 2 γίνεται εισαγωγή στη σύνταξη της XML. Πιο συγκεκριμένα, μελετάται η σύνταξη των βασικών μονάδων ενός XML εγγράφου (στοιχεία, ιδιότητες, οδηγίες επεξεργασίας, οντότητες) και παρουσιάζονται οι βασικοί συντακτικοί κανόνες για τη δημιουργία καλά ορισμένων XML έγγραφων. Το κεφάλαιο ολοκληρώνεται με τέσσερεις ασκήσεις εξοικίωσης με θέματα σύνταξης της XML.

Στο κεφάλαιο 3 μελετάται η μοντελοποίηση των XML εγγράφων. Συγκεκριμένα, εξετάζονται οι δύο πιο δημοφιλείς τεχνολογίες μοντελοποίησης των XML εγγράφων, τα DTDs και τα XML Schemas. Μελετώνται οι τεχνικές μέσω των οποίων ορίζονται οι βασικές δομικές μονάδες των XML εγγράφων (στοιχεία, ιδιότητες) όπως και οι τύποι δεδομένων που παρέχονται μέσω των τεχνολογιών μοντελοποίησης. Στο τέλος του κεφαλαίου 3 παρουσιάζεται μια σειρά ασκήσεων, που αποτελούν επέκταση των ασκήσεων που παρουσιάστηκαν στο κεφάλαιο 2, δίνοντας τη δυνατότητα στον αναγνώστη να κατανοήσει καλύτερα τον τρόπο μοντελοποίησης των XML εγγράφων.

Στο κεφάλαιο 4 μελετώνται οι χώροι ονοματοδοσίας και πιο συγκεκριμένα τα προβλήματα που έρχονται να επιλύσουν καθώς και ο τρόπος με τον οποίο δηλώνονται στα XML έγγραφα. Το κεφάλαιο αυτό κλείνει με την παράθεση ορισμένων απλών παραδειγμάτων χρήσης των χώρων ονοματοδοσίας καθώς και με μια άσκηση που ασχολείται με τη δήλωση χώρων ονοματοδοσίας σε XML έγγραφα, το μοντέλο των οποίων καθορίζεται από ένα XML Schema.

Στο κεφάλαιο 5 γίνεται μια εισαγωγή στο μετασχηματισμό XML εγγράφων με χρήση της τεχνολογίας XSLT. Συγκεκριμένα, αφού γίνει μια εισαγωγή στο μετασχηματισμό εγγράφων, μελετώνται οι δηλώσεις που απαιτούνται για την επεξεργασία των κόμβων ενός XML εγγράφου, παρουσιάζεται η γλώσσα XPath 1.0 όπως επίσης μια σειρά από στοιχεία για την περαιτέρω επεξεργασία των XML εγγράφων. Στο τέλος παρατίθονται μια σειρά από παραδείγματα για να γίνει πιο κατανοητή η γλώσσα XPath, όπως επίσης και μια απλή άσκηση μετασχηματισμού ενός XML εγγράφου.

Τέλος, στο κεφάλαιο 6,παρουσιάζονται ορισμένοι βασικοί κανόνες για την αναπαράσταση μιας Βάσης Δεδομένων σε ένα XML έγγραφο, καθώς και η XQuery που αποτελεί μια γλώσσα επερώτησης XML δεδομένων. Το κεφάλαιο 6 κλείνει με την παρουσίαση 2 ασκήσεων που στοχεύουν στην εξοικίωση του αναγνώστη με τη γλώσσα XQuery.

2 Η σύνταξη της XML

2.1 Η ανατομία XML εγγράφων

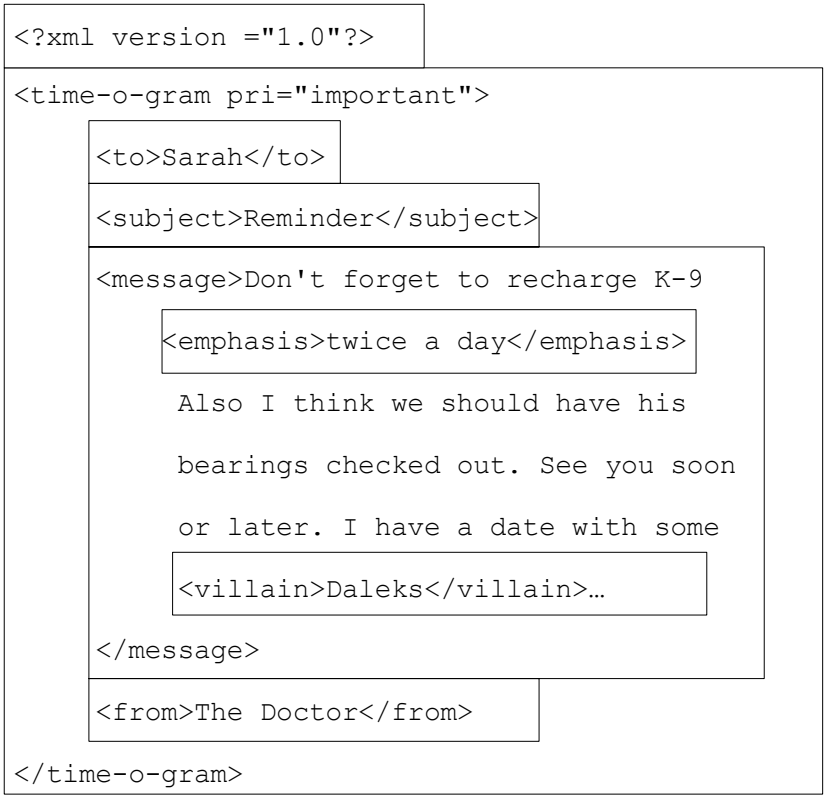
Η XML παρέχει ένα σύνολο κανόνων για τη δημιουργία σημασιολογικών ετικετών (tags) που χρησιμοποιούνται για να περιγράψουν τα δεδομένα. Ένα στοιχείο (element) της XML αποτελείται από μια ετικέτα έναρξης και μια ετικέτα τέλους με δεδομένα ενδιάμεσα. Οι ετικέτες περιγράφουν τα δεδομένα τα οποία αποτελούν την τιμή ενός στοιχείου. Το παρακάτω παράδειγμα απεικονίζει ένα απλό XML έγγραφο το οποίο αναπαριστά ένα υπόμνημα.

```
<?xml version =”1.0”>
<time-o-gram pri=”important” >
     <to>Sarah</to>
     <subject>Reminder</subject>
     <message>Don’t forget to recharge K-9
         <emphasis>twice a day</emphasis>
          Also I think we should have his
          bearings checked out. See you soon
          or later. I have a date with some
          <villain>Daleks</villain>…
     </message>
     <from>The Doctor</from>
</time-o-gram>
```

Το παραπάνω παράδειγμα απεικονίζει ένα απλό XML έγγραφο που ακολουθεί τους συντακτικούς κανόνες της XML. Η XML επιτρέπει τον καθορισμό και τη χρήση οποιαδήποτε ετικέτας (αρκεί να ακολουθούνται οι βασικοί κανόνες σύνταξής της), εν αντιθέσει με την HTML, η οποία θέτει περιορισμούς λόγω της χρήσης προκαθορισμένων ετικετών. Ωστόσο, ο τρόπος με τον οποίο οργανώνεται η πληροφορία αποτελεί ένα σημαντικό κομμάτι της XML. Η επιλογή των ονομάτων των στοιχείων θα πρέπει να γίνεται με τέτοιο τρόπο που η ανάγνωση ενός XML εγγράφου να είναι κατανοητή είτε από έναν χρήστη είτε από έναν υπολογιστή.

Το παράδειγμα αυτό, όπως και όλα τα XML έγγραφα, αποτελείται από σύμβολα σήμανσης. Οι αγκύλες (<>) και τα ονόματα που περικλείουν ονομάζονται ετικέτες (tags). Οι ετικέτες οριοθετούν και ονομάζουν τα μέρη του εγγράφου, επίσης προσθέτουν επιπλέον πληροφορίες. Το κείμενο μεταξύ των ετικετών είναι το περιεχόμενο του εγγράφου, απλές πληροφορίες που μπορούν να είναι ένας τίτλος, ή ένα πεδίο πληροφοριών.

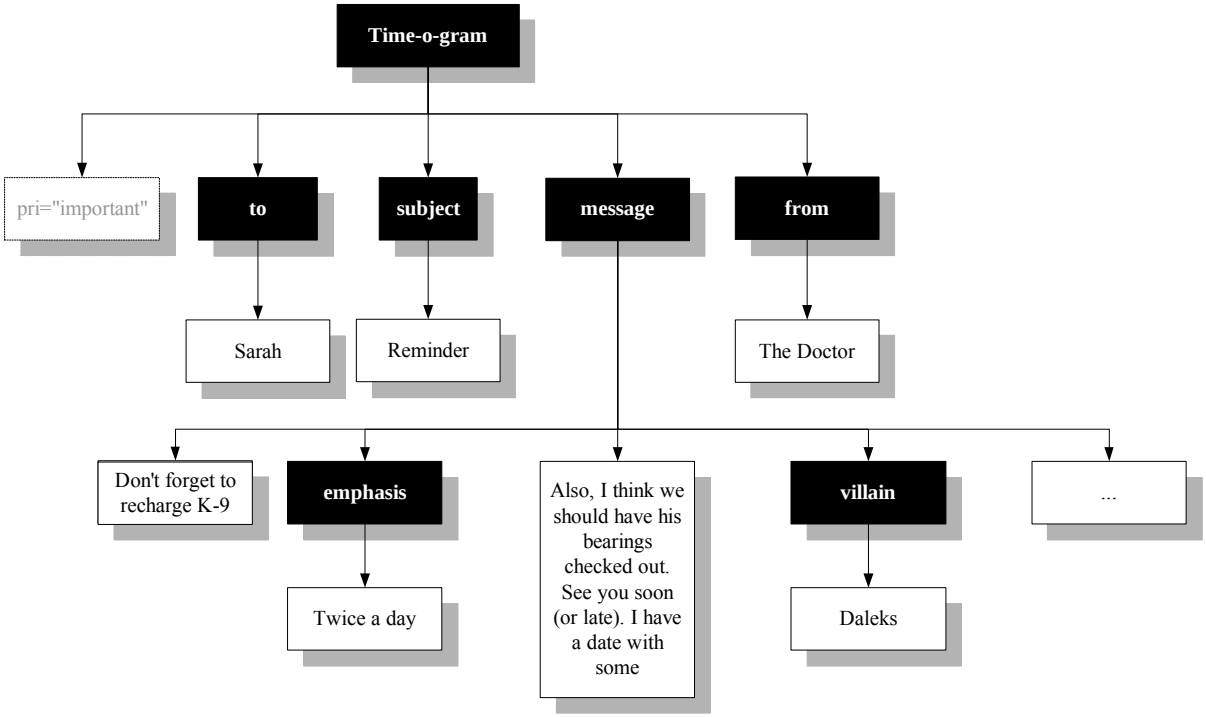
Στο σχήμα που ακολουθεί απεικονίζεται το παραπάνω XML έγγραφο, το οποίο έχει χωριστεί σε περιοχές, κάθε μια από τις οποίες αναπαρίσταται από ένα πλαίσιο και όπου κάθε ένα μπορεί να περιέχει άλλα πλαίσια. Το πρώτο πλαίσιο περιέχει ένα δηλωτικό πρόλογο που δίνει πληροφορίες διαχείρισης για το έγγραφο. Τα υπόλοιπα πλαίσια ονομάζονται στοιχεία. Το μεγαλύτερο στοιχείο, με όνομα `<time-o-gram>`, περικλείει όλα τα άλλα στοιχεία. Στο εσωτερικό του υπάρχουν ενθετημένα τα στοιχεία που αντιπροσωπεύουν τα διακριτά μέρη του εγγράφου. Από το διάγραμμα φαίνεται ότι τα κυριότερα μέρη του στοιχείου `<time-o-gram>` είναι τα στοιχεία: `<to>` που περιγράφουν τον προορισμό, ο αποστολέας `<from>`, το αντικείμενο `<subject>` και το τμήμα που περιέχει το μήνυμα `<message>`. Το τελευταίο στοιχείο είναι το πιο περίπλοκο, επειδή συνδυάζει στοιχεία και κείμενο μαζί.



Εικόνα 2-1 Τα στοιχεία ενός υπομνήματος

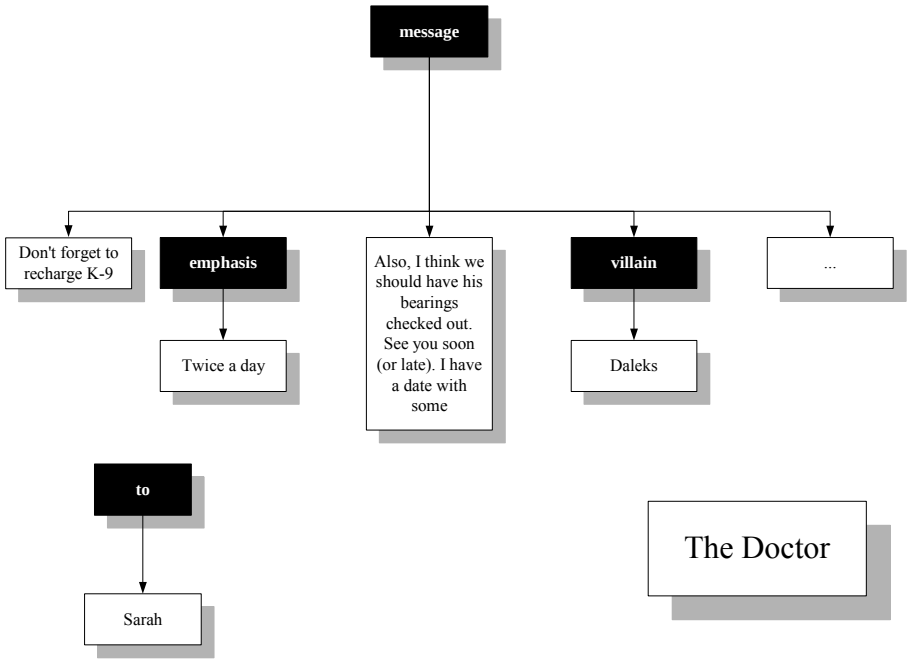
2.2 Δενδρική αναπαράσταση εγγράφου

Τα στοιχεία χωρίζουν το έγγραφο στα μέρη που το αποτελούν. Μπορούν να περιέχουν κείμενο, άλλα στοιχεία, ή και τα δύο. Η εικόνα που ακολουθεί παρουσιάζει μια ιεραρχία στοιχείων που θυμίζει δέντρο. Είναι μια πολύ χρήσιμη παρουσίαση που δείχνει την σχέση μεταξύ τους. Το στοιχείο που βρίσκεται στην κορυφή του δένδρου ονομάζεται *στοιχείο ρίζα* (`<time-o-gram>`). Επίσης ονομάζεται και *στοιχείο του εγγράφου* επειδή περιλαμβάνει όλα τα υπόλοιπα στοιχεία και επιπλέον καθορίζει τα όρια του εγγράφου. Τα αντικείμενα στο τέλος κάθε αλυσίδας ονομάζονται *φύλλα* (*leaves*) και αναπαριστούν το περιεχόμενο του εγγράφου. Κάθε αντικείμενο του δένδρου ονομάζεται *κόμβος*. Ωστόσο, υπάρχει ένα σημείο στην εικόνα 2-2 που δεν έχει ακόμα αναφερθεί: το πλαίσιο στα αριστερά με την ετικέτα `pri`. Στο XML έγγραφο περιέχονταν μέσα στο στοιχείο `<time-o-gram>`, ενώ στη δενδρική αναπαράσταση βρίσκεται εκτός αυτού. Ο κόμβος αυτός αναπαριστά ένα ειδικό είδος περιεχομένου που ονομάζεται ιδιότητα (attribute) και παρέχει επιπλέον πληροφορίες σχετικά με το στοιχείο. Όπως και ένα στοιχείο, η ιδιότητα αποτελείται από ένα όνομα (`pri`) και από μία τιμή (`important`). Οι ιδιότητες χρησιμοποιούνται κυρίως για να τροποποιούν την συμπεριφορά ενός στοιχείου και όχι για να περιέχουν δεδομένα.



Εικόνα 2-2 Η δενδρική αναπαράσταση του απλού XML παραδείγματος

Το στοιχείο ρίζα ονομάζεται και *πρόγονος* (*ancestor*) όλων των άλλων στοιχείων. Τα στοιχεία που βρίσκονται ακριβώς κάτω από το στοιχείο ρίζας αποτελούν τα παιδιά του (*child elements*). Αυτά με την σειρά τους έχουν παιδιά, και η διαδικασία συνεχίζεται μέχρι να φτάσουμε στο κόμβο που δεν έχει παιδί. Τα στοιχεία που μοιράζονται τον ίδιο γονέα ονομάζονται αδέλφια (*siblings*).



Εικόνα 2-3 Μερικά τμήματα του δένδρου

Κάθε κόμβος σε ένα δέντρο μπορεί να θεωρηθεί ως η ρίζα σε ένα μικρότερο υπό-δένδρο. Τα υπό-δένδρο έχουν όλες τις ιδιότητες ενός κανονικού δένδρου, και η κορυφή κάθε υπό-δένδρου είναι ο πρόγονος όλων των κόμβων που βρίσκονται από κάτω. Στην εικόνα 2-3 παρουσιάζονται μερικά παραδείγματα υπό-δένδρων.

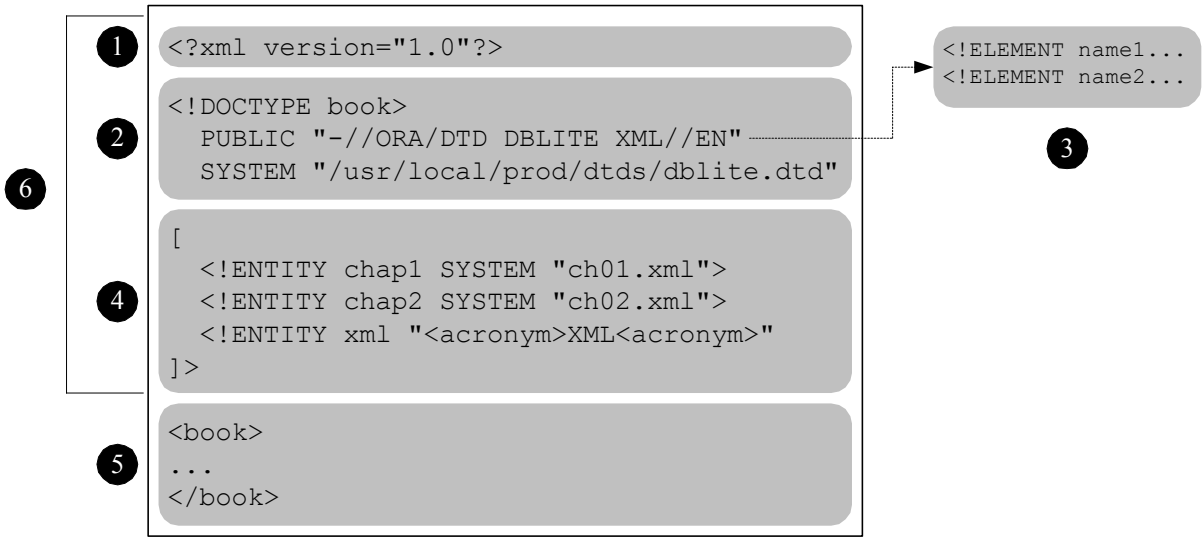
2.3 Ο πρόλογος της XML

Η κορυφή ενός XML εγγράφου παρέχει σημαντικές πληροφορίες που αφορούν το έγγραφο αυτό καθ' αυτό. Οι πληροφορίες αυτές αποτελούν τον *πρόλογο* των XML εγγράφων. Σε μια απλή μορφή, ο πρόλογος αναφέρει μόνο ότι ακολουθεί έγγραφο XML και δηλώνει την έκδοση που χρησιμοποιείται :

```
<?xml version =”1.0”?>
```

Ο πρόλογος μπορεί να περιέχει και επιπλέον πληροφορία που καθορίζει λεπτομέρειες όπως για παράδειγμα το μοντέλο περιγραφής της δομής του (είτε ένα DTD είτε ένα XML Schema), το κείμενο κωδικοποίησης, οδηγίες για τους XML επεξεργαστές κτλ. Περισσότερα για τα μοντέλα περιγραφής των XML εγγράφων αναφέρονται στο Κεφάλαιο 3.

Στη συνέχεια πρόκειται να αναλυθεί ο πρόλογος των XML εγγράφων στις επιμέρους δηλώσεις από τις οποίες αποτελείται. Στην κορυφή βρίσκεται η δήλωση της XML (1). Ύστερα, ακολουθεί η δήλωση του τύπου του εγγράφου (2) και στη συνέχεια ένα σύνολο δηλώσεων (4). Τα 4 αυτά μέρη μαζί αποτελούν το πρόλογο (6) των XML εγγράφων, ωστόσο αυτό δε σημαίνει ότι ο πρόλογος αποτελείται υποχρεωτικά από τα τέσσερα αυτά μέρη. Τέλος, το στοιχείο ρίζα (5) περιέχει το υπόλοιπο έγγραφο. Αυτή η σειρά δε μπορεί να αλλάξει: αν υπάρχει δήλωση XML, πρέπει να βρίσκεται στη πρώτη γραμμή. Αν υπάρχει δήλωση του τύπου του έγγραφο, τότε πρέπει να προηγηθεί του στοιχείου ρίζας.



Εικόνα 2-4 Τα τμήματα ενός XML εγγράφου

Ακολουθεί ένα παράδειγμα προλόγου, του οποίου κάθε γραμμή περιγράφεται αναλυτικά στη συνέχεια

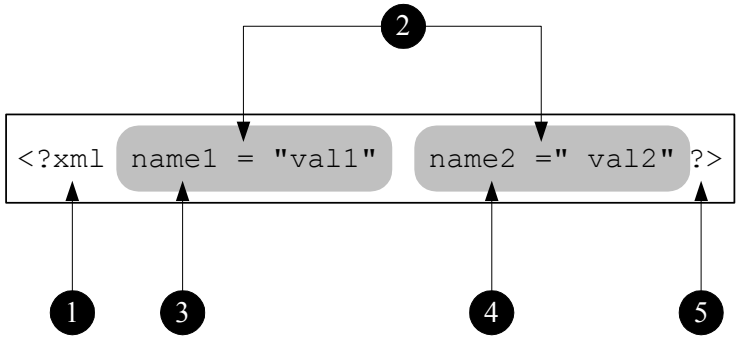
```
❶ <?xml version=”1.0” encoding=”utf-8”?>
❷ <!DOCTYPE time-o-gram
❸       PUBLIC “-//LordsOfTime//DTD TimeOGram 1.8//EN”
❹       SYSTEM “http://www.lordsoftime.org/DTDs/timeogram.dtd”
❺ [
❻       <!ENTITY sj “Sarah Jane”>
        <!ENTITY me “Doctor Who”>
❼ ]>
```

1. Η δήλωση της XML περιγράφει κάποιες από τις πιο γενικές ιδιότητες του εγγράφου, ειδοποιώντας τον επεξεργαστή ότι χρειάζεται XML αναλυτής (parser) για να μεταφράσει το έγγραφο.

2. Το μοντέλο περιγραφής του εγγράφου (συγκεκριμένα το DTD) περιγράφει τον τύπο του στοιχείου ρίζα, <time-o-gram>, και (στις σειρές 3 και 4) ορίζει ένα DTD για να ελέγχει την δομή της σήμανσης.
3. Δημόσιο αναγνωριστικό που δηλώνει το DTD που χρησιμοποιείται.
4. Το αναγνωριστικό συστήματος δείχνει την τοποθεσία του DTD. Στο παράδειγμα είναι ένα URL.
5. Είναι η αρχή ενός εσωτερικού υποσυνόλου, το οποίο παρέχει χώρο για σημαντικές δηλώσεις.
6. Μέσα στο εσωτερικό υποσύνολο υπάρχουν 2 δηλώσεις οντοτήτων.
7. Το τέλος και των 2 εσωτερικών υποσυνόλων (]) και του DTD (>) ολοκληρώνουν τον πρόλογο.

2.3.1 Η δήλωση της XML

Η δήλωση της XML ενημερώνει τον XML επεξεργαστή ότι το έγγραφο χρησιμοποιεί την XML ως γλώσσα σήμανσης. Η δήλωση της XML ξεκινά με τους 5 χαρακτήρες `<?xml` (1), που ακολουθείται από ορισμένες ιδιότητες (2), κάθε ένας έχει ένα όνομα ιδιότητας (3) και μια τιμής μέσα σε εισαγωγικά (4). Η δήλωση τερματίζει με 2-χαρακτήρων οριοθέτη `?>` (5).



Εικόνα 2-5 Η δήλωση της XML

Κατά τη δήλωση της XML, είναι δυνατό να χρησιμοποιηθούν οι εξής ιδιότητες:

version

Ορίζει τον αριθμό της XML έκδοσης που χρησιμοποιείται για τη σήμανση του εκάστοτε εγγράφου. Προς το παρόν υπάρχει μόνο μια έκδοση XML, οπότε η τιμή είναι πάντα 1.0. Παρόλα αυτά, καθώς νέες εκδόσεις εξελίσσονται, αυτή η ιδιότητα θα ειδοποιήσει τον XML επεξεργαστή ποια έκδοση θα χρησιμοποιήσει. Η ιδιότητα αυτή πρέπει να ορίζεται στον πρόλογο.

encoding

Ορίζει το είδος της κωδικοποίησης των χαρακτήρων που χρησιμοποιούνται στο έγγραφο, όπως US-ASCII ή iso-8859-1. Η χρήση άλλου είδους κωδικοποίησης πέραν του προτύπου UTF-8 που παρέχει τις τους Λατινικούς χαρακτήρες θα πρέπει να δηλώνεται.

standalone

Μέσω της ιδιότητας αυτής ο XML επεξεργαστής αν υπάρχουν και άλλα αρχεία που συνοδεύουν το XML έγγραφο και θα πρέπει να τα φορτώσει μαζί με αυτό. Για παράδειγμα, θέτοντας την τιμή της ως "no" δηλώνεται στον XML επεξεργαστή ότι υπάρχει ένα DTD που θα πρέπει να φορτώσει μαζί με το κυρίως αρχείο του εγγράφου. Αν είναι γνωστό ότι το XML έγγραφο δε συνοδεύεται από άλλα αρχεία, τότε θέτοντας standalone="yes" είναι δυνατόν να βελτιωθεί η ταχύτητα του φορτώματος.

Ακολουθούν μερικά παραδείγματα XML δηλώσεων

```
<?xml version="1.0"?>
```

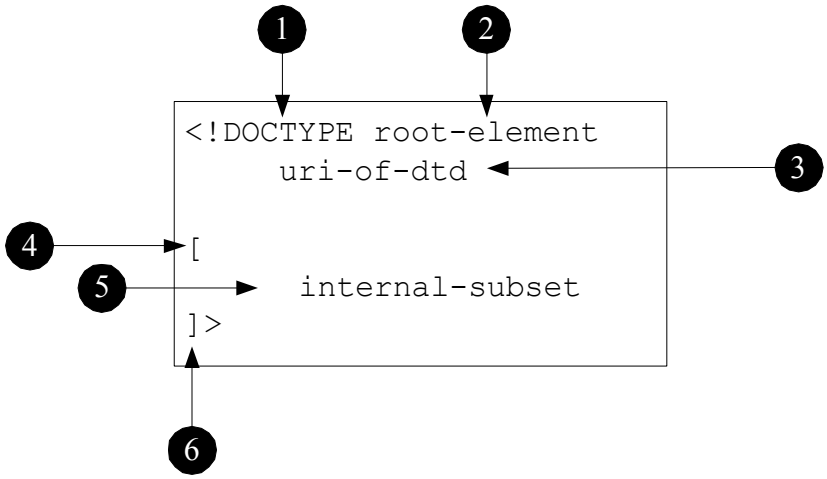
```
<?xml version="1.0"? encoding='US-ASCII' standalone'yes'?>
<?xml version="1.0"? encoding='iso-8859-1' standalone="no"?>
```

Στο πρώτο παράδειγμα δηλώνεται μόνο η έκδοση της γλώσσας XML, ενώ στις υπόλοιπες δύο ορίζεται και το είδος της κωδικοποίησης. Θα πρέπει να σημειωθεί ότι η χρήση όλων των ιδιοτήτων είναι προαιρετική, αλλά θα πρέπει τουλάχιστον να συμπεριλαμβάνεται η ιδιότητα που καθορίζει την έκδοση της XML, σε περίπτωση που αλλάξει κάτι δραστικά στο μέλλον. Τα ονόματα των παραμέτρων πρέπει να είναι πεζά, και όλες οι τιμές πρέπει να περιέχονται σε μονά ή διπλά εισαγωγικά.

2.3.2 Δήλωση του τύπου εγγράφου

Το 2ο μέρος του προλόγου αποτελείται από τη δήλωση του τύπου εγγράφου. Εδώ, είναι δυνατόν να οριστούν ποικίλες παράμετροι, όπως οι δηλώσεις οντοτήτων, το DTD που χρησιμοποιείται, και το όνομα του στοιχείου ρίζας. Όταν γίνεται η αναφορά σε ένα DTD, τότε ένας XML αναλυτής (parser) να συγκρίνει το έγγραφο με το μοντέλο εγγράφου. Η διαδικασία λέγεται έλεγχος εγκυρότητας. Ο έλεγχος εγκυρότητας δεν είναι υποχρεωτικός, αλλά είναι χρήσιμος προκειμένου να εξακριβωθεί αν το εκάστοτε έγγραφο ακολουθεί ορισμένα πρότυπα δομής και περιλαμβάνει τα απαιτούμενα δεδομένα.

Η σύνταξη της δήλωσης τύπου εγγράφου φαίνεται στην εικόνα που ακολουθεί. Η δήλωση αρχίζει με το : `<!DOCTYPE` (1) ακολουθούμενο από το στοιχείο ρίζα (2), το οποίο είναι το πρώτο XML στοιχείο που εμφανίζεται στο έγγραφο και το οποίο περιέχει το υπόλοιπο του εγγράφου. Αν μαζί με το έγγραφο χρησιμοποιείται ένα DTD για τη μοντελοποίηση της δομής του θα πρέπει να συμπεριληφθεί το URI του DTD (3), έτσι ώστε ο XML επεξεργαστής να είναι σε θέση να το αναζητήσει. Μετά από αυτά έρχεται το εσωτερικό υποσύνολο (5), το οποίο περικλείεται από αγκύλες (4 και 6). Η δήλωση τελειώνει με το χαρακτήρα >.



Εικόνα 2-6 Σύνταξη δήλωσης τύπου εγγράφου

Το εσωτερικό υποσύνολο παρέχει τη δυνατότητα εισαγωγής ποικίλων δηλώσεων που χρησιμοποιούνται από το εκάστοτε XML έγγραφο. Αυτές οι δηλώσεις ίσως να περιλαμβάνουν ορισμούς οντοτήτων και μέρη του DTD. Τα εσωτερικά υποσύνολα είναι τα μόνα που μπορούν να δεχτούν αυτές τις δηλώσεις, μέσα στο ίδιο το έγγραφο.

Το εξωτερικό υποσύνολο είναι μια συλλογή από δηλώσεις που υπάρχουν εξωτερικά του εγγράφου, όπως σε ένα DTD. Το URI που παρέχεται σε μια δήλωση τύπου εγγράφου «δείχνει» σε ένα φάκελο που περιέχει αυτές τις εξωτερικές δηλώσεις. Τα εσωτερικά-εξωτερικά υποσύνολα είναι εναλλακτικά.

2.4 Δομικές μονάδες XML εγγράφων

2.4.1 Στοιχεία

Τα στοιχεία (elements) αποτελούν τα κυριότερα μέρη ενός XML εγγράφου. Τα στοιχεία μπορεί να περιέχουν είτε άλλα στοιχεία είτε μόνο κείμενο είτε ένα συνδυασμό κειμένου και στοιχείων. Το στοιχείο που ακολουθεί περιέχει μόνο κείμενο:

```
<text_element>Hello World!</ text_element >
```

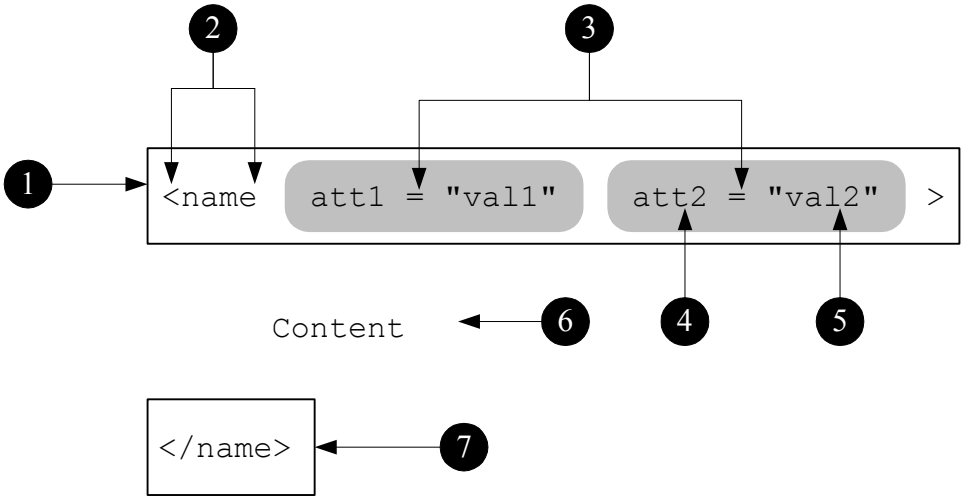
ενώ το στοιχείο που ακολουθεί περιέχει ένα συνδυασμό κειμένου και στοιχείων

```
<outer>
      this is text<inner>more text</inner>
      still more text
</outer>
```

Μερικά στοιχεία είναι δυνατό να είναι κενά και η απαιτούμενη πληροφορία να παρέχεται είτε μέσω της θέσης τους, είτε μέσω των ιδιοτήτων που περιέχουν. Ένα παράδειγμα κενού στοιχείου είναι το εξής

```
<image width="500" height= "200"/>
```

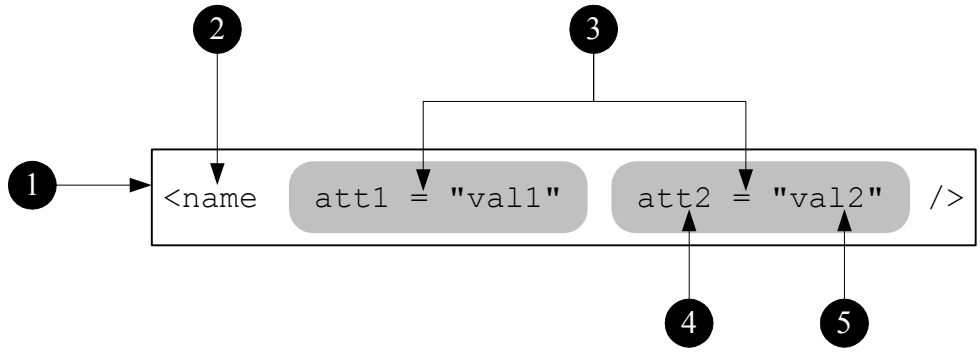
Στην εικόνα που ακολουθεί απεικονίζεται η σύνταξη ενός στοιχείου. Ξεκινάει με μια ετικέτα αρχής (1), η οποία αποτελείται από μια αριστερή γωνιακή αγκύλη (<) ακολουθούμενη από το όνομα του στοιχείου (2). Η αρχική ετικέτα μπορεί να περιέχει ορισμένες ιδιότητες (3) που χωρίζονται με κενό και τελειώνει με μια δεξιά γωνιακή αγκύλη (>). Μια ιδιότητα καθορίζει ένα χαρακτηριστικό του στοιχείου και αποτελείται από ένα όνομα (4) ενώνεται μέσω του συμβόλου της ισότητας (=) με μια τιμή σε εισαγωγικά (5). Ένα στοιχείο μπορεί να έχει οποιοδήποτε νούμερο χαρακτηριστικών, αλλά δυο χαρακτηριστικά δεν μπορούν να έχουν το ίδιο όνομα. Αυτό που ακολουθεί την αρχική ετικέτα είναι το περιεχόμενο του στοιχείου (6), το οποίο από κάτω ακολουθείται από μια τελική ετικέτα (7). Η τελική ετικέτα αποτελείται από μια αριστερή γωνιακή αγκύλη, το χαρακτήρα slash (/), το όνομα του στοιχείου, και με μια δεξιά γωνιακή αγκύλη. Η τελική ετικέτα δεν έχει χαρακτηριστικά, και το όνομα του στοιχείου πρέπει να ταιριάζει ακριβώς με το όνομα της αρχικής ετικέτας.



Εικόνα 2-7 Σύνταξη Στοιχείων

Στο σχήμα 2-8 βλέπουμε ένα κενό στοιχείο (empty element) αποτελούμενο από μια απλή ετικέτα (1) η οποία ξεκινάει με μια αριστερή γωνιακή αγκύλη (<) ακολουθείται από το όνομα του στοιχείου (2)και στη συνέχεια από ένα αριθμό ιδιοτήτων (3), που το κάθε ένα

αποτελείται από ένα όνομα (4) και μια τιμή σε εισαγωγικά (5), και το στοιχείο τελειώνει με το χαρακτήρα slash (/) και μια δεξιά γωνιακή αγκύλη.

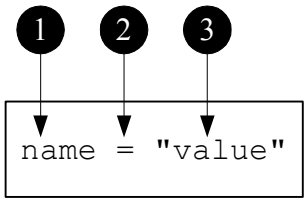


Εικόνα 2-8 Σύνταξη Κενών Στοιχείων

Το όνομα ενός στοιχείου πρέπει να ξεκινάει με ένα γράμμα ή με το χαρακτήρα υπογράμμισης (underscore) και μπορεί να περιέχει έναν οποιονδήποτε αριθμό από γράμματα, αριθμούς, παύλες, και χαρακτήρες υπογράμμισης. Τα ονόματα των στοιχείων μπορούν να περιέχουν τονισμένους Ρωμαϊκούς (Roman) χαρακτήρες, γράμματα από αλφάβητα όπως Κυριλλικό, Ελληνικό, Εβραϊκό, Αραβικό, Ταϋλανδέζικο, Hiragana, Katakana και Devanagari και ιδεογράμματα από Κινέζικο, Ιαπωνικό και Κορεάτικο αλφάβητο. Η άνω και κάτω τελεία (:) χρησιμοποιείται σε χώρους ονοματοδοσίας (namespaces)και πρέπει να αποφεύγεται η χρήση αυτού στα ονόματα των στοιχείων τα οποία δεν χρησιμοποιούν ένα χώρο ονοματοδοσίας. Κενά, ετικέτες, νέες σειρές, σύμβολο ισότητας, και εισαγωγικοί χαρακτήρες και διαχωριστές για ονόματα στοιχείων, ονόματα χαρακτηριστικών, και τιμές χαρακτηριστικών δεν επιτρέπονται. Ορισμένα παραδείγματα στοιχείων των οποίων τα ονόματα ακολουθούν τους κανόνες ονοματολογίας είναι τα εξής: <Bob>, <chapter.title>, <THX-1138>, ή ακόμα <_>. Η XML διαχωρίζει τους πεζούς από τους κεφαλαίους χαρακτήρες, συνεπώς τα <Para>, <para>, <pArA> είναι τρία διαφορετικά στοιχεία.

2.4.2 Ιδιότητες

Οι ιδιότητες παρέχουν επιπλέον πληροφορία στα στοιχεία, πέρα από όνομά του ή το περιεχόμενό του. Οι ιδιότητες εμφανίζονται πάντα με τη μορφή του ζευγαριού όνομα-τιμή (name-value). Όπως απεικονίζεται στο επόμενο σχήμα μια ιδιότητα αποτελείται από ένα όνομά της (1), το σύμβολο της ισότητας (2), και την τιμή που αυτή δέχεται, σε εισαγωγικά (3).



Εικόνα 2-9 Η σύνταξη των ιδιοτήτων

Ένα στοιχείο μπορεί να έχει οποιονδήποτε αριθμό χαρακτηριστικών, εφ' όσον το κάθε στοιχείο έχει μοναδικό όνομα. Εδώ είναι ένα στοιχείο με τρία χαρακτηριστικά:

```
<circle origin_x="10.0" origin_y="20.0" radius="10.0">
```

Οι ιδιότητες διαχωρίζονται μεταξύ τους με κενά. Εμφανίζονται μετά το όνομα του εκάστοτε στοιχείου, αλλά μπορούν να βρίσκονται σε οποιαδήποτε σειρά. Οι τιμές των ιδιοτήτων πρέπει να βρίσκονται μεταξύ μονών (') ή διπλών εισαγωγικών ("). Μια

ιδιότητα πρέπει να εμφανίζεται μόνο μια φορά μέσα σε ένα στοιχείο. Για το λόγο αυτό η επόμενη σύνταξη είναι λανθασμένη:

```
<team person="sue" person= "joe" person="jane">
```

Το παραπάνω μπορεί να αποφευχθεί με τη χρήση μιας ιδιότητας που περιέχει και τις τρεις τιμές:

```
<team persons="sue joe jane">
```

είτε με τη χρήση τριών διαφορετικών ιδιοτήτων

```
<team person1="sue" person2="joe" person3="jane">
```

είτε με τη χρήση στοιχείων

```
<team>
      <person>sue</person>
      <person>joe</person>
      <person>jane</person>
</team>
```

Προκαθορισμένες Ιδιότητες

Η XML παρέχει ένα αριθμό προκαθορισμένων ιδιοτήτων των οποίων η χρήση έχει καθοριστεί για ειδικούς σκοπούς. Το όνομα αυτών των ιδιοτήτων ξεκινάει με το πρόθεμα xml. Οι ιδιότητες xml:lang και xml:space αποτελούν δύο από τις πιο προκαθορισμένες ιδιότητες της γλώσσας σήμανσης XML v1.0. Αυτά τα ειδικά ονόματα χαρακτηριστικών περιγράφονται παρακάτω:

xml:lang

Ταξινομεί ένα στοιχείο με βάση τη γλώσσα του στοιχείου του. Για παράδειγμα, xml:lang="en" περιγράφει ένα στοιχείο που έχει περιεχόμενο στα Αγγλικά. Αυτό είναι χρήσιμο για τη δημιουργία υποθετικού κειμένου, του οποίου τα περιεχόμενα είναι επιλεγμένα από ένα XML επεξεργαστή βασισμένο σε κριτήρια όπως σε τη γλώσσα θέλει να δει ο χρήστης το αρχείο.

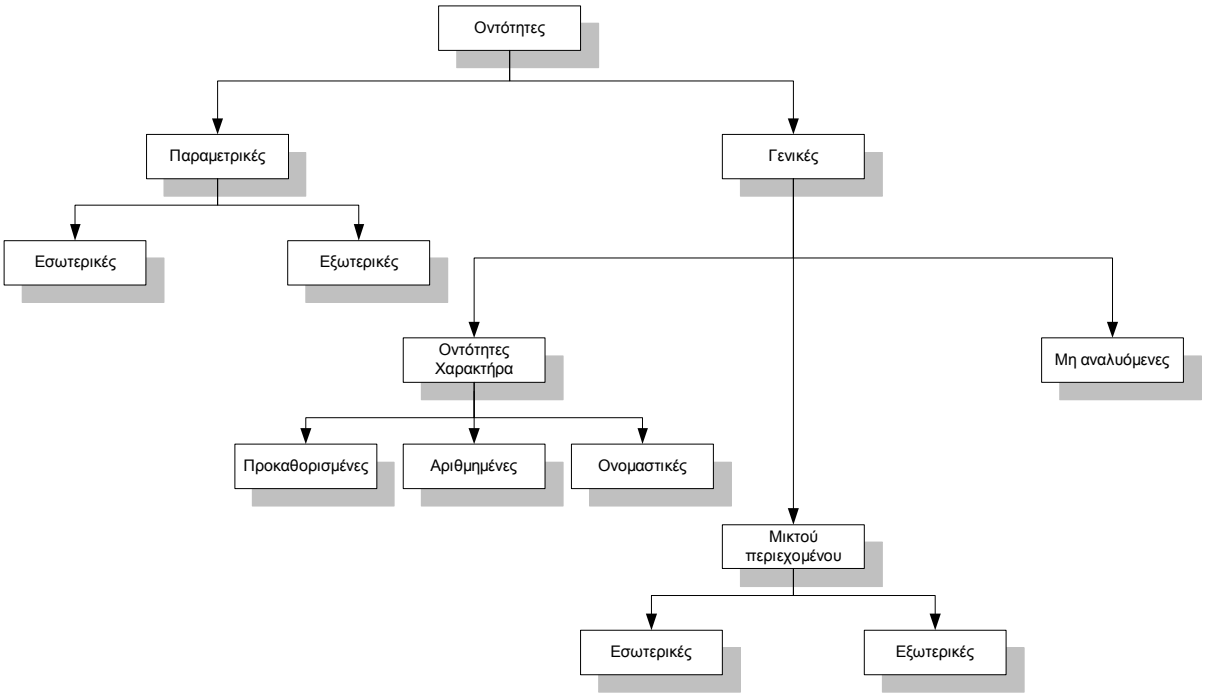
xml:space

Προσδιορίζει κατά πόσον τα διαδοχικά κενά (whitespace) πρέπει να διατηρούνται στο περιεχόμενο ενός στοιχείου. Εάν δέχεται την τιμή "preserve', ο XML επεξεργαστής θα πρέπει να διατηρήσει όλα τα κενά, τις αλλαγές γραμμών ή και τις ετικέτες που τυχόν υπάρχουν στο περιεχόμενο του στοιχείου. Εάν δέχεται την τιμή "default", τότε ο XML επεξεργαστής μπορεί να διαχειριστεί τα κενά με βάση την προκαθορισμένη του συμπεριφορά. Εάν η ιδιότητα xml:space παραλείπεται, τότε εξ' ορισμού ο επεξεργαστής διατηρεί τα κενά στο περιεχόμενο των στοιχείων. Για το λόγο αυτό, αν είναι επιθυμητή η μείωση των διαδοχικών κενών στο περιεχόμενο ενός στοιχείου η ιδιότητα xml:space, θα πρέπει να δέχεται την τιμή "default", υπό την προϋπόθεση ότι η προκαθορισμένη λειτουργία του XML επεξεργαστή είναι η αφαίρεση των επιπρόσθετων κενών.

2.4.3 Οντότητες

Οι οντότητες (entities) είναι μεταβλητές στις οποίες αποθηκεύεται περιεχόμενο το οποίο μπορεί να επαναχρησιμοποιηθεί σε αρκετά σημεία του εγγράφου. Η πιο γνωστή αναφορά μιας οντότητας στην HTML είναι η: " ". Η αναφορά σε αυτήν την οντότητα χρησιμοποιείται στην HTML για να παρεμβάλει ένα πρόσθετο κενό διάστημα σε ένα έγγραφο.

Στην εικόνα που ακολουθεί απεικονίζεται μια ταξινόμηση των οντοτήτων. Οι δύο σημαντικότεροι τύποι οντοτήτων είναι οι παραμετρικές οντότητες (που χρησιμοποιούνται μόνο στα DTDs και αναφέρονται στο κεφάλαιο 3) και οι γενικές οντότητες. Σε αυτό το μέρος θα εστιάσουμε την προσοχή μας στο δεύτερο τύπο των οντοτήτων, τις γενικές οντότητες και συγκεκριμένα στις. Οι γενικές οντότητες παρέχουν τη δυνατότητα αποθήκευσης δεδομένων τα οποία είτε περιέχονται στη ρίζα στοιχείο ενός XML εγγράφου είτε μέσα σε αυτή.



Εικόνα 2-10 Ταξινόμηση των οντοτήτων.

Μια οντότητα αποτελείται από ένα όνομα και μια τιμή. Όταν ένας XML αναλυτής ξεκινάει να επεξεργάζεται ένα έγγραφο, πρώτα διαβάζει μια σειρά από δηλώσεις σε μερικές από τις οποίες ορίζονται οντότητες στις οποίες ένα όνομα συσχετίζεται με μια τιμή. Η τιμή αυτή μπορεί να είναι οτιδήποτε από ένα απλό χαρακτήρα ως ένα XML έγγραφο. Ενώ ο αναλυτής αναλύει το XML έγγραφο, συναντάει αναφορές οντοτήτων οι οποίες είναι ειδικοί δείκτες που προέρχονται από ονόματα οντοτήτων. Αντικαθιστά την αναφορά οντότητας με το κατάλληλο κείμενο ή σήμανση που έχει αποθηκευτεί στην αντίστοιχη οντότητα και από το σημείο αυτό συνεχίζει την ανάλυση του κειμένου.

Υπάρχουν δυο είδη σύνταξης των αναφορών οντοτήτων. Ο πρώτος τύπος αποτελείται από το σύμβολο "`&`", το όνομα της οντότητας και το σύμβολο "`;`" και χρησιμοποιείται για τις γενικές οντότητες. Ο δεύτερος τύπος, διακρίνεται από το σύμβολο "`%`" αντί του συμβόλου "`&`", και χρησιμοποιείται για παραμετρικές οντότητες (Βλέπε κεφάλαιο 3.3.4).

```
1  &name;

2  %name;
```

Ακολουθεί ένα παράδειγμα ενός εγγράφου στο οποίο δηλώνονται τρεις οντότητες και η αναφορά αυτών γίνεται μέσα στο κείμενο που περιέχεται στο στοιχείο ρίζας:

```
<?xml version="1.0"?>

<! DOCTYPE message SYSTEM "/xmlstuff/dtds/message.dtd"
```

```
[
      <! ENTITY client "Mr. Rufus Xavier Sasperilla">
      <! ENTITY agent "Ms. Sal1y Tashuns">
      <! ENTITY phone "<number>617-555-1299</number>">
]>
<message>
      <opening>Dear &client;</opening>
      <body>We have an exciting opportunity for you! A set of
      ocean-front cliff dwellings in Pi&I241;ata, Mexico have been
      renovated as time-share vacation homes. They're going fast! To
      reserve a place for your holiday, call &agent; at &phone; .
      Hurry, &client; . Time is running out!</body>
</message>
```

Οι οντότητες `&client;` , `&agent;` και `&phone;` δηλώνονται στο εσωτερικό υποσύνολο του εγγράφου και αναφέρεται μέσα στο <message> στοιχείο. Μια τέταρτη οντότητα, ñ, είναι ένας αριθμημένος χαρακτήρας οντότητας που αναπαριστά τον χαρακτήρα ñ. Αυτή η οντότητα αναφέρεται αλλά δεν δηλώνεται. Καμία δήλωση δεν είναι απαραίτητη γιατί οι αριθμητικοί χαρακτήρες οντοτήτων είναι ρητά ορισμένοι στην XML σαν αναφορές σε χαρακτήρες στην τρέχων ομάδα χαρακτήρων. Ο XML αναλυτής απλά αντικαθιστά την αριθμημένη οντότητα με το σωστό χαρακτήρα. Το προηγούμενο παράδειγμα μοιάζει με αυτό με όλες τις οντότητες αναλυμένες:

```
<?xml version="1.0"?>
<! DOCTYPE message SYSTEM "/xmlstuff/dtds/message.dtd">
<message>
      <opening>Dear Mr. Rufus Xavier Sasperilla</opening>
      <body>We have an exciting opportunity for you! A set of
      ocean-front cliff dwellings in Pinãta, Mexico have been
      renovated as time-share vacation homes. They're going fast! To
      reserve a place for your holiday. call Ms. Sally Tashuns at
      <number>617-555-1299</number>.
      Hurry, Mr. Rufus Xavier Sasperilla. Time is running out!</body>
</message>
```

Όλες οι οντότητες πρέπει να δηλώνονται πριν χρησιμοποιηθούν σε ένα έγγραφο. Δύο αποδεκτά μέρη για να δηλωθούν είναι στο εσωτερικό υποσύνολο, το οποίο είναι ιδανικό για τοπικές οντότητες και σε ένα εξωτερικό DTD το οποίο είναι καταλληλότερο για οντότητες που μοιράζονται μεταξύ εγγράφων. Εάν ο αναλυτής συναντήσει μια αναφορά οντότητας ή οποία δεν έχει δηλωθεί, δεν μπορεί να εισάγει ένα κείμενο προς αντικατάσταση μέσα στο έγγραφο γιατί δεν γνωρίζει με τι να αντικαταστήσει την οντότητα. Αυτό το λάθος εμποδίζει το έγγραφο από το να θεωρείται καλά δομημένο.

Οι οντότητες οι οποίες περιέχουν ένα απλό χαρακτήρα λέγονται φυσικοί χαρακτήρες οντοτήτων. Αυτές διαιρούνται σε αρκετές ομάδες:

Προκαθορισμένοι χαρακτήρες οντοτήτων

Μερικοί χαρακτήρες δεν μπορούν να χρησιμοποιηθούν μέσα σε ένα κείμενο ενός XML εγγράφου επειδή είναι δεσμευμένοι από την XML όπως για παράδειγμα, οι γωνιακές

αγκύλες (<>) χρησιμοποιούνται για να οριοθετήσουν τις ετικέτες των στοιχείων. Η XML παρέχει τους ακόλουθους προκαθορισμένους χαρακτήρες οντοτήτων προκειμένου να αποφευχθεί αυτό.

Όνομα	Τιμή
amp	&
apos	‘
gt	>
lt	<
quot	“

Αριθμητικοί χαρακτήρες οντοτήτων.

Η XML υποστηρίζει το σύνολο χαρακτήρων Unicode συνεπώς είναι δυνατή η χρήση οποιουδήποτε χαρακτήρα του Unicode στο XML έγγραφο. Το πρόβλημα που προκύπτει είναι πώς θα εισαχθεί ένας χαρακτήρας από ένα πληκτρολόγιο που παρέχει έναν περιορισμένο αριθμό χαρακτήρων ή πώς θα αναπαρασταθεί ένας τέτοιος χαρακτήρας σε έναν απλό κειμενογράφο. Μια λύση είναι η χρήση ενός αριθμητικού χαρακτήρα οντοτήτων, της οποίας το όνομα θα έχει τη μορφή `#n`, όπου το `n` είναι ένα νούμερο το οποίο αναπαριστά τη θέση του χαρακτήρα μέσα στην ομάδα χαρακτήρων του Unicode.

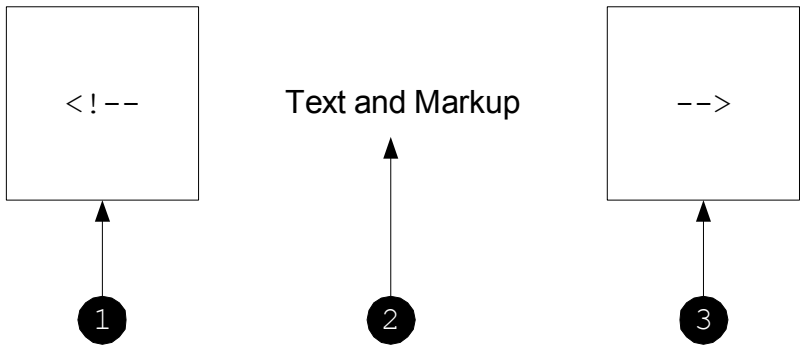
Το νούμερο μέσα στο όνομα της οντότητας μπορεί να εκφραστεί σε δεκαδική ή δεκαεξαδική μορφή. Για παράδειγμα ,ο χαρακτήρας ç είναι ο 231ος χαρακτήρας του Unicode. Μπορεί να αναπαρασταθεί στο δεκαδικό ως `&#231;` ή στο δεκαεξαδικό ως `&#xe7;`. Θα πρέπει να σημειωθεί ότι η δεκαεξαδική αναπαράσταση ξεχωρίζει με ένα x ως το πρόθεμα μπροστά από το νούμερο. Η έκταση των χαρακτήρων που μπορούν να αναπαρασταθούν με αυτό τον τρόπο ξεκινάει από μηδέν και φτάνει μέχρι 65,536.

Ονομαστικοί χαρακτήρες οντοτήτων.

Το πρόβλημα που εμφανίζεται με τους αριθμητικούς χαρακτήρες οντοτήτων η δύσκλη απομνημόνευσή τους. Μια λύση είναι η χρήση μνημονικών ονομάτων. Αυτοί οι ονομαστικοί χαρακτήρες οντοτήτων χρησιμοποιούν εύκολα στην απομνημόνευση ονόματα όπως για παράδειγμα η οντότητα Þ που συμβολίζει τον Ισλανδικό χαρακτήρα “Þ”. Αντίθετα από τους προκαθορισμένους και αριθμητικούς χαρακτήρες, πρέπει να δηλωθούν οι ονομαστικοί χαρακτήρες οντοτήτων.

2.4.4 Σχόλια

Τα σχόλια είναι σημειώσεις στο έγγραφο που δεν μεταφράζονται από τον αναλυτή. Ο τρόπος με τον οποίο συντάσσονται τα σχόλια απεικονίζεται στην παρακάτω εικόνα



Εικόνα 2-11 Η σύνταξη των σχολίων

Ένα σχόλιο αρχίζει με 4 χαρακτήρες : μια αριστερή γωνιακή αγκύλη, ένα θαυμαστικό, και δύο παύλες (1). Τελειώνει με δύο παύλες και μια δεξιά γωνιακή αγκύλη. Μεταξύ αυτών των ορίων βρίσκεται το κείμενο που δε θα μεταφραστεί (2). Το σχόλιο μπορεί να περιέχει κάθε είδος κειμένου, συμπεριλαμβανομένου κενών, νέας γραμμής καθώς και σήμανσης. Παρόλα αυτά αφού οι 2 παύλες χρησιμοποιούνται για να δείχνουν στον αναλυτή πότε ένα σχόλιο αρχίζει και πότε τελειώνει, δε μπορούν να χρησιμοποιηθούν μέσα στο σχόλιο. Αυτό σημαίνει ότι αντί να χρησιμοποιήσουμε παύλες για να φτιάξουμε μια ορατή γραμμή, μπορούμε να χρησιμοποιήσουμε άλλο σύμβολο όπως το ίσον (=) ή την υπογράμμιση (_) :

Σωστό: `<!--================--->`

Σωστό: `<!--_____________________-->`

Σωστό: `<!-- - - - - - - - - - - - - - - - - - -->`

Λάθος: `<!------------------------------ -->`

Λάθος: `<!--    -- Λάθος --   -->`

Τα σχόλια μπορούν να μπούν σε οποιοδήποτε σημείο του εγγράφου εκτός πριν από την δήλωση της XML και μέσα σε ετικέτες, όπου ο αναλυτής θα τα αγνοήσει εντελώς .Αυτό το σημείο XML:

Από τη στιγμή που τα σχόλια μπορούν να περιέχουν σήμανση, μπορούν να χρησιμοποιηθούν για να σβήσουν μέρη ενός εγγράφου. Αυτό είναι χρήσιμο όταν θέλουμε να αφαιρέσουμε μια ενότητα προσωρινά κρατώντας τη στο φάκελο για μετέπειτα χρήση του. Σε αυτό το παράδειγμα, ο κωδικός περιοχής παραλείπεται:

```
<p>Our store is located at:</p>
<!--
<address>59 sunspot avenue</address>
-->
<address>210 blather street</address>
```

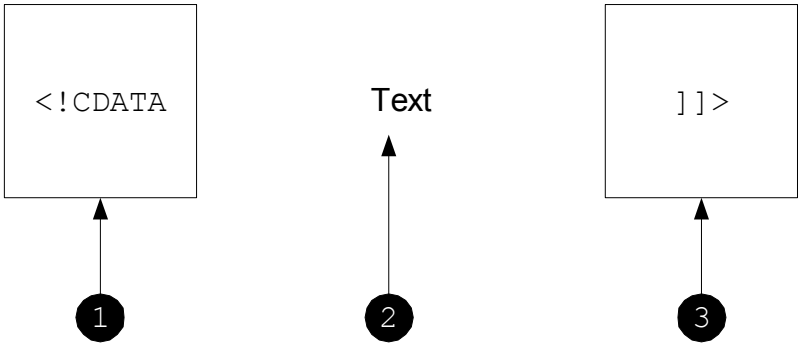
2.4.5 Τμήματα CDATA

Σε περίπτωση που χρησιμοποιούνται συχνά χαρακτήρες σήμανσης στο κείμενο, η χρήση των προκαθορισμένων οντοτήτων <, &, > είναι αρκετά κουραστική. Ωστόσο υπάρχει ένας άλλος τρόπος να τυπώσουμε πολλούς δεσμευμένους χαρακτήρες: με τη χρήση CDATA τμημάτων.

CDATA είναι συντομογραφία του 'character data' που σημαίνει 'όχι σήμανση'. Ο αναλυτής ενημερώνεται ειδοποιείς τον αναλυτή ότι αυτή η ενότητα του εγγράφου δε περιέχει σήμανση και πρέπει να χειριστεί σαν κανονικό κείμενο. Το μόνο που δε μπορεί να μπει

μέσα σε ένα τμήμα CDATA είναι ο οριοθέτης τέλους (]]>). Για το λόγο αυτό θα πρέπει να χρησιμοποιηθεί μια προκαθορισμένη οντότητα και να γραφτεί ως εξής:]]>.

Η σύνταξη των CDATA τμημάτων φαίνεται στην εικόνα που ακολουθεί. Μια CDATA ενότητα αρχίζει με έναν οριοθέτη 9 χαρακτήρων <![CDATA[(1) και τελειώνει με τον οριοθέτη]]> (3). Το περιεχόμενο της ενότητας (2) μπορεί να περιέχει χαρακτήρες σήμανσης (<, >, και &) αλλά αγνοούνται από τον XML επεξεργαστή.



Εικόνα 2-12 Η σύνταξη των CDATA ενοτήτων.

Ακολουθεί ένα παράδειγμα ενός τμήματος CDATA:

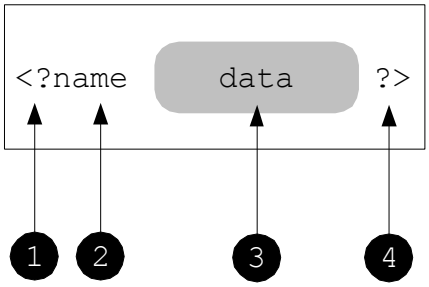
```
<para>Then you can say <![CDATA[if (&x < &y)]]> and be done with
it.</para>
```

2.4.6 Οδηγίες Επεξεργασίας

Οι πληροφορίες παρουσίασης πρέπει να διαχωρίζονται από το έγγραφο, όπου είναι δυνατό. Υπάρχουν και στιγμές που δεν έχουμε άλλη επιλογή, για παράδειγμα, αν χρειάζεται να αποθηκεύσουμε αριθμημένες σελίδες στο έγγραφο για να διευκολύνουμε την παραγωγή ενός ευρετηρίου. Αυτή η πληροφορία εφαρμόζεται μόνο σε ένα συγκεκριμένο XML επεξεργαστή και μπορεί να είναι παραπλανητική για τους άλλους. Η οδηγία για αυτού του είδους πληροφορίας είναι μια οδηγία επεξεργασίας. Είναι ένα δοχείο με δεδομένα που στοχεύουν σε ένα συγκεκριμένο XML επεξεργαστή. Οι οδηγίες επεξεργασίας (ΟΕ) περιέχουν 2 κομμάτια πληροφορίας: ο στόχος και τα δεδομένα. Ο αναλυτής περνάει τις οδηγίες επεξεργασίας στο επόμενο επίπεδο επεξεργασίας. Αν ο χειριστής των οδηγιών επεξεργασίας αναγνωρίσει το στόχο, μπορεί να διαλέξει να χρησιμοποιήσει τα δεδομένα; αλλιώς τα δεδομένα θα απορριφθούν.

Στην εικόνα που ακολουθεί φαίνεται η σύνταξη των ΟΕ. Μια ΟΕ αρχίζει με έναν οριοθέτη 2 χαρακτήρων (1) που αποτελείται από μια ανοιχτή γωνιακή αγκύλη και ένα ερωτηματικό (<?), ακολουθείται από το στόχο (2), μια προαιρετική σειρά χαρακτήρων που είναι τα δεδομένα της ΟΕ (3), και έναν τερματικό οριοθέτη (4), αποτελούμενο από ένα ερωτηματικό και μια κλειστή γωνιακή αγκύλη (?>).

Όπως μπορεί να γίνει αντιληπτό οι Οδηγίες Επεξεργασίας μοιάζουν με τη δήλωση της XML. Η XML δήλωση μπορεί να θεωρηθεί σα μια οδηγία ΟΕ για όλους τους XML επεξεργαστές οι οποίοι που λαμβάνουν γενικές πληροφορίες για το έγγραφο.



Εικόνα 2-13 Η σύνταξη ΟΕ

Ο στόχος είναι μια λέξη - κλειδί, την οποία χρησιμοποιεί ο XML επεξεργαστής για να προσδιορίσει αν τα δεδομένα προορίζονται για αυτόν ή όχι. Η λέξη - κλειδί δε σημαίνει απαραίτητα κάτι, όπως το όνομα του λογισμικού που θα το χρησιμοποιήσει. Παραπάνω από ένα προγράμματα μπορούν να χρησιμοποιήσουν μια ΟΕ, και ένα πρόγραμμα μόνο του μπορεί να δεχτεί πολλές ΟΕ. Είναι περίπου σα να γράφουμε ένα μήνυμα σε ένα τοίχο «Το πάρτι μεταφέρθηκε στο πράσινο σπίτι» και τα άτομα που ενδιαφέρονται για το πάρτι θα ακολουθήσουν τις οδηγίες, εκείνοι που δεν ενδιαφέρονται, όχι.

Η ΟΕ μπορεί να περιέχει οτιδήποτε δεδομένα εκτός από τον συνδυασμό ?>, ο οποίος μεταφράζετε ως ο τερματικός οριοθέτης. Ακολουθούν μερικά παραδείγματα νόμιμων ΟΕ:

```
<?flubber pg=9 recto?>

<?thingie?>

<?xyz stop: the presses?>
```

Αν δεν υπάρχει σειρά χαρακτήρων (δεδομένα), ο στόχος μόνος του μπορεί να θεωρηθεί ως δεδομένο. Ένα καλό παράδειγμα είναι «forced line break». Φανταστείτε ότι υπάρχει μια μεγάλη ενότητα που απλώνεται έξω από την σελίδα. Αντί να σπάσουμε τον τίτλο οπουδήποτε, θέλουμε να τον σπάσουμε σε συγκεκριμένο σημείο. Εδώ φαίνεται το αποτέλεσμα:

```
   <title>The Confabulation of Branklefitzers <?lb?>in a Portlebunky
Frammins <?lb?>Without Denaculization of <?lb?>Crunky Grabblefooties
</title>
```

2.5 Συντακτικοί κανόνες XML εγγράφων

Όλα τα έγγραφα XML, και αυτά χωρίς DTD και αυτά που είναι έγκυρα σύμφωνα με ένα DTD, πρέπει να είναι καλά ορισμένα. Πρέπει να αρχίσουν με μια XML δήλωση και να ακολουθούν τους της δήλωσης του εγγράφου ως αυτόνομο - Standalone):

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>

<foo>

  <bar>...<blort/>...</bar>

</foo>
```

Τα XML έγγραφα προκειμένου να είναι καλά ορισμένα θα πρέπει να προσαρμόζονται στους ακόλουθους ακριβείς κανόνες σύνταξης.

Τα XML στοιχεία πρέπει να έχουν μια ετικέτα τέλους

Σε μερικά HTML στοιχεία, όπως είναι η παράγραφος (<p>), δεν χρειάζεται μια ετικέτα τέλους. Ωστόσο, όλα τα XML στοιχεία πρέπει να έχουν μια ετικέτα τέλους.

Για να γίνει κατανοητό γιατί απαιτείται αυτός ο κανόνας, εξετάζεται ένα παράδειγμα από την HTML:

Το στοιχείο <IMG>, το οποίο καθορίζεται (στα DTDs της SGML για την HTML) ως ΚΕΝΟ (EMPTY), δεν έχει μια ετικέτα τέλους (δεν υπάρχει η ετικέτα </IMG>), όμως, και πολλά άλλα στοιχεία της HTML (όπως το <P>) επιτρέπουν την παράλειψη της ετικέτας τέλους χάριν συντομίας.

Εάν ένας XML επεξεργαστής διαβάζει ένα αρχείο HTML χωρίς τη γνώση αυτού (επειδή δεν χρησιμοποιεί ένα DTD), και συναντήσει το <IMG> ή το <P> ή πολλές άλλες ετικέτες έναρξης, δεν θα είχε κανέναν τρόπο για να ξέρει εάν πρέπει η όχι να αναμένει μια ετικέτα τέλους, πράγμα που καθιστά αδύνατο να γνωρίζει εάν το υπόλοιπο του αρχείου είναι σωστό ή όχι, επειδή έχει χάσει το σημάδι για το εάν είναι μέσα σε ένα στοιχείο ή εάν έχει τελειώσει με αυτό.

Τα καλοσχηματισμένα XML έγγραφα επομένως απαιτούν ετικέτες έναρξης και ετικέτες τέλους σε κάθε συνηθισμένο στοιχείο, και σε οποιαδήποτε ΚΕΝΑ (EMPTY) στοιχεία πρέπει να γίνουν σαφή, είτε με τη χρήση ετικετών έναρξης και ετικετών τέλους, είτε με την επισύναψη μιας κάθετου στην ετικέτα έναρξης πριν από το κλείσιμο > ως σημάδι ότι δεν θα υπάρξει καμιά ετικέτα τέλους.

Χρήση Κενών (Empty) Στοιχεία

Η XML επιτρέπει τα κενά στοιχεία με την παρακάτω μορφή σύντμησης:

<title></title> Κανονική σύνταξη

<title/> Σύνταξη με σύντμηση

Οποιοδήποτε κενό στοιχείο (π.χ. αυτά που δε χρησιμοποιούν ετικέτα τέλους στην HTML όπως τα <IMG>, <HR>, και
 και άλλα) πρέπει *είτε* να τελειώσουν με /> *είτε* θα πρέπει να μοιάσουν με μη – κενά στοιχεία έχοντας μια πραγματική ετικέτα τέλους (χωρίς περιεχόμενο ενδιάμεσα στις ετικέτες). Παράδειγμα: το
 θα γινόταν είτε
 είτε
</BR> (χωρίς να υπάρχει κάτι ενδιάμεσα).

Οι ετικέτες πρέπει να είναι κατάλληλα τοποθετημένες

Τα επικαλυπτόμενα στοιχεία δεν επιτρέπονται. Ένα στοιχείο πρέπει να έχει μια ετικέτα τέλους πριν από την ετικέτα έναρξης του επόμενου στοιχείου και να είναι ενθετημένα το ένα μέσα στο άλλο κατάλληλα:

This text is bold and italic Λάθος

This text is bold and italic Σωστό

Οι XML ετικέτες διακρίνουν τα κεφαλαία γράμματα από τα μικρά

Τα παρακάτω προσδιορίζουν διαφορετικά στοιχεία:

<City> <CITY> <city>

<City>Λάθος</city>

<city>Σωστό</city>

Τα XML έγγραφα πρέπει να έχουν ένα στοιχείο ρίζας

Όλα τα XML έγγραφα πρέπει να περιέχουν ένα ενιαίο, μοναδικό ζευγάρι από ετικέτες για να καθορίσουν το στοιχείο ρίζας (root element). Όλα τα άλλα στοιχεία πρέπει να τοποθετηθούν ως ένθετα μέσα στο στοιχείο ρίζας. Όλα τα στοιχεία μπορούν να έχουν (ένθετα) στοιχεία παιδιά (child elements). Τα στοιχεία παιδιά πρέπει να είναι σε ζευγάρια και σωστά τοποθετημένα μέσα στο γονικό τους στοιχείο.

<root>

<child>

<subchild>

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

Οι τιμές των ιδιοτήτων πρέπει πάντα να βρίσκονται σε εισαγωγικά

Ένα στοιχείο μπορεί προαιρετικά να περιέχει μια ή περισσότερες ιδιότητες στην ετικέτα έναρξής του. Μια ιδιότητα είναι ένα ζευγάρι της μορφής «όνομα-τιμή» που χωρίζεται από ένα σύμβολο ισότητας (=). Οι τιμές μιας ιδιότητας πρέπει πάντα να βρίσκονται σε εισαγωγικά.

<CITY ZIP="01085">Westfield</CITY>

`ZIP="01085"` είναι η ιδιότητα του στοιχείου `<CITY>`.

Οι ιδιότητες χρησιμοποιούνται για να συνδέσουν πρόσθετες, δευτερεύουσες πληροφορίες με ένα στοιχείο. Οι ιδιότητες μπορούν επίσης να δεχτούν προκαθορισμένες τιμές, ενώ τα στοιχεία δεν μπορούν. Κάθε ιδιότητα ενός στοιχείου μπορεί να προσδιοριστεί μόνο μια φορά, αλλά με οποιαδήποτε σειρά.

```
<message date="12/11/99"> Σωστό
<message date=12/11/99>   Λάθος
<message ID="100">
```

Η ιδιότητα ID μπορεί να χρησιμοποιηθεί για να προσδιορίσει το μήνυμα (message)

```
<message ID="101">
```

Όλες οι τιμές των ιδιοτήτων πρέπει να είναι σε εισαγωγικά. Μπορούν να χρησιμοποιηθούν τα μονά εισαγωγικά (η απόστροφος) εάν η τιμή περιέχει έναν χαρακτήρα διπλού εισαγωγικού, και αντίστροφα. Εάν οι χαρακτήρες των εισαγωγικών περιέχονται στα δεδομένα, θα πρέπει στη θέση τους να χρησιμοποιηθούν οι οντότητες ' (στην περίπτωση απλού εισαγωγικού) ή " (στην περίπτωση διπλού εισαγωγικού).

Ειδικοί χαρακτήρες

Τα δεδομένα δεν πρέπει να περιέχουν δεσμευμένους χαρακτήρες σήμανσης, όπως για παράδειγμα τους χαρακτήρες `<` ή `&`. Οι χαρακτήρες αυτοί θα πρέπει να αντικατασταθούν από τις οντότητες `<` ή `&` αντίστοιχα, ενώ η ακολουθία `]]>` μπορεί μόνο να εμφανιστεί στο τέλος του τμήματος CDATA. Σε περίπτωση που χρησιμοποιείται για οποιοδήποτε άλλο σκοπό πρέπει να δοθεί ως `]]>`.

2.6 Ασκήσεις

Άσκηση 1 (Θέμα 2-δ εξετάσεων Φεβρουαρίου 2002)

Έστω <maths> μια ετικέτα ενός XML αρχείου. Να δοθούν 2 εναλλακτικοί τρόποι σύνταξης της ετικέτας, όταν η τιμή της είναι: x<y & y>z.

Λύση

Η δήλωση <maths> x<y & y>z </maths> είναι λανθασμένη συντακτικά, καθότι περιέχει τους δεσμευμένους (*reserved*) χαρακτήρες <, > και &. Αυτό θα οδηγούσε σε λάθος κατά την επεξεργασία του XML εγγράφου από έναν XML αναλυτή (parser). Αυτό, μπορεί να αποφευχθεί είτε με την αντικατάσταση των χαρακτήρων αυτών με χρήση αναφορών σε οντότητες (*entity references*) είτε με τη χρήση τμημάτων CDATA.

α΄ τρόπος (χρήση αναφορών σε οντότητες)

Η XML παρέχει μια σειρά από προκαθορισμένους χαρακτήρες οντότητες (*predefined character entities*) προκειμένου να αντικαταστήσει τους χαρακτήρες που είναι δεσμευμένοι από την XML. Οι χαρακτήρες αυτοί, μαζί με τις αντίστοιχες τιμές τους φαίνονται στον παρακάτω πίνακα

Όνομα	Τιμή
amp	&
apos	‘
gt	>
Lt	<
Quot	“

Γενικά, όταν θέλουμε να αναφερθούμε σε μια οντότητα χρησιμοποιούμε την εξής δήλωση: “*&όνομα;*” .Η σωστή λοιπόν σύνταξη με τη χρήση οντοτήτων είναι η εξής:

```
<maths> x&lt;y &amp; y&gt;z </maths>
```

β΄ τρόπος (χρήση τμημάτων CDATA)

Ο δεύτερος τρόπος για τη δήλωση της παραπάνω τιμής είναι με τη χρήση τμημάτων CDATA. Η σύνταξη των τμημάτων αυτών γίνεται ως εξής:

```
<![CDATA[κείμενο]]>
```

Το κείμενο μπορεί να περιέχει οποιουσδήποτε χαρακτήρες (συμπεριλαμβανομένου και τους δεσμευμένους χαρακτήρες <, > και &), αλλά θα αγνοούνται κατά την ανάλυσή τους. Συνεπώς, η εισαγωγή ενός CDATA τμήματος στο παράδειγμά μας γίνεται ως εξής:

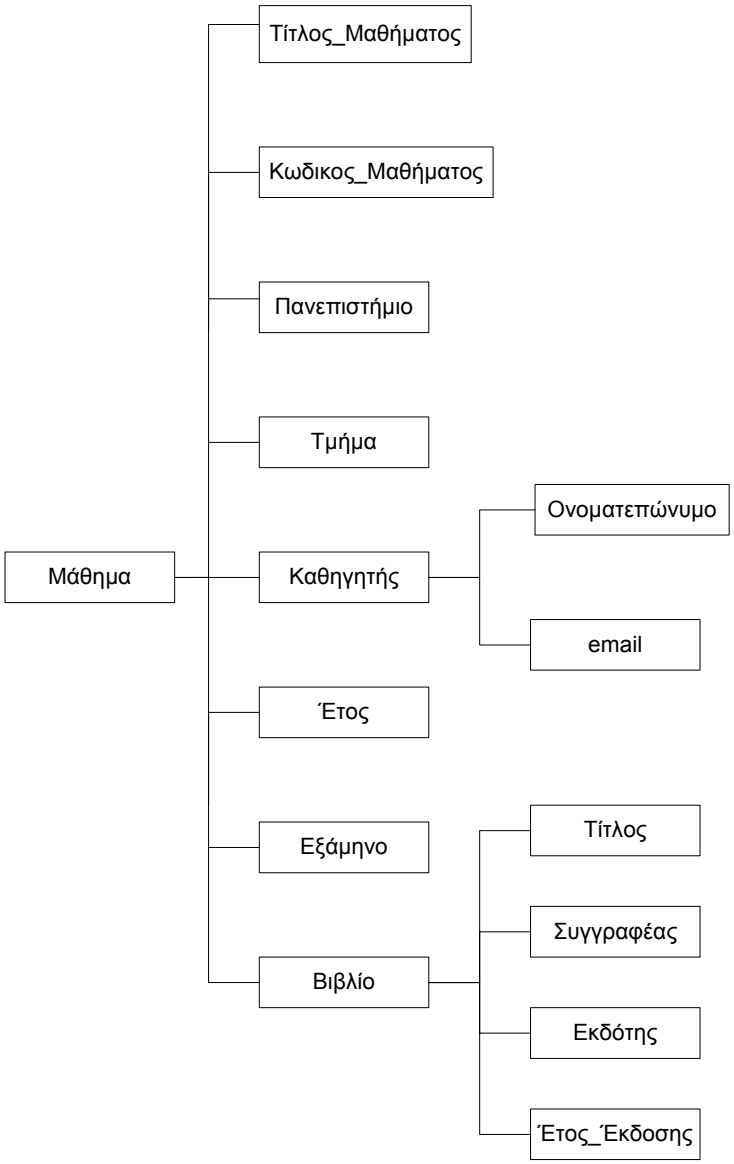
```
<maths> <![CDATA[x<y & y>z]]> </maths>
```

Άσκηση 2 (Θέμα 3 – (α) εξετάσεων Φεβρουαρίου 2002)

Έστω ότι για την περιγραφή των πανεπιστημιακών μαθημάτων είναι απαραίτητα τα παρακάτω πεδία: Τίτλος μαθήματος, κωδικός μαθήματος, πανεπιστήμιο, τμήμα, καθηγητής, έτος, εξάμηνο και βιβλίο που διανέμεται. Τα στοιχεία «καθηγητής» και «βιβλίο» αποτελούνται από επιπλέον πεδία που τα περιγράφουν πλήρως. Για τον καθηγητή είναι απαραίτητα τουλάχιστον το Ονοματεπώνυμο και το e-mail του καθώς για την περιγραφή του βιβλίου είναι απαραίτητα ο τίτλος, ο συγγραφέας, ο εκδότης και το έτος έκδοσης.

(α) Να δοθεί πλήρης περιγραφή και δενδρική αναπαράσταση της παραπάνω δομής και να αναφερθούν όλες οι παραδοχές που έγιναν για τον τύπο των δεδομένων και για την συχνότητα εμφάνισης του κάθε πεδίου.

(α) Η δενδρική αναπαράσταση της δομής που προτείνεται για την περιγραφή των πανεπιστημιακών μαθημάτων παρουσιάζεται στην επόμενη εικόνα:



Στη δενδρική αυτή αναπαράσταση το στοιχείο ρίζας είναι το στοιχείο *Μάθημα*, τα στοιχεία φύλλα (leaves) του δένδρου είναι τα στοιχεία: *Τίτλος_Μαθήματος*, *Κωδικός_Μαθήματος*, *Πανεπιστήμιο*, *Τμήμα*, *Έτος*, *Εξάμηνο*, *Ονοματεπώνυμο*, *email*, *Τίτλος*, *Συγγραφέας*,

Εκδότης, *Έτος_Έκδοσης*, και τα κλαδιά του δένδρου (branches) είναι τα στοιχεία *Καθηγητής*, *Βιβλίο*. Για λόγους απλότητας κάνουμε την παραδοχή ότι ο τύπος δεδομένων για όλα τα στοιχεία φύλλα του δένδρου θα είναι τύπου string. Επίσης γίνεται η υπόθεση ότι όλα τα στοιχεία του δένδρου είναι απαραίτητα, η συχνότητα εμφάνισής τους είναι μια φορά, εκτός από τα στοιχεία *Ονοματεπώνυμο* και *email*, όπου βρίσκονται ενθετημένα στο στοιχείο καθηγητής, και απαιτείται να εμφανίζονται τουλάχιστον μια φορά. Όλα τα στοιχεία πρέπει να εμφανίζονται με τη σειρά που φαίνεται στη δενδροειδή αναπαράσταση

Ένα απλό XML αρχείο του οποίου η δομή ακολουθεί την παραπάνω δενδρική αναπαράσταση είναι το εξής:

```
<?xml version="1.0" encoding="UTF-8"?>
<Μάθημα>
	<Τίτλος_Μαθήματος>ΑΠΟΘΗΚΕΣ ΚΑΙ ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ</Τίτλος_Μαθήματος>
	<Κωδικός_Μαθήματος>ΤΕ256</Κωδικός_Μαθήματος>
	<Πανεπιστήμιο>Πανεπιστήμιο Πειραιώς</Πανεπιστήμιο>
	<Τμήμα>Διδακτική της Τεχνολογίας και Ψηφιακών Συστημάτων</Τμήμα>
	<Καθηγητής>
		<Ονοματεπώνυμο>Δρ. Δημήτριος Σάμψων</Ονοματεπώνυμο>
		<email>sampson@iti.gr</email>
	</Καθηγητής>
	<Έτος>3</Έτος>
	<Εξάμηνο>5</Εξάμηνο>
	<Βιβλίο>
		<Τίτλος>Οδηγός της XML με παραδείγματα</Τίτλος>
		<Συγγραφέας>Benoit Marchal</Συγγραφέας>
		<Εκδότης>Β. Γκιούρδας Εκδοτική</Εκδότης>
		<Έτος_Έκδοσης>2000</Έτος_Έκδοσης>
	</Βιβλίο>
</Μάθημα>
```

Θα πρέπει να σημειωθεί ότι στο παραπάνω XML έγγραφο απουσιάζει η δήλωση του τύπου εγγράφου. Πιο αναλυτικό XML παράδειγμα θα δοθεί στο στις ασκήσεις του κεφαλαίου 3.

Άσκηση 4

Η φόρμα του δελτίου δανεισμού βιβλίων από τη βιβλιοθήκη του πανεπιστημίου έχει την εξής μορφή

ΔΕΛΤΙΟ ΔΑΝΕΙΣΜΟΥ ΒΙΒΛΙΩΝ

Παρακαλούμε συμπληρώστε τα στοιχεία του βιβλίου

ΤΑΞΙΝΟΜΙΚΟΣ ΑΡΙΘ.: ……

ΣΥΓΓΡΑΦΕΑΣ: ……..

ΤΙΤΛΟΣ: ……..

ΗΜΕΡΟΜΗΝΙΑ: …….

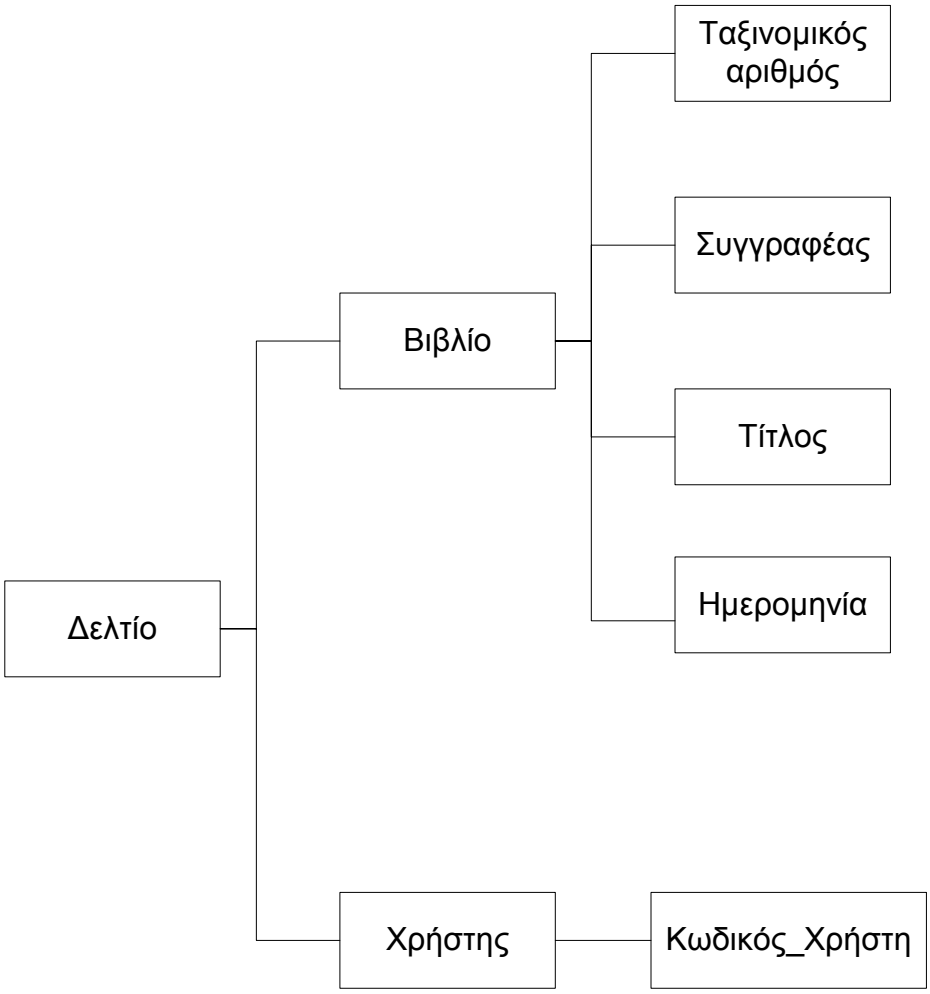
και τον προσωπικό σας κωδικό στην Βιβλιοθήκη

ΚΩΔΙΚΟΣ ΧΡΗΣΤΗ:……

(α) Να σχεδιαστεί η δενδρική αναπαράσταση του παραπάνω δελτίου και να γίνουν οι παραδοχές για τον τύπο δεδομένων του κάθε πεδίου και για τη συχνότητα εμφάνισής τους.

Λύση

(α) Η δενδρική αναπαράσταση που υλοποιεί την παραπάνω φόρμα φαίνεται στο ακόλουθο σχήμα.



Ο ταξινομικός αριθμός και ο κωδικός χρήστη είναι μοναδικά ορισμένοι αριθμοί. Δηλαδή, κάθε χρήστης και κάθε βιβλίο έχει από έναν μοναδικό αριθμό που τα χαρακτηρίζει. Η XML παρέχει την ιδιότητα ID προκειμένου να αναπαραστήσει μοναδικά ορισμένη πληροφορία.

Ένα απλό XML αρχείο του οποίου η δομή ακολουθεί την παραπάνω δενδρική αναπαράσταση είναι το εξής:

```
<?xml version="1.0" encoding="UTF-8"?>
<Δελτίο>
      <Βιβλίο>
            <Ταξινομικός_Αριθμός>Π278</Ταξινομικός_Αριθμός>
            <Συγγραφέας>Micheline Kamber</Συγγραφέας>
            <Συγγραφέας>Jiawei Han</Συγγραφέας>
            <Τίτλος>Data Mining: Concepts and Techniques</Τίτλος>
            <Ημερομηνία>2000</Ημερομηνία>
      </Βιβλίο>
      <Χρήστης>
            <Κωδικός_Χρήστη>Ε00023</Κωδικός_Χρήστη>
      </Χρήστης>
</Δελτίο>
```

Ένας δεύτερος τρόπος για να αναπαρασταθεί το παραπάνω δενδρικό διάγραμμα σε XML είναι η χρήση των κόμβων Κωδικός_Χρήστη και Ταξινομικός_Αριθμός ως ιδιότητες στα στοιχεία Χρήστης και Βιβλίο. Το XML έγγραφο θα είναι ως εξής:

```
<?xml version="1.0" encoding="UTF-8"?>
<Δελτίο>
 <Βιβλίο Ταξινομικός_Αριθμός="Π278">
      <Συγγραφέας>Micheline Kamber</Συγγραφέας>
      <Συγγραφέας>Jiawei Han</Συγγραφέας>
      <Τίτλος>Data Mining: Concepts and Techniques</Τίτλος>
      <Ημερομηνία>2000</Ημερομηνία>
 </Βιβλίο>
 <Χρήστης Κωδικός_Χρήστη="Ε00023"/>
</Δελτίο>
```

3 Μοντελοποίηση XML εγγράφων

3.1 Μοντελοποίηση Εγγράφων

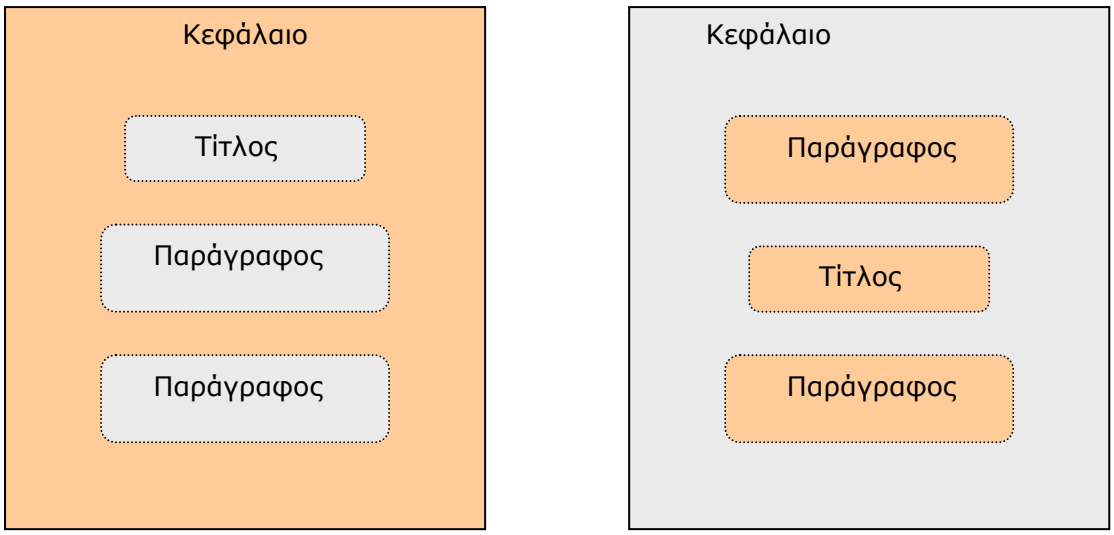
Ένα από τα πιο δυναμικά χαρακτηριστικά της XML είναι ότι επιτρέπει στο χρήστη να δημιουργήσει τη δική του γλώσσα σήμανσης, δίνοντας του τη δυνατότητα να καθορίσει τα στοιχεία και τις ιδιότητες που ταιριάζουν στην πληροφορία που θέλει να διαχειριστεί, παρά το γεγονός ότι τον περιορίζει σε μια γλώσσα «γενικού σκοπού». Αλλά αυτό που απουσιάζει από τη γλώσσα XML, και το οποίο επιτυγχάνεται με ένα μοντέλο εγγράφου, είναι ο καθορισμός της γλώσσας με τυποποιημένο τρόπο έτσι ώστε να περιοριστεί το λεξιλόγιο των στοιχείων και των ιδιοτήτων σε μια πιο «διαχειρίσιμη» βάση. Αυτό, έχει ως αποτέλεσμα να ελεγχθεί καλύτερα η γραμματική του XML εγγράφου.

Στην μοντελοποίηση των εγγράφων διασαφηνίζονται τα έγγραφα τα οποία μπορούν να παραχθούν με μια πρότυπη γλώσσα. Δηλαδή το μοντέλο του εγγράφου καθορίζει ποια κείμενα είναι συμβατά με τη γλώσσα. Ερωτήσεις του τύπου όπως: «Μπορεί αυτό το στοιχείο να πάρει τίτλο;» απαντώνται μέσω της μοντελοποίησης.

παράδειγμα

Όταν υπάρχει ένα μοντέλο εγγράφου του τύπου όπως το παραπάνω, τότε για να αποτελεί μοντέλο αυτό το έγγραφο για κάθε XML έγγραφο πρέπει να είναι ένα «ειδικό είδος εγγράφου» γραμμένο με σύνταξη σχεδιασμένη να «περιγράφει» γλώσσες βασισμένες στην XML, το οποίο θέτει με σαφήνεια στις παραμέτρους της γραμματικής και του λεξιλογίου για μια μοναδική γλώσσα σήμανσης. Έτσι με αυτόν τον τρόπο μπορεί να επιβεβαιωθεί πότε ένα XML έγγραφο αντιστοιχεί στον τύπο του προκαθορισμένου εγγράφου που έχει τεθεί.

Παράδειγμα



Έγκυρο ***Άκυρο***

3.2 Ανάγκη για μοντελοποίηση εγγράφων

Αν και ένα μοντέλο εγγράφου κάνει τον τρόπο εργασίας ευκολότερο πάνω σε ορισμένο αριθμό εγγράφων, δεν είναι σίγουρο ότι ισχύει το ίδιο και για περισσότερα από αυτά, όταν βέβαια οι ανάγκες για ποιότητα είναι υψηλές.

Γενικά πάντως η μοντελοποίηση εγγράφων είναι σημαντική όταν:

1. Τα έγγραφα γράφονται από ανθρώπους και χρησιμοποιούνται ως δεδομένα εισόδου για κάποιο λογισμικό (software). Χωρίς τη μοντελοποίηση το λογισμικό είναι ασταθές και όχι πάντα συμβατό στις μορφές (formats) των δεδομένων, διότι είναι δύσκολος ο σχεδιασμός προγραμμάτων τα οποία δουλεύουν με «παραλλαγές». Περιορίζοντας έτσι τον τρόπο εγγραφής σε μία πιο προβλεπόμενη μορφή, κάνει τον σχεδιασμό του λογισμικού πιο εύκολο και μειώνει την πιθανότητα των λαθών.
2. Το έγγραφο απαιτεί αξιόπιστα σύνολα καταχωρήσεων. Για παράδειγμα όταν μία φόρμα παραγγελίας ενός προϊόντος ζητά να δηλωθεί η ταχυδρομική διεύθυνση, δηλαδή να γίνει γνωστό πού θα σταλεί το προϊόν, μπορεί να χρησιμοποιηθεί το μοντέλο εγγράφου που βεβαιώνει ότι όλα τα απαιτούμενα σύνολα καταχωρήσεων είναι δηλωμένα.
3. Ζητούνται φόρμες εγγράφων από άτομα ξένα προς τη γλώσσα. Από τη στιγμή που η γλώσσα είναι από μόνη της ένα είδος εγγράφου, μπορεί να γίνει δημόσια πηγή εξόρυξης. Ένα μοντέλο εγγράφου μπορεί να παραχθεί ως δεδομένο για δομημένα κατασκευαστικά περιβάλλοντα, όπως ένα πρόγραμμα σύνταξης (editor) της XML. Σε κάθε πρόγραμμα ο editor μπορεί αυτόματα να εφοδιάσει με τα απαιτούμενα σύνολα καταχωρήσεων και να οδηγήσει τον σχεδιαστή-προγραμματιστή παρέχοντάς του λίστες από κατάλληλα στοιχεία.
4. Υπάρχει ανάγκη για μία σταθερή και μόνιμη πλαισίωση μιας αναπτυσσόμενης νέας γλώσσας ή και οικογένειας γλωσσών. Ένα μοντέλο εγγράφου παρέχει ένα εύκολο τρόπο για να δημιουργηθούν πρότυπα, όπως η HTML 4.0. Η διατήρηση των εκδόσεων κάθε γλώσσας αποτελεί στοιχείο για να διατηρείται το λογισμικό της XML, αφού παλιά προγράμματα μπορεί να είναι ασύμβατα με νέες εκδόσεις.

3.3 Μοντελοποίηση XML εγγράφων με χρήση της τεχνολογίας DTD

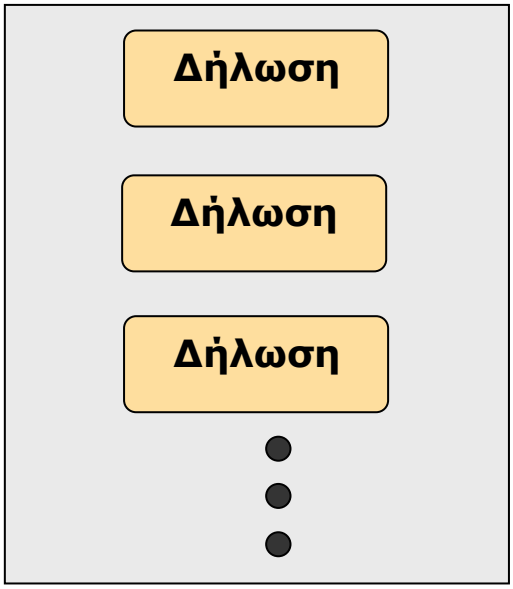
3.3.1 Εισαγωγή στα DTD

Ένα DTD παρέχει στις εφαρμογές τη χρήσιμη πληροφορία για το ποια ονόματα και δομές μπορούν να χρησιμοποιηθούν σε ένα συγκεκριμένο τύπο εγγράφου. Η χρήση ενός DTD κατά τη σύνταξη εγγράφων σημαίνει ότι μπορεί να βεβαιωθεί ότι όλα τα έγγραφα που ανήκουν σε αυτόν τον συγκεκριμένο τύπο θα δομηθούν και θα ονομαστούν σύμφωνα με ένα συνεπή και ομοιόμορφο τρόπο. Τα DTDs είναι λιγότερο σημαντικά για την επεξεργασία των εγγράφων που είναι γνωστό ότι είναι ήδη καλά ορισμένα, αλλά απαιτούνται ακόμα, για να εκμεταλλευτούν τους ειδικούς τύπους των ιδιοτήτων που χρησιμοποιούνται στην XML όπως είναι οι ενσωματωμένοι μηχανισμοί για την αναφορά παραπομπής ID/IDREF, ή τη χρήση των προεπιλεγμένων ιδιοτήτων.

Η XML παρέχει επίσης έναν τρόπο διαμοιρασμού των δεδομένων ανεξάρτητο από κάθε εφαρμογή. Με ένα DTD, ανεξάρτητες ομάδες ανθρώπων μπορούν να συμφωνήσουν να χρησιμοποιήσουν ένα κοινό DTD για την ανταλλαγή δεδομένων.

Τα DTDs είναι απλοϊκά και περιορισμένα. Ο διάδοχος των DTDs, το XML Schema, θα επιτρέψει στους σχεδιαστές να προσδιορίσουν τους δικούς τους τύπους δεδομένων (datatypes) για τα στοιχεία και τις ιδιότητες. Δεδομένου ότι το XML Schema είναι ακόμα μια προδιαγραφή (specification) δεν έχει καθοριστεί ακόμα σε τελική μορφή, και οι κατασκευαστές λογισμικού είναι προτιμότερο να χρησιμοποιούν τα DTDs, τα οποία αποτελούν πρότυπο (standard) και υποστηρίζονται πρακτικά από όλα τα εργαλεία της XML.

Κάθε DTD αποτελείται από ένα σύνολο από κανόνες ή δηλώσεις. Κάθε δήλωση προσθέτει ένα νέο στοιχείο, σύνολο από ιδιότητες, οντότητες (entities), ή σημειώσεις (notations) που πρόκειται να χρησιμοποιηθούν σε ένα έγγραφο XML. Τα DTD μπορούν να συνδυαστούν με τη χρήση παραμετρικών οντοτήτων (parameter entities), μια τεχνική που ονομάζεται παραμετροποίηση (modularization). Μπορούν επίσης να προστεθούν δηλώσεις μέσα στο εσωτερικό υποσύνολο του κειμένου.



Η σειρά των δηλώσεων είναι σημαντική σε δυο περιπτώσεις. Στην πρώτη περίπτωση, εάν υπάρχουν πλεονάζουσες δηλώσεις (όταν υπάρχει, για παράδειγμα, η δήλωση του ίδιου ονόματος ενός στοιχείου πάνω από μια φορά), επικρατεί η πρώτη δήλωση που εμφανίζεται και οι υπόλοιπες αγνοούνται. Στη δεύτερη περίπτωση, εάν χρησιμοποιούνται

παραμετρικές οντότητες (parameter entities), αυτές πρέπει να δηλώνονται πριν τη χρήση της αναφοράς τους.

Η σύνταξη των δηλώσεων είναι ευανάγνωστη όταν χρησιμοποιούνται κενά διαστήματα. Η πρόσθεση επιπλέον διαστημάτων μπορεί να γίνει οπουδήποτε, εκτός από την αλφαριθμητική σειρά των χαρακτήρων στην αρχή της δήλωσης, όπου προσδιορίζεται ο τύπος της δήλωσης. Όλες οι παρακάτω δηλώσεις είναι αποδεκτές:

```
<!ELEMENT CUSTOMER ALL>

<!ELEMENT

CUSTOMER

ALL>

<!ELEMENT ΧΡΩΜΑ (ΠΡΑΣΙΝΟ|

                 ΚΟΚΚΙΝΟ|

                 ΜΠΛΕ  )*>
```

3.3.2 Δημιουργία DTD

Για τη δημιουργία ενός DTD πρέπει να χρησιμοποιηθεί η δηλωτική σύνταξη της XML (πολύ απλή: οι κωδικές λέξεις δήλωσης αρχίζουν με <! παρά με το < και ο τρόπος με τον οποίο διαμορφώνονται οι δηλώσεις διαφέρει ελαφρώς). Το παρακάτω είναι ένα παράδειγμα ενός DTD για την ενοικίαση αυτοκινήτων:

```
<!ELEMENT CAR_RENTAL (CUSTOMER, RENTAL)>
<!ELEMENT CUSTOMER (#PCDATA)>
<!ELEMENT RENTAL (#PCDATA)>
```

Επειδή δεν υπάρχει κανένα άλλο στοιχείο που να περιέχει το CAR_RENTAL, αυτό το στοιχείο υποτίθεται ότι είναι το στοιχείο ρίζας (root element), το οποίο εσωκλείει όλα τα άλλα στο έγγραφο. Μπορεί τώρα το DTD να χρησιμοποιηθεί για τη δημιουργία ενός XML αρχείου, αρκεί να δοθούν οι ακόλουθες δηλώσεις:

```
<?xml version="1.0"?>
<!DOCTYPE CAR_RENTAL "RENTAL.dtd">
```

Υποτίθεται ότι το DTD τοποθετήθηκε στο αρχείο RENTAL.dtd. Τώρα το πρόγραμμα σύνταξης της XML μπορεί να επιτρέψει τη δημιουργία αρχείων XML σύμφωνα με την παρακάτω μορφή:

```
<CAR_RENTAL>
  <CUSTOMER>George Papas</CUSTOMER>
  <RENTAL>BMW Z3</RENTAL>
</CAR_RENTAL>
```

Η δήλωση ενός στοιχείου ρίζας δε μπορεί να γίνει ρητά σε ένα DTD. Αυτό γίνεται στη δήλωση του τύπου εγγράφου, όχι στο DTD. Μια δήλωση τύπου εγγράφου γίνεται όπως παρακάτω:

```
<!DOCTYPE CAR_RENTAL SYSTEM "RENTAL.dtd">
```

Το DTD για να χρησιμοποιηθεί μέσα σε ένα XML έγγραφο απαιτείται να δηλωθεί με μια DOCTYPE δήλωση αμέσως μετά από την XML δήλωση που περιλαμβάνεται μέσα σε ένα έγγραφο XML. Όταν υπάρχει όμως απομονωμένο το DTD χωρίς να αναφέρεται σε κάποιο XML έγγραφο, τότε αυτή η δήλωση του DOCTYPE δεν είναι απαραίτητη.

Εσωτερικά DTDs

Εάν το DTD συμπεριλαμβάνεται στο αρχείο πηγής XML, πρέπει να περιληφθεί σε έναν DOCTYPE καθορισμό με την ακόλουθη σύνταξη:

Εάν το DTD συμπεριλαμβάνεται στο αρχείο πηγής XML, πρέπει να περιληφθεί σε έναν DOCTYPE καθορισμό με την ακόλουθη σύνταξη:

```
<!DOCTYPE root-element [element-declarations]>
```

Ένα παράδειγμα ενός XML εγγράφου με εσωτερικό DTD είναι το παρακάτω:

```
<?xml version="1.0"?>
<!DOCTYPE CAR_RENTAL [
      <!ELEMENT CAR_RENTAL (CUSTOMER, RENTAL)>
      <!ELEMENT CUSTOMER (#PCDATA)>
      <!ELEMENT RENTAL (CAR, INSURANCE)>
      <!ELEMENT CAR (#PCDATA)>
      <!ELEMENT INSURANCE EMPTY>
]>
<CAR_RENTAL>
      <CUSTOMER>Mike's Store</CUSTOMER>
      <RENTAL>
            <CAR>BMW Z3</CAR>
            <INSURANCE/>
      </RENTAL>
</CAR_RENTAL>
```

το παραπάνω DTD ερμηνεύεται ως εξής:

`!DOCTYPE CAR_RENTAL` (στη γραμμή 2) προσδιορίζει ότι το στοιχείο ρίζας του εγγράφου είναι το στοιχείο `CAR_RENTAL`.

`!ELEMENT CAR_RENTAL` (στη γραμμή 3) προσδιορίζει ότι το στοιχείο `CAR_RENTAL` έχει δυο child elements: (`CUSTOMER, RENTAL`).

`!ELEMENT CUSTOMER` (στη γραμμή 4) προσδιορίζει ότι το στοιχείο `CUSTOMER` θα είναι του τύπου `#PCDATA`.

`!ELEMENT RENTAL` (στη γραμμή 4) προσδιορίζει ότι το στοιχείο `RENTAL` θα έχει δυο στοιχεία παιδιά (child elements) (`CAR, INSURANCE`). Και ούτω καθεξής...

Στην πρώτη γραμμή το `<!DOCTYPE`, δηλώνει ότι αυτό είναι η αρχή ενός καινούριου DTD και ότι το στοιχείο του εγγράφου του XML αρχείου ονομάζεται `CAR_RENTAL`. Η δήλωση DTD εσωκλείεται μέσα σε αυτήν την ετικέτα και τελειώνει με το κλείσιμο της ετικέτας `>`. Μόνο η περίπτωση μιας δήλωσης DTD μπορεί να εμφανιστεί σε αυτή τη δήλωση αλλά είναι δυνατό να περιληφθούν και εξωτερικοί ορισμοί. Το πραγματικό μέρος του DTD περιέχει τους ορισμούς με τη διάταξη των στοιχείων.

Εξωτερικά DTD

Η χρησιμοποίηση ενός εσωτερικού ορισμού είναι πρακτική όταν υπάρχουν μόνο μερικά έγγραφα τα οποία είναι διαθέσιμα offline, δεδομένου ότι ο ορισμός του DTD βρίσκεται πάντα μέσα στο αρχείο. Όμως, εάν παραδείγματος χάριν, το DTD ορίζει ένα πρωτόκολλο XML που χρησιμοποιείται για την επικοινωνία δυο ξεχωριστών συστημάτων, η μετάδοση του DTD μαζί με κάθε έγγραφο προσθέτει επιπλέον πληροφορία (overhead) κατά την επικοινωνία. Η ύπαρξη ενός εξωτερικού DTD εξαλείφει την ανάγκη να στέλνεται αυτό εκ νέου κάθε φορά. Θα μπορούσε να αφαιρεθεί το DTD από το έγγραφο, και να τοποθετηθεί σε ένα αρχείο DTD το οποίο θα βρίσκεται σε έναν υπολογιστή εξυπηρέτησης Ιστού που θα είναι προσιτός και από τα δύο συστήματα.

Εάν το DTD είναι εξωτερικό στο πηγαίο XML αρχείο, πρέπει να περιληφθεί με έναν καθορισμό DOCTYPE με την ακόλουθη σύνταξη:

```
<!DOCTYPE root-element SYSTEM "filename">
```

Το παρακάτω είναι το ίδιο XML έγγραφο με αυτό που αναφέρθηκε παραπάνω, στα εσωτερικά DTDs, αλλά με ένα εξωτερικό DTD:

```
<?xml version="1.0"?>

<!DOCTYPE root-element SYSTEM " CAR_RENTAL.dtd">
```

```
<CAR_RENTAL>
      <CUSTOMER>Mike's Store</CUSTOMER>
      <RENTAL>
            <CAR>BMW Z3</CAR>
            <INSURANCE/>
      </RENTAL>
</CAR_RENTAL>
```

Το παρακάτω είναι ένα αντίγραφο του αρχείου CAR_RENTAL.dtd που περιλαμβάνει το DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT CAR_RENTAL (CUSTOMER, RENTAL)>
<!ELEMENT CUSTOMER (#PCDATA)>
<!ELEMENT RENTAL (CAR, INSURANCE)>
<!ELEMENT CAR (#PCDATA)>
<!ELEMENT INSURANCE EMPTY>
```

3.3.3 Δηλώσεις Στοιχείων

Το πρώτο και σημαντικότερο που πρέπει να εξεταστεί στην XML είναι το σύνολο από τα στοιχεία που περιέχει. Για κάθε στοιχείο που πρόκειται να χρησιμοποιηθεί στο έγγραφο, πρέπει να υπάρξει και η δήλωσή του στο DTD. Η δήλωση ενός στοιχείου χρησιμοποιείται για δυο σκοπούς: προσθέτει μια ονομασία για το στοιχείο και προσδιορίζεται το περιεχόμενο του. Λαμβάνοντας υπόψη τα παραπάνω, οι δηλώσεις των στοιχείων δημιουργούν μια γραμματική για τη γλώσσα της XML, ένα υπόδειγμα για να προσδιορίζεται πια έγγραφα είναι έγκυρα και πια όχι.

Στο όνομα του στοιχείου σημαντικό ρόλο παρουσιάζει ο τρόπος εμφάνισης του. Έτσι εάν ένα όνομα έχει τη μορφή FirstName στη μια δήλωση, πρέπει να γράφεται πάντα με αυτόν τον τρόπο, και όχι με τη μορφή FIRSTNAME ή firstname. Αυτή είναι μια μεγάλη διαφορά από την SGML, η οποία απλοποιεί τη συγγραφή λογισμικού που επεξεργάζεται η XML.

Στο DTD, τα XML στοιχεία δηλώνονται με μια απλή δήλωση των στοιχείων. Μια δήλωση ενός στοιχείου έχει την ακόλουθη σύνταξη:

```
<!ELEMENT όνομα_στοιχείου κατηγορία>
```

ή

```
<!ELEMENT όνομα_στοιχείου (στοιχεία-παιδιά)>
```

Υπάρχουν πέντε διαφορετικά είδη περιεχομένου που μπορούν να έχουν τα στοιχεία και οι οποίοι αναφέρονται στη συνέχεια:

Κενά Στοιχεία

Ο απλούστερος τύπος περιεχομένου που μπορεί να δηλωθεί είναι αυτό του κενού στοιχείου (empty element), ο οποίος αποτελείται από την κωδική λέξη EMPTY. Η δήλωση του γίνεται με τον εξής τρόπο:

```
<!ELEMENT όνομα_στοιχείου EMPTY>
```

Τα υποχρεωτικά στοιχεία πρέπει να προσδιορισθούν ακόμα και όταν το περιεχόμενό τους είναι κενό. Το ακόλουθο παράδειγμα XML παρουσιάζει μια εμφάνιση του στοιχείου INSURANCE όπου το περιεχόμενο του είναι κενό.

```
<!ELEMENT INSURANCE EMPTY>
```

Η αναπαράσταση του στο XML έγγραφο γίνεται είτε με τη δήλωση των ετικετών έναρξης και τέλους των στοιχείων χωρίς να περιέχεται τίποτα ενδιάμεσα ή με την ενιαία αναπαράστασή τους με την εμφάνιση μόνο της ετικέτας έναρξης και την πρόσθεση μιας κάθετου πριν το κλείσιμό της:

```
<INSURANCE></INSURANCE>
```

ή

```
<INSURANCE/>
```

Στοιχεία χωρίς περιορισμό περιεχομένου

Με αυτόν τον τύπο περιεχομένου δηλώνεται ότι ένα στοιχείο μπορεί να περιέχει οποιαδήποτε άλλα στοιχεία ή ακόμα και κείμενο. Χρησιμοποιείται η κωδική λέξη ANY:

```
<!ELEMENT όνομα_στοιχείου ANY>
```

Παράδειγμα:

```
<!ELEMENT NOTE ANY>
```

Τα στοιχεία που δηλώνονται με την κωδική λέξη ANY, μπορούν να περιέχουν οποιοδήποτε συνδυασμό αναλυόμενων δεδομένων.

Φυσικά, ένα στοιχείο που μπορεί να περιέχει οτιδήποτε είναι περιορισμένης αξίας σε ένα DTD, και αυτός ο τύπος περιεχομένου δεν ελέγχει τόσο τη δομή του εγγράφου όσο θα την έλεγχε ένας πιο αυστηρός τύπος. Όμως για γρήγορη προτυποποίηση εγγράφων, ίσως αναδειχθεί μια καλή λύση.

Στοιχεία που περιέχουν μόνο κείμενο (parsed data)

Για τα στοιχεία που περιέχουν μόνο κείμενο και όχι άλλα στοιχεία χρησιμοποιείται ο τύπος δεδομένων (#PCDATA):

```
<!ELEMENT όνομα-στοιχείου (#PCDATA)>
```

Παράδειγμα:

```
<!ELEMENT CAR (#PCDATA)>
```

Η κωδική λέξη (#PCDATA) στην πραγματικότητα αντιστοιχεί σε δεδομένα που αναλύονται από τον επεξεργαστή της XML, δηλαδή σε κείμενο το οποίο ελέγχεται για αναφορές οντοτήτων (entity references), οι οποίες τότε θα αντικατασταθούν από τις τιμές τους.

Στοιχεία που περιέχουν μόνο στοιχεία

Τα στοιχεία με ένα ή περισσότερα στοιχεία-παιδιά (child elements) ορίζονται με τα ονόματα των στοιχείων-παιδιών μέσα σε παρενθέσεις:

```
<!ELEMENT όνομα_στοιχείου (όνομα_στοιχείου_παιδιού)>
```

ή

```
<!ELEMENT όνομα_στοιχείου
 (όνομα_στοιχείου_παιδιού, όνομα_στοιχείου_παιδιού,.....)>
```

Παράδειγμα:

```
<!ELEMENT CAR_RENTAL (CUSTOMER, RENTAL)>
```

Όταν τα στοιχεία-παιδιά δηλώνονται σε μια ακολουθία που χωρίζεται από κόμματα, τα στοιχεία-παιδιά πρέπει να εμφανιστούν με την ίδια ακολουθία στο έγγραφο. Σε μια πλήρη δήλωση, τα στοιχεία-παιδιά πρέπει επίσης να δηλωθούν, και τα στοιχεία-παιδιά μπορούν επίσης να περιέχουν άλλα στοιχεία-παιδιά. Η πλήρης δήλωση του στοιχείου "CAR_RENTAL" θα είναι:

```
<!ELEMENT CAR_RENTAL (CUSTOMER, RENTAL)>
<!ELEMENT CUSTOMER (#PCDATA)>
<!ELEMENT RENTAL (CAR, INSURANCE)>
<!ELEMENT CAR (#PCDATA)>
<!ELEMENT INSURANCE EMPTY>
```

Στον τύπο περιεχομένου που περιέχει μόνο στοιχεία είναι δυνατό να εφαρμοστούν στα στοιχεία που δηλώνονται, ένα σύνολο από ειδικούς χαρακτήρες που τους προσδίδουν ορισμένα χαρακτηριστικά όπως φαίνεται και στον παρακάτω πίνακα. Οι παρακάτω δηλώσεις παρουσιάζουν μερικές μορφές που μπορεί να έχει αυτός ο τύπος:

```
<!ELEMENT   article   (para+)>
<!ELEMENT   article   (title, (para | sect1)+)>
<!ELEMENT   article   (title, subtitle?, ((para, sect1*) | sect1+))>
```

Οι τύποι περιεχομένων στους οποίους υπάρχουν στοιχεία, χρησιμοποιούν ένα ειδικό σύνολο συμβόλων, που δείχνουν ποια στοιχεία μπορούν να εμφανιστούν, πόσες φορές μπορεί να εμφανιστεί το καθένα και με ποια σειρά. Τα σύμβολα που χρησιμοποιούνται φαίνονται στον παρακάτω πίνακα:

Σύμβολο	Σημασία	Παράδειγμα
, (κόμμα)	Περιγράφει μια απαιτούμενη σειρά παραγόντων. Επίσης λειτουργεί και ως ο τελεστής ΚΑΙ (AND)	Α, Β σημαίνει ότι το Β πρέπει να ακολουθεί του Α. Το αντίθετο δεν είναι αποδεκτό
|	Περιγράφει κάτι εναλλακτικό. Ονομάζεται και τελεστής Η (OR)	Κόκκινο | κίτρινο | πράσινο σημαίνει ότι μια τιμή από αυτές είναι αποδεκτή. Ακριβώς μια επιλογή επιτρέπεται: όχι λιγότερες, ούτε περισσότερες. Είναι λάθος να χρησιμοποιηθεί το ίδιο όνομα πάνω από μια φορά σε μια τέτοια λίστα στοιχείων
(περιεχόμενο)	Ομαδοποιεί το περιεχόμενο, ώστε ένας από τους παραπάνω τελεστές να εφαρμοσθεί στο σύνολο. Οι παρενθέσεις μπορούν να εκτείνονται σε οποιοδήποτε βάθος	(Α | Β), C σημαίνει ότι ένα Α ή ένα Β πρέπει να ακολουθείται από ένα C, οπότε Α C και Β C είναι οι δυο επιτρεπόμενες ακολουθίες. Στην παράσταση Α | (Β, C) οι επιτρεπόμενες τιμές είναι το Α μόνο του, ή Β C
?	Καθιστά τα στοιχεία ή τις ομάδες αυτών ως προαιρετικά στοιχεία	Αυτό το σύμβολο σημαίνει ότι τα στοιχεία μπορούν να χρησιμοποιηθούν ή όχι, σύμφωνα με την κρίση του συντάκτη του εγγράφου
+	Απαιτεί ότι πρέπει να εμφανιστούν τουλάχιστο μια φορά τα στοιχεία (στοιχεία ή ομάδα στοιχείων) που υπάρχουν πριν το σύμβολο. Δεν	(cat | dog)+ σημαίνει ότι πρέπει να υπάρξει μια μη μηδενική τιμή από το σύνολο. Μπορούν για παράδειγμα να υπάρξουν τα εξής: cat ή dog dog dog cat

	υπάρχει όμως άνω όριο για το πόσα μπορεί να είναι	
*	Ορίζει ότι οποιοσδήποτε αριθμός των στοιχείων που υπάρχουν πριν από το σύμβολο μπορεί να εμφανιστεί, και είναι ο πιο αόριστος από όλους τους τελεστές	Customer* σημαίνει ότι αυτό το στοιχείο μπορεί να εμφανίσει καμία, ή από μια και πάνω φορές

3.3.4 Οντότητες – Entities

Υπάρχουν δύο είδη «οντοτήτων» (entities) που χρησιμοποιούνται στην XML. Οι «γενικές» (general) και οι «παραμετρικές» (parameter) οντότητες. Οι γενικές οντότητες χρησιμοποιούνται ευρέως διότι εμπεριέχονται σε αναλογία με τα XML έγγραφα, όμως και οι παραμετρικές οντότητες οι οποίες χρησιμοποιούνται σε DTD έγγραφα είναι επίσης διαθέσιμες και είναι πολύ ισχυρές.

Οι οντότητες είναι μεταβλητές που χρησιμοποιούνται για να προσδιορίσουν τη συντομότερη προσθήκη αναλυόμενων δεδομένων χαρακτήρων σε ένα έγγραφο. Συνήθως είναι κείμενο, αλλά μπορεί επίσης να περιέχει και δυαδικά δεδομένα. Οι οντότητες δηλώνονται στο DTD και στη συνέχεια γίνεται αναπαράσταση των δεδομένων με την αναφορά των οντοτήτων μέσα στο έγγραφο. Η αναφορά στις γενικές οντότητες ξεκινάει με & και τελειώνει με *;*, ενώ η αναφορά στις παραμετρικές οντότητες αρχίζει με % και τελειώνει με *;*. Για τις οντότητες που περιέχουν κείμενο, όταν γίνεται αναφορά σε αυτές η αναφορά αντικαθίσταται από το ίδιο το κείμενο όταν αναλύονται από έναν επεξεργαστή (parser) XML.

Με άλλα λόγια οι οντότητες δηλώνονται στο DTD και η αναφορά σε αυτές γίνεται, είτε στο περιεχόμενο του εγγράφου για τις γενικές οντότητες, είτε στο DTD για τις παραμετρικές οντότητες.

Οι οντότητες μπορεί να είναι εσωτερικές (internal) ή εξωτερικές (external). Μια εσωτερική οντότητα καθορίζεται ολότελα μέσα στο XML έγγραφο στο οποίο και αναφέρεται. Οι εξωτερικές οντότητες, από την άλλη, αντλούν το περιεχόμενό τους από εξωτερικές πηγές, όπως ένα αρχείο, και η αναφορά σε αυτά συνήθως περιέχει ένα URI στο οποίο ίσως βρεθούν. Οι οντότητες μπορούν επίσης να είναι αναλυόμενες (parsed) ή μη αναλυόμενες (unparsed) από τον επεξεργαστή της XML. Το περιεχόμενο των αναλυόμενων οντοτήτων είναι, καλά ορισμένο, XML κείμενο και οι μη αναλυόμενες οντότητες είναι δεδομένα τα οποία δεν είναι επιθυμητό να αναλυθούν, όπως συμπιεσμένο κείμενο ή δυαδικά δεδομένα. Στη συνέχεια θα αναφερθεί ο χειρισμός όλων των ειδών των οντοτήτων.

Εσωτερικές και εξωτερικές γενικές Οντότητες

Μπορούν να οριστούν οντότητες για κάθε περίσταση δηλώνοντάς τες στο DTD. Για να δηλωθεί μια οντότητα χρησιμοποιείται ο παράγοντας <!ENTITY>. Η δήλωση μιας γενικής οντότητας συντάσσεται με τον εξής τρόπο:

```
<!ENTITY όνομα_οντότητας "τιμή_οντότητας">

Παράδειγμα DTD:

<!ENTITY COLOUR "Μαύρο χρώμα">
<!ENTITY MODEL "Μαρτιος, 2000">

Παράδειγμα XML:
<CAR>&COLOUR; &MODEL;</CAR>
```

Όπου όνομα_οντότητας είναι το όνομα της οντότητας και τιμή_οντότητας είναι ο προσδιορισμός του ονόματος. Το όνομα χρησιμοποιείται για την αναφορά στην οντότητα και ο ορισμός μπορεί να πάρει πολλές διαφορετικές μορφές όπως θα αναφερθεί παρακάτω.

Ο πιο απλός και πιθανός προσδιορισμός οντότητας είναι το απλό κείμενο το οποίο θα αντικαθίσταται με την αναφορά της αντίστοιχης οντότητας όπως δείχνει και το παραπάνω παράδειγμα. Στη συνέχεια αναφέρονται ορισμένα στοιχεία σχετικά με της γενικές οντότητες:

- Υπάρχει η δυνατότητα «ένθεσης» (nesting) των αναφορών των γενικών οντοτήτων, όπως στο παρακάτω παράδειγμα

```
<!ENTITY ONOMA "Αθανασίου">
<!ENTITY ΣΥΓΓΡΑΦΕΑΣ "&ONOMA; Η οδός με τις Λεύκες"
```

- Οι αναφορές στης οντότητες δεν μπορούν να είναι κυκλικές, διότι ο επεξεργαστής της XML θα λειτουργεί ανώμαλα, όπως στο εξής παράδειγμα

```
<!ENTITY ONOMA "Αθανασίου &ΣΥΓΓΡΑΦΕΑΣ;"
<!ENTITY ΣΥΓΓΡΑΦΕΑΣ "&ONOMA; Η οδός με τις Λεύκες "
```

  Σε αυτήν την περίπτωση, όταν ο επεξεργαστής της XML προσπαθεί να αναλύσει την αναφορά &ONOMA; βρίσκει ότι στο κείμενο που αντιστοιχεί στην οντότητα ONOMA, χρειάζεται να αντικαταστήσει το κείμενο για την οντότητα ΣΥΓΓΡΑΦΕΑΣ, αλλά στο κείμενο που αντιστοιχεί στην οντότητα ΣΥΓΓΡΑΦΕΑΣ γίνεται αναφορά στην οντότητα ONOMA και έτσι δημιουργείται μια κυκλική κατάσταση. Οι αναφορές σε οντότητες που δημιουργούν κυκλικές καταστάσεις είναι μη επιτρεπτές σε έγκυρα έγγραφα XML.

- Αξίζει επίσης να σημειωθεί ότι δεν μπορούν αναφορές σε γενικές οντότητες να εισάγουν κείμενο που προορίζεται να χρησιμοποιηθεί μόνο σε περιεχόμενο DTD και όχι στο περιεχόμενο του XML εγγράφου. Ένα παράδειγμα που θεωρείται μη επιτρεπτό είναι το εξής:

```
<!ENTITY TAGS "(NAME, DATE, ORDERS)">
<!ELEMENT CUSTOMER &TAGS;>
```

Ο σωστός τρόπος για να γίνει αυτό είναι με της παραμετρικές οντότητες, οι οποίες θα αναφερθούν σε επόμενη παράγραφο.

Εκτός από τις εσωτερικές, οι οντότητες μπορούν να είναι και εξωτερικές, δηλαδή μπορεί να οριστεί ένα URI που θα καθοδηγεί τον επεξεργαστή της XML στην οντότητα. Μπορούν να χρησιμοποιηθούν αναφορές σε εξωτερικές οντότητες για την ενσωμάτωση αυτών των οντοτήτων στο έγγραφο. Μπορούν επίσης να δηλωθούν εξωτερικές οντότητες που να μην αναλύονται από τον επεξεργαστή της XML, δηλαδή να σχετίζονται με δυαδικά δεδομένα και να συνδέονται με ένα έγγραφο.

Οι εξωτερικές οντότητες μπορεί να είναι απλές γραμμές κειμένου, ολόκληρα κείμενα ή τμήματα κειμένων. Αυτό που έχει σημασία είναι ότι όταν εισάγονται στο περιεχόμενο ενός εγγράφου XML, για τον επεξεργαστή της XML αρκεί το κείμενο να είναι καλά δομημένο και έγκυρο.

Η δήλωση των εξωτερικών οντοτήτων γίνεται στο DTD χρησιμοποιώντας την κωδική λέξη SYSTEM ή PUBLIC. Οι οντότητες που δηλώνονται με τη κωδική λέξη SYSTEM είναι για ιδιωτική χρήση (π.χ: από παράγοντες οργανισμών), και οι οντότητες που δηλώνονται με τη λέξη κλειδί PUBLIC είναι για κοινή χρήση και χρειάζεται να ακολουθεί ένας «τυπικός δημόσιος προσδιοριστής» (Formal Public Identifier-FPI). Στη συνέχεια φαίνεται η χρήση των κωδικών λέξεων SYSTEM και PUBLIC για τη δήλωση μιας εξωτερικής οντότητας:

```
<!ENTITY ONOMA SYSTEM URI>
```

```
<!ENTITY ONOMA PUBLIC FPI URI>
```

Για παράδειγμα ας υποτεθεί ότι έχει αποθηκευτεί μια ημερομηνία όπως «Μάρτης, 2001» σε ένα αρχείο που ονομάζεται model.ent. Η τοποθέτηση μιας οντότητας που ονομάζεται MODEL και συνδέεται με αυτό το αρχείο γίνεται ως εξής

```
Παράδειγμα DTD:
<!ENTITY MODEL SYSTEM "model.ent">


Παράδειγμα XML:
<CAR>&MODEL;</CAR>
```

Η δυνατότητα αυτής της τεχνικής είναι αξιοσημείωτη διότι είναι δυνατή η δημιουργία εγγράφων από διαφορετικά τμήματα άλλων εγγράφων, που από μόνα τους μπορούν να αποτελέσουν ξεχωριστά έγγραφα. Αν χρειαστεί να χρησιμοποιηθεί μια «δημόσια» οντότητα αντί για μια «ιδιωτική», αυτό γίνεται με τη χρησιμοποίηση της κωδικής λέξης PUBLIC μαζί με ένα FPI όπως στο παράδειγμα:

```
<!ENTITY TODAY PUBLIC
"//starpowder//Custom Entity Version 1.0//EN" "model.ent">
```

Ο ορισμός εξωτερικών οντοτήτων, έχει ως αποτέλεσμα αυτές να μπορούν να γίνουν διαθέσιμες για πολλαπλά έγγραφα. Αυτό είναι χρήσιμο, για παράδειγμα, σε περιπτώσεις που χρειάζεται ένα κείμενο να εμφανίζεται πολλές φορές σε ένα ή σε διαφορετικά έγγραφα, ή σε ένα κείμενο που αλλάζει συχνά ώστε να γίνεται εύκολα η επέμβαση σε αυτό μια μόνο φορά.

Δημιουργία εγγράφων από τμήματα

Ένας τρόπος με τον οποίο μπορούν να χρησιμοποιηθούν οι γενικές οντότητες είναι στη δημιουργία εγγράφων από τμήματα, όπου κάθε τμήμα μεταχειρίζεται ως μια γενική οντότητα. Στο παρακάτω έγγραφο εμπεριέχεται μια οντότητα που αναφέρεται στο αρχείο `data.xml`:

```
<?xml version = "1.0" standalone = "no"?>
<!DOCTYPE CAR_RENTAL [
<!ELEMENT CAR_RENTAL (CUSTOMER, HIRE+, DATE)>
.
.
.
<!ENTITY data SYSTEM "data.xml">
]>

<DOCUMENT>
&data;
</DOCUMENT>
```

Στο αρχείο `data.xml` εμπεριέχονται τα πραγματικά δεδομένα για το έγγραφο:

```
<CAR_RENTAL>
      <CUSTOMER>
            Smith Sam
      </CUSTOMER>
      <CAR>
BMW Z3
</CAR>
</CAR_RENTAL>
```

Με αυτό τον τρόπο είναι δυνατή η δημιουργία εγγράφων από διαφορετικά τμήματα, με την επιλογή αυτών που χρειάζονται.

Προκαθορισμένες αναφορές για τις γενικές οντότητες

Υπάρχουν πέντε προκαθορισμένες αναφορές για οντότητες στην XML, που έχουν ορισθεί για χαρακτήρες που ερμηνεύονται ως χαρακτήρες σήμανσης ή άλλους χαρακτήρες ελέγχου. Αυτές οι αναφορές είναι:

- `&` αντιστοιχεί στο χαρακτήρα &
- `&apos;` αντιστοιχεί στο χαρακτήρα ‘
- `>` αντιστοιχεί στο χαρακτήρα >
- `<` αντιστοιχεί στο χαρακτήρα <
- `&quot;` αντιστοιχεί στο χαρακτήρα “

Αποδεικνύεται ότι μπορούν να δημιουργηθούν αναφορές σε οντότητες για ιδιαίτερους χαρακτήρες στην XML – το μόνο που χρειάζεται είναι να καθοριστεί ο σωστός κωδικός χαρακτήρας για την κωδικοποίηση που χρησιμοποιείται. Για παράδειγμα στην κωδικοποίηση UTF-8 ο κωδικός χαρακτήρας για το @ είναι #64 (όπου # καθορίζει ότι η τιμή είναι στο δεκαεξαδικό σύστημα), άρα μπορεί να ορισθεί για παράδειγμα μια οντότητα που ονομάζεται at_new, ώστε όταν γίνεται αναφορά σε αυτό να αντικαθίσταται με το @ όταν αναλύεται. Η οντότητα θα έχει την εξής μορφή:

```
<!ENTITY at_new “&#64;”>
```

Στην πραγματικότητα μπορούν να ορισθούν ακόμα και οι προκαθορισμένες αναφορές για τις οντότητες, στην περίπτωση που υπάρξει κάποιος επεξεργαστής XML που δεν τις καταλαβαίνει.

Εσωτερικές και εξωτερικές παραμετρικές Οντότητες

Όπως αναφέρθηκε παραπάνω οι αναφορές γενικών οντοτήτων χρησιμοποιούνται στα έγγραφα για την αντικατάστασή τους από τον επεξεργαστή της XML μέσα σε αυτά, με την αναφορά στην οποία αντιστοιχούν. Η χρήση τους περιορίζεται στην εισαγωγή κειμένου στο περιεχόμενο του εγγράφου και δε μπορούν να χρησιμοποιηθούν και να λειτουργήσουν με αποτέλεσμα μέσα σε DTD.

Στην πράξη για δηλώσεις στοιχείων (elements) και ιδιοτήτων (attributes) χρησιμοποιούνται οι παραμετρικές οντότητες. Οι αναφορές σε παραμετρικές οντότητες μπορούν να χρησιμοποιηθούν μόνο σε DTDs. Και σε αυτές όμως υπάρχουν επιπλέον περιορισμοί. Κάθε αναφορά σε παραμετρικές οντότητες που χρησιμοποιείται σε δηλώσεις ενός DTD πρέπει να εμφανίζεται κυρίως στο εξωτερικό τμήμα του DTD. Μπορούν να χρησιμοποιηθούν παραμετρικές οντότητες μέσα στο εσωτερικό τμήμα, αλλά με περιορισμένο τρόπο όπως θα αναφερθεί στη συνέχεια.

Αντίθετα από τις αναφορές στις γενικές οντότητες, οι αναφορές στις παραμετρικές οντότητες αρχίζουν με % και όχι με &. Η δημιουργία των παραμετρικών οντοτήτων μοιάζει με αυτή των γενικών, εκτός από το ότι περιέχουν ένα % στο παράγοντα του <!ENTITY> όπως στο παράδειγμα:

```
<!ENTITY % όνομα-οντότητας "τιμή-οντότητας ">
```

Μπορούν επίσης να δηλωθούν εξωτερικές παραμετρικές οντότητες, σε αντιστοιχία με τις γενικές οντότητες, με τις κωδικές λέξεις SYSTEM και PUBLIC όπως παρακάτω:

```
<!ENTITY % όνομα-οντότητας SYSTEM URI>
<!ENTITY % όνομα-οντότητας PUBLIC FPI URI>
```

Στο παράδειγμα που ακολουθεί δηλώνεται μια εσωτερική παραμετρική οντότητα με το όνομα DAMAGE που διατηρεί θέση για τη δήλωση <!ELEMENT DAMAGE EMPTY> μέσα στο DTD. Η αναφορά στην οντότητα γίνεται με % DAMAGE; για να συμπεριλάβει στο DTD το στοιχείο της δήλωσης <!ELEMENT DAMAGE EMPTY>:

```
<?xml version = “1.0” standalone = “yes”?>
<!DOCTYPE CAR_RENTAL [
```

```
<!ENTITY % DAM "<!ELEMENT DAMAGE EMPTY>">
<!ELEMENT RENTAL (CUSTOMER, HIRE+, DATE)>
.
.
.
% DAM;
]>
<CAR_RENTAL>
  <RENTAL>
      <CUSTOMER>
            Smith  Sam
      </CUSTOMER>
      <CAR>BMW Z3</CAR>
      <DAMAGE/>
  <RENTAL>
</CAR_RENTAL>
```

Παρατηρείται πως η δήλωση της παραπάνω οντότητας δεν ήταν και πολύ χρήσιμη, και ίσως θα ήταν πιο βολική η τοποθέτηση της δήλωσης του στοιχείου κατευθείαν μέσα στο DTD. Από την άλλη, δεν μπορεί να γίνει κάτι παραπάνω με τις εσωτερικές παραμετρικές οντότητες επειδή ορίζονται στο εσωτερικό τμήμα του DTD και δεν μπορούν να χρησιμοποιηθούν μέσα σε άλλες δηλώσεις. Η χρήση των εξωτερικών παραμετρικών οντοτήτων όμως είναι πιο αποτελεσματική όπως αναφέρεται στη συνέχεια.

Όταν χρησιμοποιείται μια παραμετρική οντότητα σε ένα εξωτερικό υποσύνολο ενός DTD, μπορεί να αναφερθεί αυτή η οντότητα οπουδήποτε μέσα στο DTD. Στο παράδειγμα που ακολουθεί χρησιμοποιείται ένα εξωτερικό DTD που ονομάζεται vehicle.dtd για το παρακάτω έγγραφο:

```
<?xml version = "1.0" standalone = "no"?>
<!DOCTYPE CAR_RENTAL SYSTEM "vehicle.dtd">
<CAR_RENTAL>
      <CUSTOMER>
            Smith  Sam
      </CUSTOMER>
      <CAR>BMW Z3</CAR>
      <DAMAGE/>
</CAR_RENTAL>
```

Στο εξωτερικό υποσύνολο του DTD, vehicle.dtd, πρόκειται να τοποθετηθούν χαρακτηριστικά, ώστε εκτός από ενοικιάσεις αυτοκινήτων να γίνονται και ενοικιάσεις μηχανών. Έτσι το στοιχείο <CAR_RENTAL> εκτώς από τον στοιχείο <CAR> θα περιέχει επιπλέον το στοιχείο <BIKE> που έχει τα ίδια χαρακτηριστικά με το στοιχείο <CAR>. Καθένα από αυτά διατηρεί το ίδιο «μοντέλο περιεχομένου» (content model), που περιέχει τα στοιχεία, <INSURANCE>, <FUEL> και <OTHERS>. Είναι πολύ χρήσιμο να εξοικονομηθεί χρόνος εκχωρώντας το μοντέλο περιεχομένου, (INSURANCE, FUEL, OTHERS), σε μια παραμετρική οντότητα που έχει ονομαστεί record. Η παραμετρική οντότητα record μπορεί να αναφερθεί οπουδήποτε, χρησιμοποιώντας την στις δηλώσεις των στοιχείων <CAR> και <BIKE>:

```
<!ENTITY % record "(INSURANCE, FUEL?, OTHERS?)">
<!ELEMENT CAR_RENTAL (CUSTOMER, (CAR | BIKE)+)>

<!ELEMENT CUSTOMER (#PCDATA)>
<!ELEMENT CAR %record;>
<!ELEMENT BIKE %record;>
.
.
.
```

Το έγγραφο λειτουργεί και αναλύει τα δεδομένα όπως ήταν αναμενόμενο. Και τα δύο στοιχεία <CAR> και <BIKE> που χρησιμοποιούνται μέσα στο στοιχείο <CAR_RENTAL> περιέχουν το ίδιο μοντέλο περιεχομένου:

```
<?xml version = "1.0" standalone = "no"?>
<!DOCTYPE CAR_RENTAL SYSTEM "hire.dtd">
<CAR_RENTAL>
.
.
.
      <CAR>
            <INSURANCE>Κλοπή</INSURANCE>
            <FUEL>Βενζίνη αμόλυβδη</FUEL>
            <OTHERS>Παράδοση</OTHERS>
      </CAR>
      <BIKE>
            <INSURANCE>
            Κλοπή, διάρρηξη, φυσικές καταστροφές
            </INSURANCE>
      </BIKE>
</CAR_RENTAL>
```

Το παράδειγμα τονίζει τη μεγάλη χρηστικότητα των παραμετρικών οντοτήτων, η οποία είναι να διατηρεί κείμενο το οποίο επαναλαμβάνεται συχνά στις δηλώσεις των στοιχείων (elements), σε ένα DTD. Στην παραπάνω περίπτωση προσδιορίστηκε το μοντέλο περιεχομένου των τριών στοιχείων χρησιμοποιώντας την ίδια παραμετρική οντότητα, αλλά θα ήταν επίσης εύκολο να τεθεί μια παραμετρική οντότητα, και εκεί να καθοριστεί μια λίστα ιδιοτήτων, η οποία να είναι η ίδια για όσα στοιχεία χρειάζεται. Με αυτόν τον τρόπο μπορούν να ελέγχονται οι δηλώσεις πολλών στοιχείων (elements) και ιδιοτήτων (attributes), ακόμα και σε μεγάλα DTD. Έτσι αν χρειαστεί να διαμορφωθεί μια δήλωση, αυτή η διαμόρφωση γίνεται μόνο στην παραμετρική οντότητα, και όχι σε κάθε δήλωση χωριστά.

Συμπεριλαμβάνοντας ένα DTD σε ένα άλλο DTD

Η συμπερίληψη ενός DTD μέσα σε ένα άλλο DTD διευκολύνει την παραμετροποίηση (modularization) του σχήματος (τα μεγάλα σχήματα μπορούν να χωριστούν σε μικρότερα). Χρησιμοποιούνται οι οντότητες για να περιληφθεί ένα έγγραφο μέσα σε ένα DTD.

Για να περιληφθεί ένα DTD (ή ένα τμήμα του) σε ένα άλλο, ο τρόπος είναι ίδιος ακριβώς όπως και στην SGML. Πρώτα δηλώνεται η οντότητα (entity) που πρέπει να περιληφθεί, και έπειτα γίνεται αναφορά σε αυτή με το όνομα της ως παράμετρος:

```
<!ENTITY % mylists SYSTEM "dtds/listfrag.ent">
...
%mylists;
```

Τέτοιες δηλώσεις γίνονται χαρακτηριστικά όλες μαζί στην κορυφή του κύριου αρχείου DTD, όπου μπορούν να ρυθμιστούν και να διατηρηθούν, αλλά αυτό δεν είναι απαραίτητο εφόσον δηλώνονται προτού να χρησιμοποιηθούν. Χρησιμοποιείται για αυτό η σύνταξη των παραμετρικών οντοτήτων (το σύμβολο %) επειδή το αρχείο πρόκειται να συμπεριληφθεί σε ένα DTD κατά το χρόνο της μεταγλώττισης, και όχι όταν το συγκεκριμένο έγγραφο αναλύεται από τον επεξεργαστή (parser) της XML.

Το κύριο πρόβλημα με την παραμετροποίηση (modularization) θεωρείται τα namespaces. Τα DTDs δεν υποστηρίζουν τα namespaces της XML 1.0. Επομένως όλα τα ονόματα των στοιχείων πρέπει να είναι μοναδικά.

Το παρακάτω παράδειγμα επιδεικνύει πώς μπορεί να συμπεριληφθεί ένα DTD σε ένα άλλο DTD. Το δεύτερο DTD περιλαμβάνει το πρώτο DTD και προσθέτει το στοιχείο "bar".

```
first.dtd:
<!ELEMENT foo ANY>
second.dtd:
<!ENTITY % first SYSTEM "first.dtd">
%first;
<!ELEMENT bar (#PCDATA | foo )*>
```

Τα έγγραφα μπορούν να αναφερθούν σε DTDs μέσω του πρωτοκόλλου HTTP. Δεδομένου ότι η υποστήριξη URI είναι ακόμα αδύνατη, πλήρη URLs χρησιμοποιούνται συνήθως για τη σύνδεση με τα DTDs. Διάφοροι κατατμητές (parsers) όπως είναι ο parser για τη Java της Oracle αφήνουν το χρήστη να προσδιορίσει το βασικό URL του DTD πριν αναλυθεί ένα έγγραφο έτσι ώστε να μπορούν να χρησιμοποιηθούν τα URIs. Αυτό είναι μάλλον φορτικό, όμως, ακόμα και το τυποποιημένο DOM API δεν το υποστηρίζει

Ένας τρόπος για να εξεταστεί αυτό είναι να δημιουργηθεί ένα ευρύ-εταιρικό όνομα εξυπηρετητή (server name) που ενεργεί ως αποθήκη DTD. Παραδείγματος χάριν: http://dtdrepository/dtd/dtdname.dtd. Εάν δεν είναι επιθυμητό να αφιερωθεί μια ολόκληρη μηχανή για την αποθήκευση των DTDs, μπορεί να δημιουργηθεί μόνο μια είσοδος DNS για το dtdrepository που δείχνει στο web-server.

παράδειγμα DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Doc (Joe)>
<!ELEMENT Joe (#PCDATA)>
<!ATTLIST Joe Foo CDATA #IMPLIED>
```

παράδειγμα αρχείου:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE Doc SYSTEM "http://dtdrepository/dtd/doc.dtd">
<Doc><Joe>hi</Joe></Doc>
```

3.3.5 Δηλώσεις Ιδιοτήτων

Ύστερα από τη δήλωση των στοιχείων (elements) είναι δυνατή η δήλωση ιδιοτήτων (attributes) για αυτές. Για κάθε στοιχείο, συνήθως δηλώνονται όλες οι ιδιότητές της σε ένα σημείο, χρησιμοποιώντας μια «λίστα δήλωσης ιδιοτήτων». Η δήλωση ενός attribute έχει την ακόλουθη σύνταξη:

```
<!ATTLIST όνομα-στοιχείου όνομα-ιδιότητας
 τύπος-ιδιότητας προκαθορισμένη-τιμή>
παράδειγμα:
DTD:
<!ATTLIST payment type CDATA "check">

XML:
<payment type="check"/>
```

Η δήλωση ξεκινάει με τη συμβολοσειρά <!ATTLIST, ακολουθούμενη από το όνομα του στοιχείου (όνομα-στοιχείου) στο οποίο ανήκουν οι ιδιότητες. Στην συνέχεια ακολουθεί ο ορισμός τις ιδιότητας που δηλώνεται και το σύμβολο τέλους >. Κάθε «δήλωση ιδιότητας» αποτελείται από το όνομά της (όνομα-ιδιότητας), από τον «τύπο δεδομένων» ή τις «απαριθμημένες τιμές» (τύπος-ιδιότητας) που δέχεται και την «περιγραφή της συμπεριφοράς της» (προκαθορισμένη-τιμή).

Παρακάτω αναφέρονται εν συντομία οι «τύποι δεδομένων» που μπορούν να χρησιμοποιηθούν στις ιδιότητες:

Τιμή	**Επεξήγηση**

CDATA	Η τιμή είναι character data (χαρακτήρες δεδομένων)
(en1|en2|..)	Η τιμή πρέπει να είναι μια από τις τιμές τις απαριθμημένης λίστας
ID	Η τιμή είναι ένα μοναδικό προσδιοριστικό
IDREF	Η τιμή είναι το ID ενός άλλου στοιχείου
IDREFS	Η τιμή είναι μια λίστα από άλλα IDs
NMTOKEN	Η τιμή είναι ένα έγκυρο XML όνομα
NMTOKENS	Η τιμή είναι μια λίστα από έγκυρα XML ονόματα
ENTITY	Η τιμή είναι μια οντότητα (entity)
ENTITIES	Η τιμή είναι μια λίστα από οντότητες
NOTATION	Η τιμή είναι το όνομα ενός notation (σημείωση)
xml:	Η τιμή είναι μια προκαθορισμένη xml τιμή

Παρακάτω αναφέρονται εν συντομία οι «τιμές της συμπεριφοράς» των ιδιοτήτων:

Τιμή	**Επεξήγηση**
value	Η προεπιλεγμένη τιμή μιας ιδιότητας
#REQUIRED	Η τιμή της ιδιότητας πρέπει να συμπεριληφθεί στο στοιχείο
#IMPLIED	Η ιδιότητα δεν είναι απαραίτητο να συμπεριληφθεί
#FIXED value	Η τιμή της ιδιότητας είναι προκαθορισμένη (fixed)

Στο παρακάτω παράδειγμα δηλώνεται μια λίστα ιδιοτήτων όπου οι ιδιότητες δηλώνονται για την οντότητα <memo>:

```
<!ATTLIST memo
      id                ID          #REQUIRED
      security     (high | low)     "high"
      keywords  NMTOKENS             #IMPLIED
```

Η πρώτη ιδιότητα, id είναι τύπου ID. Η κωδική λέξη #REQUIRED σημαίνει ότι αυτή η ιδιότητα πρέπει να οριστεί από τον συντάκτη του εγγράφου.

Η ιδιότητα security μπορεί να πάρει μια από τις δυο απαριθμημένες τιμές που ορίζονται, high ή low, με προεπιλεγμένη την τιμή high.

Η τρίτη ιδιότητα, keywords, δέχεται μια τιμή του τύπου NMTOKENS που θα περιγραφεί αργότερα, ενώ η κωδική λέξη #IMPLIED σημαίνει ότι η ιδιότητα είναι προαιρετική και δεν έχει προκαθορισμένη τιμή.

Στη συνέχεια παρουσιάζονται λεπτομερώς οι παραπάνω τύποι δεδομένων καθώς και οι τύποι συμπεριφοράς που μπορούν να πάρουν. Στα παραδείγματα που ακολουθούν χρησιμοποιείται το παρακάτω αρχείο DTD, rental.dtd:

```
<!ELEMENT CAR_RENTAL (CUSTOMER, RENTAL+)>

<!ELEMENT CUSTOMER (#PCDATA)>
<!ELEMENT RENTAL (CAR+, INSURANCE?, FUEL?, OTHERS*>

<!ELEMENT CAR (#PCDATA)>
<!ELEMENT INSURANCE (#PCDATA)>
<!ELEMENT FUEL (#PCDATA)>
<!ELEMENT OTHERS (#PCDATA)>
```

Οριζόμενοι «τύποι συμπεριφοράς» για τις ιδιότητες

Άμεσα Προσδιορισμένες Τιμές

Σε περιπτώσεις που δεν παρέχεται κάποια τιμή σε μια ιδιότητα από το χρήστη, ο επεξεργαστής της XML εφαρμόζει την τιμή που έχει προσδιοριστεί για αυτήν την ιδιότητα στο DTD. Η δήλωση της «προσδιορισμένης τιμής» γίνεται με το κλείσιμο αυτής σε διπλά εισαγωγικά, στο τμήμα που ορίζεται η συμπεριφορά της ιδιότητας. Οι προσδιορισμένες τιμές είναι χρήσιμες συνήθως όταν μια τιμή είναι συνηθισμένη για κάποια ιδιότητα:

Παράδειγμα λίστας ιδιοτήτων:

```
<!ATTLIST CUSTOMER
      TYPE CDATA "0"
      PAY CDATA "0">
Παράδειγμα XML αρχείου:
<CAR_RENTAL>
      <CUSTOMER TYPE = "excellent"  PAY = "cash"> … </CUSTOMER>
…
      <CUSTOMER TYPE = "good"  PAY = "credit"> … </CUSTOMER>
</CAR_RENTAL>
```

Υποχρεωτική ύπαρξη της ιδιότητας (#REQUIRED)

Όταν στη συμπεριφορά της ιδιότητας αντιστοιχεί η κωδική λέξη #REQUIRED, τότε δεν έχει προσδιοριστεί συγκεκριμένη τιμή για την ιδιότητα, αλλά απαιτείται από τον χρήστη που χρησιμοποιεί το DTD να ορίσει μία, μέσα στο έγγραφο. Σε περίπτωση που παραλειφθεί ή δοθεί μια κενή τιμή στην ιδιότητα θα εμφανισθεί σφάλμα ανάλυσης. Ακολουθεί ένα παράδειγμα δημιουργίας λίστας ιδιοτήτων:

```
<!ATTLIST CUSTOMER
      PAY CDATA #REQUIIRED>
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
  <CUSTOMER PAY ="cash"> … </CUSTOMER> …
  <CUSTOMER PAY ="credit"> …</CUSTOMER>
</CAR_RENTAL>
```

Προαιρετική ύπαρξη της ιδιότητας (#IMPLIED)

Η χρησιμοποίηση του οριζόμενου τύπου συμπεριφοράς #IMPLIED για τον τύπο δεδομένων μιας ιδιότητας τη δηλώνει ως προαιρετική. Αφήνεται στη κρίση του χρήστη της XML αν πρέπει ή δεν πρέπει να συμπεριληφθεί η ιδιότητα στο έγγραφο, ενώ ο επεξεργαστής της XML δε θα διαμαρτυρηθεί αν η ιδιότητα δε χρησιμοποιηθεί, διότι δεν απαιτείται από το DTD. Ακολουθεί ένα παράδειγμα δημιουργίας λίστας ιδιοτήτων:

```
<!ATTLIST CUSTOMER
      TYPE CDATA #IMPLIED>
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
  <CUSTOMER TYPE ="excellent"> … </CUSTOMER> …
  <CUSTOMER> … </CUSTOMER>
</CAR_RENTAL>
```

Προκαθορισμένη τιμή της ιδιότητας (#FIXED)

Είναι δυνατό να τεθεί μια τιμή σε μια ιδιότητα ώστε να τη διατηρεί πάντα. Η τιμή αυτή τίθεται με την κωδική λέξη #FIXED, που ορίζει ότι η τιμή για μια ιδιότητα είναι σταθερή, και ύστερα ακολουθεί η τιμή της ιδιότητας. Οι τιμές των ιδιοτήτων, που ορίζονται με αυτήν τη συμπεριφορά στο DTD, είναι λιγότερο ευέλικτες, καθώς δεν μπορούν να αλλάξουν από το χρήστη του εγγράφου.

Στο παρακάτω παράδειγμα ορίζεται η ιδιότητα LANGUAGE για το στοιχείο <COSTUMER> στα Αγγλικά (EN) και καθορίζει ότι μόνο αυτή η τιμή είναι έγκυρη για την ιδιότητα:

Παράδειγμα λίστας ιδιοτήτων:

```
<!ATTLIST CUSTOMER
      LANGUAGE CDATA #FIXED "EN">
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
  <CUSTOMER> … </CUSTOMER>
<CAR_RENTAL>
```

Παρατηρείται ότι αν και δεν χρησιμοποιείται η ιδιότητα LANGUAGE στην οντότητα <CUSTOMER>, ο επεξεργαστής της XML εμφανίζει την ιδιότητα και την τιμή της στην υποκείμενη εφαρμογή επειδή έχει δηλωθεί ως #FIXED. Εφόσον όμως θελήσει ο χρήστης να χρησιμοποιήσει την ιδιότητα, πρέπει να θέσει την τιμή που έχει προκαθοριστεί στο DTD, αλλιώς ο επεξεργαστής της XML θα παρουσιάσει μήνυμα λάθους. Στη συνέχεια παρουσιάζονται οι δυνατοί τύποι που μπορούν να χρησιμοποιηθούν στις ιδιότητες.

Τύποι Ιδιοτήτων

Έως τώρα έχει χρησιμοποιηθεί μόνο ο τύπος ιδιότητας CDATA στις δηλώσεις ιδιοτήτων, που είναι και ο πιο συνηθισμένος τύπος, διότι επιτρέπει στο χρήστη να χρησιμοποιήσει απλό κείμενο για τις τιμές των ιδιοτήτων. Είναι όμως δυνατό να προσδιοριστεί ένας αριθμός από διαφορετικούς τύπους ιδιοτήτων, οι οποίοι παρουσιάζονται στη συνέχεια. Αυτοί οι τύποι, όμως, δεν είναι αρκετά λεπτομερείς ώστε να δηλωθούν συγκεκριμένοι τύποι δεδομένων όπως ρητοί ακέραιοι (float) ή ακέραιοι αριθμοί (int), αλλά εφοδιάζουν το χρήστη με τη δυνατότητα ελέγχου της σύνταξης ενός εγγράφου.

Δεδομένα χαρακτήρων (CDATA)

Ο πιο απλός τύπος ιδιοτήτων που υπάρχει είναι ο τύπος CDATA, που είναι δεδομένα απλών χαρακτήρων. Στην ιδιότητα μπορεί να τεθεί ως τιμή μια οποιαδήποτε συμβολοσειρά που δεν περιέχει χαρακτήρες σήμανσης. Απαγορεύεται ρητά να χρησιμοποιηθεί ως τιμή μιας ιδιότητας, οποιαδήποτε συμβολοσειρά που περιέχει τους χαρακτήρες <, >, ', ", ή &. Εάν χρειάζεται να χρησιμοποιηθεί κάποιος από αυτούς τους χαρακτήρες, χρησιμοποιούνται οι προκαθορισμένες αναφορές οντοτήτων που έχουν οριστεί για αυτούς. Αυτές οι αναφορές όταν αναλυθούν από τον επεξεργαστή, θα αντικατασταθούν στην υποκείμενη εφαρμογή από τους αντίστοιχους χαρακτήρες. Παρακάτω φαίνεται η χρήση του τύπου σε μια λίστα ιδιοτήτων, που χρησιμοποιήθηκε και σε προηγούμενο παράδειγμα:

```
<!ATTLIST CUSTOMER
      TYPE CDATA "0"
      PAY CDATA "0"
      DEFAULTS CDATA "0">
```

Απαρίθμηση τιμών (Enumerated)

Ο απαριθμημένος τύπος δε χρησιμοποιεί κάποια κωδική λέξη όπως οι άλλοι τύποι ιδιοτήτων, αλλά αντί για αυτό παρέχει μια «λίστα» από πιθανές τιμές που μπορεί να πάρει η ιδιότητα. Κάθε πιθανή τιμή πρέπει να έχει μια έγκυρη ονομασία για την XML, δηλαδή ο πρώτος χαρακτήρας να αρχίζει από γράμμα ή το σύμβολο της υπογράμμισης. Στο παρακάτω παράδειγμα δηλώνεται η ιδιότητα KIND, που μπορεί να πάρει μόνο μία από τις πιθανές τιμές που αναφέρονται στην λίστα, με προεπιλεγμένη τιμή την τιμή "ΑΠΛΗ":

Παράδειγμα λίστας ιδιοτήτων:

```
<!ATTLIST INSURANCE
      KIND (ΑΠΛΗ | ΜΕΙΚΤΗ) "ΑΠΛΗ">
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
            …
            <INSURANCE KIND = "ΑΠΛΗ">Κλοπή</INSURANCE>
            …
            <INSURANCE>Κλοπή</INSURANCE>
            …
            <INSURANCE KIND = "ΜΕΙΚΤΗ">
                  Κλοπή, φυσικές καταστροφές, διάρρηξη, εμπρησμός
            </INSURANCE>
</CAR_RENTAL>
```

Οι «απαριθμημένες τιμές» χρησιμοποιούνται όταν χρειάζεται να οριστούν οι τιμές που μπορεί να πάρει μια ιδιότητα, και αυτή να περιοριστεί μόνο σε αυτές.

Ονόμα συμβόλων (NMTOKEN, name token)

Ένας άλλος τύπος ιδιότητας που χρησιμοποιείται συχνά είναι ο τύπος NMTOKEN. Ένα "name token" είναι μια συμβολοσειρά χαρακτήρων που αρχίζει με γράμμα ή το σύμβολο της υπογράμμισης, και μπορεί να περιέχει αριθμούς, γράμματα και σημεία στίξης. Οι τιμές που δέχεται ένα NMTOKEN δεν πρέπει να συμπεριλαμβάνουν κενούς χαρακτήρες:

```
<!ATTLIST CAR
      CHARACTERISTICS NMTOKEN #REQUIRED>
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
      …
  <CAR CHARACTERISTICS = "mod.97' 16v">BMW Z3 </CAR>
      …
</CAR_RENTAL>
```

Λίστα ονομάτων συμβόλων (NMTOKENS, name token list)

Μια λίστα με "name token" ορίζει ότι η τιμή μιας ιδιότητας μπορεί να αποτελείται από μια ακολουθία με ένα η περισσότερα NMTOKENS, χωρισμένα μεταξύ τους με κενό. Παρακάτω ορίζεται η ιδιότητα CONTACT_NAME με τύπο NMTOKENS, ώστε οι τιμές τις ιδιότητας να διατηρούν το όνομα και το επίθετο των πελατών:

Παράδειγμα λίστας ονομάτων συμβόλων:

```
<!ATTLIST CUSTOMER
CONTACT_NAME NMTOKENS #IMPLIED>
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
  <CUSTOMER CONTACT_NAME = "George Starr">
            …
  </CUSTOMER>
```

```
</CAR_RENTAL>
```

Προσδιοριστικό ID (unique identifier)

Πρόκειται για έναν ειδικό τύπο ιδιότητας, που παρέχει στα στοιχεία (elements) ενός εγγράφου μια μοναδική τιμή. Δηλώνεται με την κωδική λέξη ID και δεν επιτρέπεται δυο διαφορετικά στοιχεία να έχουν την ίδια τιμή στον τύπο ID μιας ιδιότητας. Ο τύπος ιδιότητας ID έχει την ίδια συμπεριφορά και σύνταξη με την ιδιότητα NMTOKEN, και μπορεί να δοθεί για κάθε στοιχείο μόνο μια ιδιότητα αυτού του τύπου διότι χρησιμοποιείται από τις εφαρμογές για να προσδιορίσουν μοναδικά τα στοιχεία. Η παρακάτω ιδιότητα CUSTOMER_ID προσδιορίζει μοναδικά το στοιχείο <CUSTOMER>

```
<!ATTLIST CAR
LABEL ID #REQUIRED>
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
            ……
            <CAR LABEL = "HPB 4797">BMW Z3</CAR>
            ……
<CAR_RENTAL>
```

Σε αυτό το σημείο σημειώνεται ότι δεν μπορεί να χρησιμοποιηθεί ο τύπος ιδιότητας ID με την κωδική λέξη `#FIXED` για να περιγράψει την ιδιότητα, επειδή όλες οι ιδιότητες με αυτήν τη συμπεριφορά έχουν την ίδια τιμή. Συνήθως χρησιμοποιείται η κωδική λέξη `#REQUIRED`.

Οι τιμές του τύπου ιδιότητας ID δεν μπορούν να είναι απλοί αριθμοί, διότι δεν αποτελούν έγκυρα ονόματα για την XML που ορίζει ότι κάθε όνομα πρέπει να αρχίζει με γράμμα ή με το σύμβολο της υπογράμμισης.

Προσδιοριστικό IDREF (identifier reference)

Αυτός ο τύπος ιδιότητας είναι όμοιος με τον τύπο ιδιότητας ID, αλλά αντί να προσδιορίζει μια μοναδική τιμή για ένα στοιχείο (element), αναφέρεται στην τιμή ID ενός άλλου στοιχείου. Εάν δεν υπάρχει στοιχείο με την προσδιοριζόμενη τιμή ο επεξεργαστής αποδίδει μήνυμα λάθους. Χρησιμοποιείται για να δηλώσει κάποια πιθανή σχέση μεταξύ των στοιχείων, όπως μια σχέση πατέρα-παιδιού που δεν μπορεί να απεικονιστεί με συνδέσμους στη δομή ενός εγγράφου. Παρακάτω δηλώνονται οι δυο ιδιότητες, `CUSTOMER_ID` τύπου `ID` και `CAR_ID` τύπου `IDREF` που διατηρεί την τιμή `ID` ενός πελάτη.

```
<!ATTLIST CAR
LABEL ID #REQUIRED>
<!ATTLIST CUSTOMER
CUSTOMER_ID ID #REQUIRED>
CAR_ID IDREF #IMPLIED>
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
<CUSTOMER CUSTOMER_ID = "A123B" CAR_ID = "HPB 4797">
…
</CUSTOMER>
…
<CUSTOMER CUSTOMER_ID = "A123B"> … </CUSTOMER>
</CAR_RENTAL>
```

Λιστα προσδιοριστικών (IDREFS, identifier reference list)

Πρόκειται για μια λίστα με IDREF τιμές που έχει την ίδια μορφή με τα NMTOKENS. Κάθε ιδιότητα IDREF στη λίστα πρέπει να ταιριάζει σε μια τιμή ιδιότητας ID στο έγγραφο.

Οντότητα (ENTITY, entity name)

Αυτός ο τύπος ιδιότητας δέχεται ως τιμή το όνομα μιας «γενικής οντότητας». Χρησιμοποιείται αφού δηλωθεί η οντότητα στο DTD. Στο παρακάτω παράδειγμα δηλώνεται η οντότητα PHOTO, που αναφέρεται σε ένα εξωτερικό αρχείο εικόνας. Στη συνέχεια δημιουργείται η ιδιότητα IMAGE, στην οποία τίθεται ως τιμή η οντότητα που δηλώθηκε.

```
<!ENTITY PHOTO SYSTEM "image.gif">
<!ATTLIST CAR
      IMAGE ENTITY #IMPLIED>
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
      <CAR IMAGE = "PHOTO">BMW Z3</CAR>
                  …
</CAR_RENTAL>
```

Λίστα οντοτήτων (ENTITIES, entity name list)

Σε αντιστοιχία με τα NMTOKEN μπορεί να οριστεί μια λίστα με οντότητες στη λίστα ιδιοτήτων ενός στοιχείου. Πρόκειται για τον τύπο ιδιοτήτων ENTITIES που ως τιμές δέχεται μια «λίστα οντοτήτων» χωρισμένες μεταξύ τους με κενό όπως φαίνεται και στο παρακάτω παράδειγμα:

```
<!ENTITY PHOTO1 SYSTEM "image.gif">
<!ENTITY PHOTO2 SYSTEM "image2.gif">
<!ATTLIST CAR
      IMAGE ENTITIES #IMPLIED>
```

Παράδειγμα XML αρχείου:

```
<CAR_RENTAL>
      <CAR IMAGE = "PHOTO1 PHOTO2">BMW Z3</CAR>
                  …
</CAR_RENTAL>
```

Ο τύπος τις λίστας οντοτήτων χρησιμοποιείται σε περιπτώσεις που χρειάζεται να τεθεί ένας αριθμός από οντότητες στην ίδια ιδιότητα. Επειδή οι οντότητες μπορεί μερικές φορές να είναι αρκετά πολύπλοκες και να περιέχουν ακόμα και άλλες οντότητες, αυτός είναι ένας τρόπος για να αποθηκεύονται λεπτομερή δεδομένα σε ένα έγγραφο απλά με τη χρήση των ιδιοτήτων.

Τύπος Σημείωσης (NOTATION)

Η XML έχει σχεδιαστεί κυρίως για να περιέχει πληροφορίες κειμένου, και δεν είναι ιδανικό να αποθηκεύονται σε αυτή δυαδικά δεδομένα όπως εικόνες bitmap ή συμπιεσμένο κείμενο. Θα περιοριζόταν πολύ όμως η XML αν μπορούσε να χειριστεί μόνο απλό κείμενο. Για αυτόν το λόγο παρέχεται ένας ειδικός τύπος ιδιότητας που ονομάζεται NOTATION και μπορεί να δεχτεί τιμές που έχουν δηλωθεί ως "notations" (σημειώσεις). Στη συνέχεια θα παρουσιαστεί λεπτομερέστερα αυτός ο τύπος ιδιότητας, αφού πρώτα αναφερθεί η χρήση και η δήλωση των notations

3.3.6 Η χρήση των Notations (Σημειώσεων) και των Unparsed Data (μη αναλυόμενων δεδομένων)

Χρήση των NOTATIONS

Τα "notations" προσδιορίζουν τον τύπο των δεδομένων που δεν είναι XML. Χρησιμοποιούνται για να περιγράψουν «εξωτερικές οντότητες» και να πληροφορήσουν τον επεξεργαστή της XML για το είδος των δεδομένων που πρόκειται να ενσωματώσει στο έγγραφο. Η δήλωση των notations συντάσσεται με τον εξής τρόπο:

```
<!NOTATION όνομα προσδιοριστής>
```

όπου στην αρχή καθορίζεται η δήλωση ενός notation με τον παράγοντα <!NOTATION>, και στη συνέχεια ακολουθεί το όνομά του (name) και ένας εξωτερικός προσδιοριστής που έχει κάποιο νόημα για τον επεξεργαστή της XML. Το νόημα του προσδιοριστή εξαρτάται κάθε φορά από την υποκείμενη εφαρμογή. Ο προσδιοριστής αποτελείται από την κωδική λέξη SYSTEM ή από την κωδική λέξη PUBLIC, όταν πρόκειται να χρησιμοποιηθεί κάποιος συμβατικός δημόσιος προσδιοριστής (Formal Public Identifier – FPI), και από τον εξωτερικό προσδιοριστή που πρόκειται να χρησιμοποιηθεί.

Ένας δημοφιλής τύπος προσδιοριστή για τα notations είναι οι «επεκτάσεις διαδικτυακής αλληλογραφίας πολλαπλού σκοπού» (Multipurpose Internet Mail Extensions – MIME), όπως οι τύποι image/gif, application/xml και text/html. Άλλοι πιθανοί προσδιοριστές μπορεί να είναι ένα URL ή το όνομα μιας εφαρμογής λογισμικού σε ένα τοπικό σύστημα.

Δεν έχει προταθεί ακόμα κάποιο σύμφωνο σχήμα για τους προσδιοριστές των notations, και έτσι οτιδήποτε από όσα αναφέρθηκαν παραπάνω μπορεί να χρησιμοποιηθεί. Το σημαντικότερο είναι ότι οι προσδιοριστές είναι μοναδικοί, και παρέχουν αρκετή πληροφορία στον επεξεργαστή της XML για να προσπελάσει τα δεδομένα.

Η δήλωση ενός notation δημιουργεί έναν «χαρακτηρισμό» που μπορεί να συνδυαστεί με τη δήλωση μιας ιδιότητας, ή μιας εξωτερικής, «μη αναλυόμενης από τον επεξεργαστή, οντότητας» (unparsed entity). Στο παρακάτω παράδειγμα δηλώνονται δυο notations, GIF και JPG, για τους τύπους MIME, image/gif και image/jpeg, και στη συνέχεια δηλώνεται η ιδιότητα IMAGE_TYPE, τύπου notation, στην οποία μπορούν να εκχωρηθούν ως τιμές κάποιο από τα notations που δηλώνονται προηγουμένως.

```
<?xml version = "1.0" standalone = "no"?>
<!DOCTYPE CAR_RENTAL SYSTEM "rental.dtd"[
<!NOTATION GIF SYSTEM "image/gif ">
<!NOTATION JPG SYSTEM "image/ jpeg">
<!ATTLIST CAR
      PHOTO NMTOKEN #IMPLIED
      IMAGE_TYPE NOTATION (GIF | JPG) #IMPLIED>
]>
<CAR_RENTAL>
      <CAR PHOTO = "image.gif" IMAGE_TYPE = "GIF" >
     BMW Z3
   </CAR>
              ……
</CAR_RENTAL>
```

Στο παραπάνω παράδειγμα απλά τέθηκε η τιμή μιας ιδιότητας, IMAGE, στο όνομα ενός αρχείου εικόνας, το image.gif. Είναι δυνατό όμως, με τη χρήση των notations, μία «μη αναλυόμενη οντότητα» (unparsed entity), όπως μια εικόνα, να ενσωματωθεί σε ένα έγγραφο XML, και να γίνει πραγματικό τμήμα αυτού.

Μη αναλυόμενες οντότητες (Unparsed entities)

Όπως αναφέρθηκε και προηγουμένως, τα αρχεία των εγγράφων XML αποτελούνται κυρίως από κείμενο, και δεν είναι πρακτική η αποθήκευση δεδομένων σε αυτά που δε θεωρούνται έγκυρα από την XML, όπως είναι τα δυαδικά αρχεία και το συμπιεσμένο κείμενο. Για αυτόν το λόγο ορίστηκαν οι «μη αναλυόμενες οντότητες», ώστε να είναι δυνατό να συσχετιστούν αυτά τα δεδομένα με τα έγγραφα της XML. Η δήλωση των μη αναλυόμενων οντοτήτων είναι όμοια με αυτή των «εξωτερικών γενικών οντοτήτων», εκτός από τη διαφορά ότι υπάρχει επιπλέον η κωδική λέξη NDATA (που δηλώνει ότι αναφέρεται σε "μη αναλυόμενη οντότητα"), και ένα δηλωμένο notation.

Η ενσωμάτωση των δεδομένων σε ένα έγγραφο που δεν είναι έγκυρα για την XML γίνεται με τον εξής τρόπο: αρχικά δηλώνεται ένα "notation" για τον τύπο δεδομένων που δε θεωρείται έγκυρο για την XML. Στη συνέχεια δημιουργείται μια «μη αναλυόμενη

οντότητα», που αναφέρεται σε ένα αρχείο που περιέχει τα μη έγκυρα δεδομένα. Ύστερα δημιουργείται μια ιδιότητα τύπου ENTITY για την εκχώρηση της οντότητας σε αυτή. Στο παράδειγμα που ακολουθεί παρουσιάζεται η σειρά των παραπάνω βημάτων.

```
<?xml version = "1.0" standalone = "no"?>
<!DOCTYPE CAR_RENTAL SYSTEM "rental.dtd">
<!NOTATION GIF SYSTEM "image/gif ">
<!ENTITY PHOTO SYSTEM "image.gif" NDATA GIF>
<!ATTLIST CAR
      IMAGE ENTITY #IMPLIED>
<CAR_RENTAL>
            <CAR IMAGE = "PHOTO">BMW Z3</CAR>
            ……
            </CUSTOMER>
</CAR_RENTAL>
```

Η χρησιμοποίηση των εξωτερικών μη αναλυόμενων οντοτήτων (external unparsed entities) με αυτόν τον τρόπο, έχει ως αποτέλεσμα, οι έγκυροι επεξεργαστές της XML να μην προσπαθούν να τις διαβάσουν και να τις αναλύσουν, αλλά συχνά ελέγχουν αν υπάρχουν ώστε το έγγραφο να είναι πλήρες.

3.3.7 Η χρήση του INCLUDE και του IGNORE

Δύο σημαντικές οδηγίες του DTD που συχνά χρησιμοποιούνται στις παραμετρικές οντότητες είναι το INCLUDE και το IGNORE. Αυτές οι εντολές χρησιμοποιούνται για να ενσωματώσουν ή να αποβάλλουν αντίστοιχα τμήματα του DTD. Συντάσσονται σύμφωνα με το παρακάτω υπόδειγμα:

```
<![ INCLUDE [DTD Section]]>   και
<![ IGNORE [DTD Section]]>
```

Χρησιμοποιώντας αυτές τις δύο εντολές είναι δυνατόν να προσαρμοστεί κατάλληλα το επιθυμητό DTD. Για παράδειγμα τα στοιχεία: SALES, COUNTING_ROOM, MANAGEMENT μπορούν να προστεθούν ή να αφαιρεθούν από το DTD με τον παρακάτω τρόπο:

```
<![INCLUDE[
<!ELEMENT SALES (#PCDATA)>
<!ELEMENT COUNTING_ROOM (#PCDATA)>
<!ELEMENT MANAGEMENT (#PCDATA)>
]]>
```

ΚΑΙ

```
<![IGNORE[
<!ELEMENT SALES (#PCDATA)>
<!ELEMENT COUNTING_ROOM(#PCDATA)>
<!ELEMENT MANAGEMENT (#PCDATA)>
]]>
```

Δε φαίνεται με την πρώτη ματιά η χρησιμότητα του INCLUDE του IGNORE. Η σπουδαιότητα της χρήσης τους γίνεται πιο αναγνωρίσιμη και κατανοητή, όταν χρησιμοποιούνται και τα δυο, μαζί με οντότητες, για να προσαρμοστεί το DTD σύμφωνα με τις απαιτήσεις του χρήστη. Τότε δηλαδή μπορεί, ανάλογα με την τιμή που θα δίνεται στην παραμετρική οντότητα, να θεωρηθούν έγκυρα ή άκυρα τα μέρη του DTD αντικαθιστώντας το IGNORE και το INCLUDE και αντίστροφα.

Παράδειγμα:

Πρώτα απ' όλα για να μπορέσουν να χρησιμοποιηθούν οι παραμετρικές οντότητες σε μέρη του DTD όπου υπάρχουν το INCLUDE και το IGNORE, η εργασία πρέπει να γίνει σύμφωνα με το εξωτερικό DTD που θα καλείται company.dtd:

```
<?xml version = "1.0"standalone="no"?>
<!DOCTYPE CAR_RENTAL SYSTEM "rental.dtd">
<CAR_RENTAL>
  <COSTUMER>John Pappas</COSTUMER>
  <CAR>BMW Z3</CAR>
  <INSURANCE/>
</CAR_RENTAL>
```

Η χρησιμοποίηση του INCLUDE σε μια παραμετρική οντότητα πραγματοποιείται όπως στο παρακάτω παράδειγμα:

```
<!ENTITY % includer  "INCLUDE">
<!ELEMENT CAR_RENTAL (COSTUMER,CAR,INSURANCE)*>
<!ELEMENT COSTUMER (#PCDATA)>
<!ELEMENT CAR (#PCDATA)>
<!ELEMENT INSURANCE (#PCDATA)>

<![%includer;[
<!ELEMENT SALES (#PCDATA)>
<!ELEMENT COUNTING_ROOM(#PCDATA)>
<!ELEMENT MANAGEMENT (#PCDATA)>
]]>
```

Ανάλογα γίνεται και η εργασία με τη χρήση του IGNORE. Έτσι μπορεί να συμπεριληφθεί ή να αγνοηθεί το προκείμενο τμήμα του DTD αλλάζοντας μόνο την τιμή της οντότητας includer. Χρησιμοποιώντας μία τέτοια μέθοδο μπορεί πιο εύκολα να ομαδοποιηθούν οι οντότητες που χρειάζεται να χρησιμοποιηθούν για να διαμορφωθεί αυτόματα το DTD.

3.4 Μοντελοποίηση XML εγγράφων με χρήση των XML Schemas

3.4.1 Εισαγωγή στην τεχνολογία XML Schema

Τα DTD παρέχουν μια βασική γραμματική για τον καθορισμό ενός εγγράφου XML από την άποψη των μεταδεδομένων που περιλαμβάνουν τη μορφή του εγγράφου. Ένα XML Schema το παρέχει αυτό, και επιπλέον παρέχει έναν λεπτομερή τρόπο για να καθοριστεί τι μπορούν ή δεν μπορούν να περιέχουν τα δεδομένα. Παρέχει πολύ περισσότερο έλεγχο για τον σχεδιαστή του τύπου εγγράφου, και παρέχει μια αντικειμενοστρεφή προσέγγιση, με όλα τα οφέλη που αυτό συνεπάγεται.

Το XML Schema αποτελεί από το Μαϊο του 2001 μια επίσημη σύσταση της W3C και οι λειτουργίες που παρέχει μπορούν να περιγραφτούν ως εξής:

- καθορίζει τα στοιχεία που μπορούν να εμφανιστούν σε ένα έγγραφο
- καθορίζει τις ιδιότητες που μπορούν να εμφανιστούν σε ένα έγγραφο
- καθορίζει ποια στοιχεία είναι στοιχεία-παιδιά
- καθορίζει τη διάταξη των στοιχείων-παιδιών
- καθορίζει τον αριθμό των στοιχείων-παιδιών
- καθορίζει εάν ένα στοιχείο είναι κενό ή αν μπορεί να περιέχει κείμενο ή να είναι
- καθορίζει τους τύπους δεδομένων (data types) για τα στοιχεία και τις ιδιότητες

- καθορίζει τις προεπιλογές (default) και τις σταθερές τιμές (fixed values) για τα στοιχεία και τις ιδιότητες

Τα Schemas γράφονται ως αρχεία XML, αποφεύγοντας την ανάγκη για το λογισμικό να μπορεί να επεξεργαστεί τη δηλωτική σύνταξη της XML μέσω του DTD. Για την ακρίβεια, αυτό θα καθιστούσε ευκολότερη τη σύνταξη και το χειρισμό του σχήματος με τα συνηθισμένα εργαλεία χειρισμού εγγράφων. Αφετέρου, θεωρήθηκε ότι η παραδοσιακή σύνταξη του DTD δεν επέτρεπε στους σχεδιαστές μοντέλων εγγράφων τη δυνατότητα να επιβάλουν αρκετούς περιορισμούς στα δεδομένα (παραδείγματος χάριν, η δυνατότητα να δηλωθεί ότι ένας ορισμένος τύπος στοιχείου πρέπει πάντα να έχει μια θετική τιμή ακέραιων αριθμών, ότι δεν μπορεί να είναι κενός, ή ότι οι τιμές που δέχεται να προέρχονται από ένα σύνολο τιμών). Αυτό διευκολύνει την ανάπτυξη του λογισμικού χρησιμοποιώντας αυτά τα δεδομένα επειδή ο προγραμματιστής πρέπει να γράψει λιγότερο κώδικα για των έλεγχο λαθών. Οι σημαντικότεροι λόγοι για την αποφυγή της χρήσης DTD είναι οι εξής:

- Το DTD δε χρησιμοποιεί τη σύνταξη της
- Αναμειγνύονται με την προδιαγραφή της XML 1.0 (θα υπήρχε πολύ λιγότερη σύγχυση εάν οριζόταν ξεχωριστά. Ακόμη και οι επεξεργαστές μη-επικύρωσης πρέπει να εξετάσουν το DTD)
- Δεν υπάρχει κανένας περιορισμός στα δεδομένα χαρακτήρων (character data) (εάν ένα character data επιτρέπεται, τότε *οποιαδήποτε* character data επιτρέπεται)
- Έχουν πολύ απλά μοντέλα τιμών για τις ιδιότητες (οι απαριθμημένες τιμές είναι σαφώς ανεπαρκείς)
- Δεν μπορούν να αναμίξουν τα δεδομένα χαρακτήρων με άλλο μοντέλο περιεχομένου (και τα μοντέλα περιεχομένων είναι γενικά δύσκολο να χρησιμοποιηθούν για τις σύνθετες απαιτήσεις)
- Δεν παρέχουν καμία υποστήριξη για ονοματοδοσίες (Namespaces) (φυσικά, η XML 1.0 καθορίστηκε πριν από τα Namespaces)
- Πολύ περιορισμένη υποστήριξη για συνδυασμό υπομονάδων κώδικα και για την επαναχρησιμοποίηση τους (ο μηχανισμός των οντοτήτων (entity) είναι πάρα πολύ χαμηλού επιπέδου)
- Καμία υποστήριξη για την εξέλιξη του σχήματος, την επέκταση, ή την κληρονομικότητα των δηλώσεων (δύσκολο να γραφτούν, να διατηρηθούν, και να διαβάσουν μεγάλα DTDs, και να καθοριστούν οικογένειες σχετικών σχημάτων)
- Πάρα πολύ απλός μηχανισμός της ιδιότητας ID (δε σημειώνει τις απαιτήσεις, τη μοναδικότητα, την εμβέλεια κ.λπ.)
- Υπάρχουν προεπιλογές μόνο για τις ιδιότητες, όχι για τα στοιχεία (αλλά αυτό θα ήταν συχνά βολικό)
- Οι προεπιλογές δεν μπορούν να προσδιοριστούν ξεχωριστά από τις δηλώσεις (θα ήταν βολικό να υπάρχουν οι προεπιλογές σε ξεχωριστές υπομονάδες)

3.4.2 Πλεονεκτήματα των XML Schema

Υπάρχουν διάφοροι λόγοι για τους οποίους το XML Schema υπερτερεί σχεδόν εξολοκλήρου από το DTD. Παρακάτω αναφέρονται οι κυριότεροι από αυτούς τους λόγους.

Το XML Schema υποστηρίζει τύπους δεδομένων

Τα XML Schemas παρέχουν περισσότερες δυνατότητες από τα DTDs εκ των οποίων μια από τις πιο σημαντικές είναι η υποστήριξη τύπων δεδομένων:

- Είναι ευκολότερο να περιγραφεί η επιτρεπόμενη περιεκτικότητα σε έγγραφα
- Είναι ευκολότερο να επικυρωθεί η ακρίβεια των στοιχείων
- Είναι ευκολότερη η συνεργασία με τα στοιχεία από μια βάση δεδομένων

- Είναι ευκολότερο να καθοριστούν οι όψεις (views) δεδομένων (περιορισμοί στα δεδομένα)
- Είναι ευκολότερο να καθοριστούν οι μορφές των δεδομένων
- Είναι ευκολότερο να μετατραπούν τα στοιχεία μεταξύ των διαφορετικών τύπων δεδομένων

Τα XML Schemas χρησιμοποιούν τη σύνταξη της XML

Μια ακόμα σημαντική δυνατότητα των XML Schemas είναι ότι γράφονται σε XML. Επειδή τα XML Schemas γράφονται σε XML έχουν τα εξής πλεονεκτήματα:

- Ο χρήστης δεν είναι απαραίτητο να μάθει μια άλλη γλώσσα
- Ο χρήστης μπορεί να χρησιμοποιήσει το πρόγραμμα σύνταξης της XML για να διαμορφώσει τα αρχεία με τα XML Schemas
- Ο χρήστης μπορεί να χρησιμοποιήσει τον αναλυτή (parser) της XML για να αναλύσει τα αρχεία των XML Schemas
- Ο χρήστης μπορεί να μετασχηματίσει το XML Schema με το XSLT

Τα XML Schemas εξασφαλίζουν τη μετάδοση των δεδομένων

Όταν τα δεδομένα στέλνονται από έναν αποστολέα σε έναν δέκτη είναι αναγκαίο και τα δύο μέρη να έχουν τις ίδιες «προσδοκίες» για το περιεχόμενο.

Με τα XML Schemas, ο αποστολέας μπορεί να περιγράψει τα στοιχεία με έναν τρόπο που ο δέκτης θα τα καταλάβει. Μια ημερομηνία για παράδειγμα όπως 1999-03-11 (σε μερικές χώρες) μπορεί να ερμηνευθεί ως τις 3 Νοεμβρίου ή (σε μερικές άλλες χώρες) ως τις 11, Μαρτίου, αλλά ένα στοιχείο XML με ένα τύπο δεδομένων όπως ο παρακάτω εξασφαλίζει μια αμοιβαία κατανόηση του περιεχομένου επειδή ο τύπος δεδομένων date στην XML απαιτεί το σχήμα YY-MM-DD (Year-Month-Day).

```
<date type="date">1999-03-11</date>
```

Τα XML Schemas είναι επεκτάσιμα

Τα XML Schemas είναι επεκτάσιμα, ακριβώς όπως είναι και η XML, επειδή γράφονται και αυτά σε XML. Με έναν επεκτάσιμο καθορισμό σχημάτων είναι δυνατό να:

- Επαναχρησιμοποιηθεί ένα σχήμα σε άλλα σχήματα
- Δημιουργηθούν καινούριοι τύποι στοιχείων που θα προέρχονται από τύπους που ήδη υπάρχουν
- Γίνεται αναφορά σε πολλαπλά σχήματα από το ίδιο έγγραφο

Τα XMLSchemas τηρούν τους κανόνες σύνταξης της XML

Ένα καλοσχηματισμένο έγγραφο XMLSchema είναι ένα έγγραφο που προσαρμόζεται στους κανόνες σύνταξης της XML:

- Πρέπει να αρχίσει με τη δήλωση XML
- Πρέπει να έχει ένα μοναδικό στοιχείο-ρίζας (root element)
- Όλες οι ετικέτες έναρξης πρέπει να ταιριάζουν με τις ετικέτες τέλους
- Οι ετικέτες τις XML διακρίνουν τους κεφαλαίους από τους μικρούς χαρακτήρες
- Όλα τα στοιχεία πρέπει να είναι κλειστά
- Όλα τα στοιχεία πρέπει να είναι κατάλληλα τοποθετημένα
- Όλες οι τιμές των οντοτήτων (entities) πρέπει να αναφέρονται επακριβώς
- Οι οντότητες στην XML πρέπει να χρησιμοποιηθούν για τους ειδικούς χαρακτήρες

Παράδειγμα δημιουργίας XML Schema εγγράφου

Στην παράγραφο αυτή, γίνεται μια πρώτη εισαγωγή στη σύνταξη των XML Schemas, με την παρουσίαση ενός απλού XML Schema εγγράφου. Ακολουθεί ένα απλό XML έγγραφο, το οποίο αποτελεί και τη βάση για τη δημιουργία του XML Schema εγγράφου:

```
<?xml version="1.0"?>
<Customer>
      <FirstName>John</FirstName>
      <LastName>Smith</LastName>
      <Email>johnsmith@agent.com</Email>
</Customer>
```

Το DTD που απαιτείται για την μοντελοποίηση του XML εγγράφου θα έχει την εξής μορφή:

```
<!ELEMENT Customer (FirstName, LastName, Email)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
```

Το XML Schema που απαιτείται για την μοντελοποίηση του παραπάνω XML εγγράφου θα έχει την παρακάτω μορφή.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="Customer">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="FirstName" type="xs:string"/>
                        <xs:element name="LastName" type="xs:string"/>
                        <xs:element name="Email" type="xs:string"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Το στοιχείο ρίζας ενός XML Schema εγγράφου είναι το στοιχείο `<xs:schema>` μέσα στο οποίο δηλώνεται ο χώρος ονοματοδοσίας ο οποίος αντιστοιχεί στο URI `http://www.w3.org/2001/XMLSchema` και παρέχει τα βασικά στοιχεία και τους τύπους δεδομένων για τη δημιουργία του μοντέλου δομής του εκάστοτε XML εγγράφου.

Για τη δήλωση ενός στοιχείου `Customer` χρησιμοποιείται το στοιχείο `<xs:element>` όπου μέσω της ιδιότητας `name` δηλώνεται το όνομα του στοιχείου. Επειδή το στοιχείο `Customer` περιέχει και άλλα στοιχεία, τότε θα πρέπει να δηλωθεί με τη χρήση του στοιχείου `<xs:complexType>`. Επίσης, με τη χρήση σύνθετων στοιχείων δηλωνονται τα στοιχεία που περιέχουν και ιδιότητες. Το στοιχείο sequence δηλώνει ότι τα στοιχεία που περιέχονται πρέπει να εμφανίζονται με την ίδια σειρά στο XML έγγραφο. Στο παράδειγμα αυτό, θα πρέπει πρώτα να εμφανίζεται το στοιχείο `<FirstName>`, ύστερα το στοιχείο `<LastName>` και τέλος το στοιχείο `<Email>`. Τα στοιχεία αυτά επειδή δεν περιέχουν άλλα στοιχεία ή ιδιότητες ονομάζονται *απλού τύπου* (*simple type*). Στο παράδειγμα αυτό,

κατά τη δήλωση των τριών στοιχείων, χρησιμοποιείται η ιδιότητα `type` του οποίου η τιμή καθορίζει και τον τύπο δεδομένων των στοιχείων αυτών. Ο τρόπος με τον οποίο δηλώθηκε το στοιχείο `Customer` (όπου το στοιχείο `<xs:complexType>` περιέχεται στο στοιχείο `<xs:element>`) ονομάζεται *ανώνυμος τύπος δήλωσης* (*anonymous type definition*)

Ένας δεύτερος τρόπος που επιτρέπει τη δήλωση στοιχείων σύνθετου τύπου είναι ο ακόλουθος:

```
<xs:element name="Customer" type="CustomerType"/>
<xs:complexType name="CustomerType">
      <xs:sequence>
            <xs:element name="FirstName" type="xs:string"/>
            <xs:element name="LastName" type="xs:string"/>
            <xs:element name="Email" type="xs:string"/>
      </xs:sequence>
</xs:complexType>
```

Στο παράδειγμα αυτό, κατά τη δήλωση του στοιχείου `Customer` χρησιμοποιείται η ιδιότητα `type` που συσχετίζει το όνομα του στοιχείου με το μοντέλο περιεχομένου του. Το μοντέλο περιεχομένου καθορίζεται από το στοιχείο `<xs:complexType>` του οποίου η ιδιότητα `name` έχει την ίδια τιμή με αυτή της ιδιότητας `type` του στοιχείου `<xs:element>`.

Με τον τρόπο αυτό παρέχεται η δυνατότητα επαναχρησιμοποίησης του μοντέλου περιεχομένου από διαφορετικά στοιχεία. Αρκεί κατά τη δήλωση αυτών των στοιχείων η τιμή της ιδιότητας `type` να είναι ίδια με το όνομα του μοντέλου περιεχομένων

3.4.3 Δήλωση στοιχείων

Η δήλωση των στοιχείων γίνεται με τη χρήση του στοιχείου `<xs:element>`

```
<xs:element name="bookType" />
```

Αν δηλώνεται ο σύνθετος τύπος (complex type) γι' αυτό το στοιχείο στην αρχή του εγγράφου, είναι δυνατό να ξαναχρησιμοποιηθεί μέσα στο υπόλοιπο του εγγράφου χρησιμοποιώντας την ιδιότητα ref.

```
<xs:element name="Book" />
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="title" />
                  <xs:element name="author" />
            </xs:sequence>
      </xs:complex type>
</xs:element>
<xs:element name="Catalog" >
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="Book"/>
            </xs:sequence>
```

```
      </xs:complexType>
</xs:element>
```

Εδώ χρησιμοποιείται ένας σύνθετος τύπος (ο οποίος χρησιμοποιείται για να περιγράψει ένα μοντέλο περιεχομένου).

Θα μπορούσε ακόμα να δηλωθεί ένας απλός τύπος (simple type), ο οποίος δείχνει πώς δηλώνονται προσωπικά οριζόμενοι τύποι από τον κάθε χρήστη (derived types).

```
<xs:element name=”myelement” type=”mySimpleType” />
```

Θα εξεταστεί αργότερα πώς να δηλώνονται προσωπικά οριζόμενοι τύποι, για την ώρα, αυτό το παράδειγμα σημαίνει ότι το myElement θα δεσμευθεί από τον προσωπικά οριζόμενο τύπο.

Προκαθορισμένο ή Ορισμένο (Default ή Fixed) περιεχόμενο Στοιχείων

Μπορεί επίσης να εφοδιαστεί ένα στοιχείο με προκαθορισμένες ή ορισμένες ιδιότητες. Αν το στοιχείο είναι κενό μέσα σε ένα έγγραφο και καθοριστεί προκαθορισμένο, τότε ένας XML επεξεργαστής κατά την επεξεργασία του στοιχείου αυτού θα θεωρήσει ότι το στοιχείο αυτό περιέχει την προκαθορισμένη του τιμή. Ένα παράδειγμα καθορισμού ενός στοιχείος με προκαθορισμένη τιμή είναι το εξής:

```
<element name= “Subscribe” type= “string” default= “yes”/>
```

Ορισμός Κενών (Null) Τιμών

Για να αναπαραστηθεί ένα στοιχείο που θα έχει μηδενική τιμή όταν στέλνει ή λαμβάνει δεδομένα από μια βάση δεδομένων, μπορεί να χρησιμοποιηθεί η ιδιότητα nullable στη δήλωση του στοιχείου (η οποία παίρνει ένα Boolean του οποίου η προκαθορισμένη τιμή είναι false). Σ’ αυτή την περίπτωση η μηδενική τιμή δε στέλνεται σαν περιεχόμενο του στοιχείου, αλλά είναι ορισμένη σαν ιδιότητα. Για παράδειγμα:

```
<xs:element name=”ackReceived” nullable=”true”/>
```

Συχνότητα εμφάνισης στοιχείων – minOccurs και maxOccurs

Αν υπάρξει η ανάγκη να δηλωθεί η συχνότητα εμφάνισης των παιδιών ενός στοιχείου, υπάρχουν δυο προαιρετικές ιδιότητες που βοηθούν σ’ αυτό, οι minOccurs και maxOccurs. Αν αυτές οι δύο ιδιότητες παραβλεφθούν, τότε εξ᾽ορισμού το στοιχείο πρέπει εμφανίζεται υποχρεωτικά μια και μόνο φορά. Ο ακόλουθος πίνακας παρέχει την αντιστοίχηση των τελεστών πολλαπλότητας των DTD με τις ισοδύναμες τιμές των ιδιοτήτων minOccurs και maxOccurs.

Τελεστής πολλαπλότητας	**minOccurs Value**	**maxOccurs Value**	**Αριθμός των Child Elements**
[none]	1	1	Ένα και μόνο ένα
?	0	1	Κανένα ή ένα
*	0	unbounded	Κανένα ή περισσότερα
+	1	unbounded	Ένα ή περισσότερα

Αν δεν δηλωθεί μια τιμή για το minOccurs, η προκαθορισμένη τιμή του είναι 1, ωστόσο για το maxOccurs δεν .

Αν το minOccurs παρουσιάζεται και το maxOccurs παραλείπεται, η τιμή θεωρείται πως είναι ίση με την τιμή του minOccurs.

```
<xs:element ref=”emailAddress” minOccurs=”1” />
```

θα σήμαινε ότι το στοιχείο emailAddress έχει δηλωθεί πως θα πρέπει να εμφανίζεται τουλάχιστον μια φορά.

Η ειδική τιμή unbounded, που επιτρέπεται μόνο για το maxOccurs, καθορίζει ότι δεν υπάρχει ένα άνω όριο εμφάνισης ενός συγκεκριμένου στοιχείου.

Κενά στοιχεία (Empty Elements)

Δημιουργώντας κενά στοιχεία

Τα κενά στοιχεία δεν έχουν περιεχόμενο, μπορούν όμως να περιέχουν ιδιότητες. Η δήλωση κενών στοιχείων γίνεται με τη χρήση του στοιχείου `<xs:complexType>` του οποίου η ιδιότητα `content` παίρνει την τιμή `empty`.

Στο παράδειγμα που ακολουθεί, δηλώνεται ένα νέο στοιχείο με όνομα `image`, που μπορεί να έχει τρείς ιδιότητες: `source`, `width` και `height`.

```
<xs:element name=" image">
   <xs:complexType content= “empty”>
      <xs:attribute name="source" type="xs:string"/>
      <xs:attribute name="width" type="xs:decimal"/>
      <xs:attribute name="height" type="xs:decimal"/>
   </xs:complexType>
</xs:element>
```

Στοιχεία Πολλαπλού Περιεχομένου (Mixed content elements)

Δημιουργώντας Στοιχεία Πολλαπλού Περιεχομένου

Μέσω των XML Schema, παρέχεται η δυνατότητα για τη δήλωση στοιχείων που υποστηρίζουν πολλαπλό περιεχόμενο, δηλαδή ταυτόχρονα κείμενο και άλλα στοιχεία. Ένα παράδειγμα εγγράφου XML όπου γίνεται χρήση ενός στοιχείου πολλαπλού περιεχομένου είναι το εξής

```
<?xml version=”1.0”>
<reminder>
      Dear <name>John Smith</name>;
            The book <bookTitle>Professional XML Schemas</bookTitle>
      Was supposed to be out for only <maxDaysOut>14</maxDaysOut>
      days. Please return it or pay
      <replacementValue>34.99</replacementValue> €.
      Thank you.
</reminder>
```

Το έγγραφο αυτό χρησιμοποιεί στοιχεία που έχουν ήδη οριστεί, character data, και το καινούργιο στοιχείο <reminder>. Το στοιχείο <reminder> είναι αυτό το οποίο χρησιμοποιεί το μοντέλο πολλαπλού περιεχομένου. Για να δηλωθεί αυτό σε ένα Σχήμα, για αρχή θα δημιουργηθεί ένας νέος ανώνυμος σύνθετος τύπος μέσα στη δήλωση του <reminder>:

```
<xs:element name=”reminder”>
      <xs:complexType>
      .
```

```
      .
      .
      </xs:complexType>
</xs:element>
```

Υπενθυμίζεται ότι το στοιχείο <xs:complexType> έχει την ιδιότητα content με το οποίο ορίζεται ένα μοντέλο περιεχομένου. Στην περίπτωση αυτή το μοντέλο περιεχομένου είναι πολλαπλό:

```
<xs:element name=”reminder”
      <xs:complexType>
      .
      .
      .
      </xs:complexType>
</xs:element>
```

Τώρα, πρέπει να προστεθούν οι δηλώσεις για τα στοιχεία, που μπορούν να χρησιμοποιηθούν μέσα στο στοιχείο <reminder>, έτσι :

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="reminder">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="bookTitle" type="xs:string"/>
        <xs:element name="maxDaysOut">
          <xs:simpleType>
            <xs:restriction base="xs:integer">
              <xs:maxExclusive value="14"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="replacementValue" type="xs:decimal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

3.4.4 Δήλωση Ιδιοτήτων

Όπως παρουσιάστηκε σε προηγούμενο παράδειγμα, οι ιδιότητες μπορούν να δηλωθούν χρησιμοποιώντας το στοιχείο <xs:attribute>. Όταν δηλώνεται μια ιδιότητα χρησιμοποιείται την παρακάτω σύνταξη:

```
<xs:attribute name=”firstName” />
```

Εδώ χρησιμοποιείται η ιδιότητα name για να δωθεί το όνομα της ιδιότητας. Αν πρέπει να είναι διαθέσιμο για διαφορετικά μοντέλα περιεχομένων, δηλώνεται στην αρχή του

εγγράφου. Ύστερα, όταν χρειαστεί να εμφανιστεί σε μια σύνθετου τύπου δήλωση, χρησιμοποιείται η ιδιότητα `ref`:.

```
<xs:attribute name=”firstName”/>
<xs:element name=”fullName” type=”fullNameType” />
<xs:complexType name=”fullNameType”>
      <xs:attribute ref=”firstName”/>
      <xs:attribute name=”lastName”/>
</xs:complexType>
<xs:element name=”contactDetails” type=”contactDetailstype”/>
<xs:complexType name=”contactDetailstype”>
      <xs:attribute ref=”firstName” />
</xs:complexType>
```

Εδώ χρησιμοποιείται η ιδιότητα firstName σε δύο διαφορετικά στοιχεία, οπότε έχει δηλωθεί γενικά, αφού η ιδιότητα lastName χρησιμοποιείται μόνο στο στοιχείο fullName.

Καθορίζοντας πριορισμούς και προκαθορισμένες τιμές ιδιοτήτων

Οι ιδιότητες δηλώνονται με το στοιχείο `<xs:attribute>`. Το στοιχείο αυτό, περιέχει την ιδιότητα `type`, που παρέχει τον (απλό) τύπο της ιδιότητας. Για να διαπιστωθεί αν μια ιδιότητα είναι υποχρεωτική ή προαιρετική, γίνεται χρήση του `<xs:attribute>` και της ιδιότητας `use`.

Για παράδειγμα, έχει προστεθεί μια ιδιότητα ονομαζόμενη phone στον τύπο persons. Αυτή η ιδιότητα είναι τύπου xs:string και η χρήση της είναι προαιρετική.

```
<xs:complexType name="persons">
     <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="street" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
     </xs:sequence>
     <xs:attribute name="phone" type="xs:string" use="required"/>
  </xs:complexType>
```

Οι δυνατές χρήσεις της ιδιότητας `use` είναι οι εξής:

- `Required` (η ιδιότητα είναι υποχρεωτική και μπορεί να έχει οποιαδήποτε τιμή).
- `Optional` (η ιδιότητα είναι προαιρετική και μπορεί να έχει οποιαδήποτε τιμή).
- `Fixed` (η τιμή της ιδιότητας είναι σταθερή και δίνεται με την ιδιότητα `value`).
- `Default` (εάν η ιδιότητα δεν εμφανίζεται, η τιμή της είναι προκαθορισμένη και δίνεται με την ιδιότητα `value`. Εάν εμφανίζεται, η τιμή της είναι αυτή που έχει τεθεί στο XML Schema έγγραφο.
- `Prohibited` (η ιδιότητα δεν πρέπει να εμφανιστεί).

Για παράδειγμα, η παρακάτω δήλωση ορίζει μια ιδιότητα με όνομα `counter` η οποία δέχεται ακέραιες τιμές, η σταθερή τιμή είναι 260 και η εμφάνισή της είναι υποχρεωτική. Η τιμή αυτή είναι σταθερή και δε μπορεί να αλλάξει σε ένα XML έγγραφο.

```
<xs:attribute name="counter" type="xs:integer" use="fixed" value="260"/>
```

Αντίθετα, στην παρακάτω δήλωση ορίζεται η ίδια ιδιότητα της οποίας η τιμή είναι προκαθορισμένη και ίση με 260, σε περίπτωση που δε χρησιμοποιηθεί ή στην αντίθετη περίπτωση, η ιδιότητα περιέχει την τιμή που έχει οριστεί. Η τιμή αυτή βέβαια μπορεί να αλλαξει σε ένα XML έγγραφο ανάλογα με τις ανάγκες του χρήστη.

```
<xs:attribute name="counter" type="xs:integer" use="default" value="260"/>
```

3.4.5 Τα στοιχεία Choice – Sequence – All

Το στοιχείο Choice

Ένα choice δίνει τη δυνατότητα προσδιορισμού ενός αριθμού στοιχείων, εκ των οποίων μόνο ένα θα επιλεχθεί. Για τη δημιουργία ενός Choice, χρησιμοποιείται το <xs:Choice> στοιχείο.

```
<xs:element name = "Customer">
      <xs:complexType>
            <xs:choice>
                  <xs:element name = "FirstName" type= "xs:string"/>
                  <xs:element name = "LastName" type= "xs:string"/>
            </xs:choice>
      </xs:complexType>
</xs:element>
```

Στο παράδειγμα αυτό δημιουργείται μια ομάδα στοιχείων `FirstName` και `LastName`, από τα οποία μόνο ένα θα πρέπει να είναι ενθετημένο στο στοιχείο `Customer`.

Το στοιχείο Sequence

Όπως έχουμε δει, μέσω του στοιχείου <sequence> τα στοιχεία που δηλώνονται σε αυτό το στοιχείο, θα πρέπει να εμφανίζονται με την ίδια σειρά και στο XML έγγραφο που ακολουθεί το συγκεκριμένο μοντέλο.

Το στοιχείο All

'Ολα τα στοιχεία σε ένα all group μπορούν να εμφανίζονται μία ή και καμμία φορά και μπορούν να εμφανίζονται σε οποιαδήποτε σειρά. Το group αυτό πρέπει να χρησιμοποιείται στο ανώτατο επίπεδο περιεχομένων του μοντέλου και τα «παιδιά» group να είναι ανεξάρτητα στοιχεία. Αντίθετα, οποιοδήποτε στοιχείο στο μοντέλο αυτό του περιεχομένου μπορεί να εμφανίζεται όχι παραπάνω από μία φορά. Αυτό σημαίνει ότι τα στοιχεία σε αυτό τον τύπο μπορούν τώρα να εμφανιστούν σε οποιαδήποτε σειρά αλλά μόνο μία φορά το πολύ. Ένα σημαντικό, ακόμη, σημείο είναι ότι εάν χρησιμοποιηθεί το group <xs:all>, πρέπει να περιλαμβάνει όλους τους ορισμούς των στοιχείων σε ένα περιεχόμενο μοντέλο.

Για παράδειγμα αν φανταστεί κανείς ότι μπορούν να γίνουν συναλλαγές με πελάτες που μπορεί να μην έχουν e-mail address και δε ενδιαφέρει αν το όνομα ή το επώνυμο τους είναι το πρώτο στοιχείο παιδί, θα μπορούσε να επαναπροσδιοριστεί έτσι:

```
<element name="PersonName">
      <complexType>
            <all>
                  <element name="FirstName" minOccurs="1"/>
                  <element name="LastName" minOccurs="1"/>
                  <element ref="emailAddress" minOccurs="0" maxOccurs="1"/>
```

```
            </all>
            </complexType>
      </element>
```

Εξαιτίας ενός περιορισμού στη δήλωση του <all>, έπρεπε να δηλωθεί ότι το element – παιδί <emailAddress> πρέπει να εμφανίζεται μία ή καμία φορά, αντί να επιτρέπει πολλαπλές διευθύνσεις e-mail όπως πριν.

Παραδείγματα εγγράφων που χρησιμοποιούν αυτό το μοντέλο περιεχομένου είναι:

```
<Customer>
      <LastName>Rivers</LastName>
      <FirstName>Joan</FirstName>
</Customer>
```

και

```
<Customer>
      <emailAddress>joan@rivers.org</emailAddress>
      <FirstName>Joan</FirstName>
      <LastName>Rivers</LastName>
</Customer>
```

Εφόσον δεν χρειάζεται η ιδιότητα name για το στοιχείο <all>, μπορεί να έχει έναν μοναδικό αναγνωριστικό για αναφορικούς σκοπούς.

3.4.6 Χρήση Σχολίων στα XML Schemas (Schema Annotation)

Εφόσον ένα XML Schema είναι απλά ένα καλά δομημένο XML έγγραφο, μπορεί να περιλαμβάνει σχόλια, χρησιμοποιώντας τη σύνταξη σχολίων της XML:

```
<!--  This is an inline XML comment! -->
```

Όμως, εξαιτίας των κανόνων του επεξεργαστή της XML 1.0, ένας XML επεξεργαστής δεν χρειάζεται να ελέγξει τα σχόλια – όλα τα σχόλια που εμφανίζονται σε ένα XML έγγραφο μπορούν να παραβλεφθούν. Αν και η απόφαση του σχεδιασμού βγάζει νόημα μέσα από τη διαδεδομένη (και προβληματική) μέθοδο της εμπέδωσης των γλωσσών σχεδιασμού μέσω των HTML σχολίων, πρέπει κάποιες φορές να είναι σε θέση ο χρήστης να υπομνηματίζει ένα Σχήμα και να διατηρεί αυτά τα υπομνήματα (annotations).

Τα XML Schemas έχουν εφοδιαστεί με τρία στοιχεία για Μεταδεδομένα Σχημάτων και για εφαρμογές και για χρήστες:

- `<annotation>` - ο πατέρας των άλλων δύο, μπορεί να εμφανίζεται σχεδόν οπουδήποτε μέσα σε ένα schema, συνήθως σαν πρώτο παιδί κάποιων άλλων στοιχείων.
- `<appInfo>` - αυτό το element είναι ορισμένο πληροφορίες των Σχημάτων οι οποίες είναι χρήσιμες για εξωτερικές εφαρμογές.
- `<documentation>` - αυτό το element είναι το μέρος για τα «σχόλια» τα οποία προορίζονται για τους ανθρώπους που χρησιμοποιούν το schema, όπως ένα απόσπασμα, copyright ή άλλες νομικές πληροφορίες.

Είναι σημαντικό να θυμάται κανείς ότι το `<appInfo>` και το `<documentation>` δε μπορούν να χρησιμοποιηθούν μόνα τους, πρέπει να είναι παιδιά του στοιχείου `<annotation>`.

Ένα ενδιαφέρον παράδειγμα χρησιμοποίησης του στοιχείου <appInfo> υπάρχει μέσα στο ίδιο το specification του XML Schema. Το schema που περιγράφει τις περιοριστικές όψεις των απλών datatype περιλαμβάνει αυτό το annotation element. Μια εφαρμογή μετά χρησιμοποιεί αυτά τα στοιχεία για να γενικεύσει ένα επιπρόσθετο κείμενο για το κομμάτι του specification που αναφέρεται στους τύπους δεδομένων.

Χρησιμοποιώντας σωστά αυτά τα στοιχεία, φτιάχνουμε το δικό μας schema self-documenting. Προσθέτοντας ένα style sheet στο schema, μπορούμε να κατασκευάσουμε μια ευανάγνωστη έκδοση που θα εξηγεί τις χρήσεις του. Αυτό είναι ιδιαίτερα χρήσιμο αν θέλουμε να μοιραστούμε το Σχήμα με άλλους, ή να το κάνουμε πρότυπα καθορισμένο.

Schema Annotation

Στα DTD, μπορούν να χρησιμοποιηθούν σχόλια της XML για να προστεθούν σημειώσεις και για να δωθεί τεκμηρίωση. Στα Σχήματα, η κατάσταση είναι λίγο πιο πολύπλοκη. Τα XML Σχήματα ορίζουν τρία νέα στοιχεία που χρησιμοποιούνται για να προστεθούν σημειώσεις σε αυτά, τα <xs:annotation>, <xs:documentation> και <xs:appInfo>.

Το <xs:annotation> συμπεριλαμβάνει τα στοιχεία <xs:documentation> και <xs:appInfo>. Το στοιχείο <xs:documentation> έχει κείμενο. Το στοιχείο <xs:appInfo> έχει σημειώσεις που είναι πρέπουσες για εφαρμογές που διαβάζουν το document. Τέτοιες εφαρμογές μαζεύουν πληροφορίες από το <xs:appInfo>, εάν αυτά τα στοιχεία είναι κατασκευασμένα με τέτοιο τρόπο, ώστε να αναγνωρίζονται.

Παράδειγμα, χρησιμοποίησης των στοιχείων <xs:annotation> και <xs:appInfo>. Το παράδειγμα αυτό είναι από το Σχήμα που εκδίδει ο W3C για τους τύπους δεδομένων που χρησιμοποιεί στα XML Σχήματα, και είναι κομμάτι από αυτό που καλείται Σχήμα όλων των Σχημάτων. Αυτός είναι ο ορισμός του απλού τύπου string και το <appInfo> στοιχείο καθορίζει τί ιδιότητες έχει αυτός ο απλός τύπος.

```
<simpleType name=”string" base="urSimpleType">
      <annotation>
            <appinfo>
            <has-facet name=”length"/>
            <has-facet name=”minLength”/>
      <has-facet name=”maxLength”/>
      <has-facet name=”pattern"/>
      <has-facet name="enumeration”/>
      <has-facet name="maxInclusive”/>
      <has-facet name="maxExclusive"/>
      <has-facet name="minInclusive”/>
      <has-facet name=”minExclusive”/>
                  <has-property name=”ordered” νalue=”true”/>
                  <has-property name="bounded” value=”false”/>
                  <has-property name=”cardinality" value="countably
            inffnite”/>       <has-property name="numeric” value="false"/>

      </appinfo>
      </annotation>
</simpleType>
```

Ένα ακόμα παράδειγμα είναι το εξής

```
<xs:schema xmlns:xsd=http://www.w3.org/1999/XMLSchema>
      <xs:annotation>
            <xs:documentation>
                  Book borrowing transaction schema.
            </xs:documentation>
```

```
      </xs:annotation>
      <xs:element name="transaction" type="transactionType"/>
      <xs:complexType name="transactionType">
            <xs:element name="Lender" type="address"/>
            <xs:element name="Borrower" type="address"/>
            <xs:element ref="note" minOccurs="0"/>
            <xs:element name="books" type="books"/>
            <xs:attribute name="borrowDate" type="xs:date"/>
      </xs:complexType>
</xs:schema>
```

Μάλιστα, το στοιχείο <xs:annotation> μπορεί να χρησιμοποιηθεί στην αρχή των περισσοτέρων οδηγιών των schema, όπως και τα <xs:schema>, <xs:complexType>, <xs:simpleType>, <xs:element> και <xs:attribute>.

3.4.7 Τύποι Δεδομένων

Έχουμε ήδη δει κάποια από τα πλεονεκτήματα των τύπων δεδομένων κι ότι τα XML Schemas παρέχουν δύο βασικούς τύπους δεδομένων:

- **Primitive datatypes** (Στοιχειώδης τύπος δεδομένων): αυτοί που δεν καθορίζονται σε σχέση με άλλους τύπους δεδομένων.
- **Derived datatypes** (απορρέοντες τύποι δεδομένων): αυτοί που καθορίζονται σε σχέση με υπάρχοντες τύπους.

Ας δούμε πώς οι τύποι δεδομένων μπορούν να μας βοηθήσουν να περιορίσουμε το περιεχόμενο των XML εγγράφων.

Απλοί Τύποι (Simple Type)

Το Στοιχείο <simpleType>

Ένας ορισμός Απλού Τύπου χρησιμοποιεί το στοιχείο <simpleType>, τις ιδιότητες του, και κάθε έγκυρο περιοριστικό facet. Τα ονόματα των ιδιοτήτων, και οι primitive datatype ή enumerated τιμές είναι:

- name – NCName
- base – Qname – ΠΡΟΑΙΡΕΤΙΚΟ
- abstract – Boolean – ΠΡΟΑΙΡΕΤΙΚΟ
- derivedBy – (list | restriction) – ΠΡΟΑΙΡΕΤΙΚΟ (default είναι το restriction)

Η σημασία της ιδιότητας name μάλλον είναι προφανής, είναι το όνομα του τύπου δεδομένων που περιγράφεται. Όπως όλα τα ονόματα στα XML Schemas, πρέπει να συμμορφώνεται με τους κανόνες ονοματοδοσίας της XML 1.0.

Η ιδιότητα base είναι το όνομα του datatype base. Πρέπει να χρησιμοποιεί ένα qualified όνομα, ένα όνομα με namespace identifier. Αν αυτό παραλείπεται, τότε το ur-type θεωρείται σαν base type.

Αν η τιμή abstract είναι true, τότε ο τύπος δε μπορεί να εμφανίζεται όπως το type definition στη δήλωση ενός στοιχείου, δε μπορεί να αναφερθεί σαν xsi: type ιδιότητα σε ένα έγγραφο. Αυτή η ιδιότητα είναι προαιρετική, και η default τιμή της είναι false.

Η ιδιότητα derivedBy χρησιμοποιείται για να δηλώσει αν το καινούριο <simpleType> είναι atomic τύπου ή τύπου list.

Το Simple Type αναγνωρίζεται από το όνομα και το target namespace, και θα πρέπει να είναι μοναδικό σε ένα Schema. Καμιά δήλωση Simple Type δε μπορεί να έχει το ίδιο

όνομα με καμιά άλλη δήλωση Simple ή Complex Type μπορεί όμως να είναι ίδια με αυτό των ιδιοτήτων ή των στοιχείων.

Το περιεχόμενο των <simpleType> αποτελείται από ένα ή περισσότερα περιοριστικά facets, τα οποία αναπαριστούνται από empty child elements. Μπορεί, επίσης, να περιέχεται ένα annotation element με το περιεχόμενο. Όπως έχει δειχθεί, η λίστα των περιοριστικών facets εξαρτάται από το base datatype.

Εξαγωγή από Περιορισμούς

Ο built-in derived τύπος negativeInteger είναι ένα παράδειγμα ενός τύπου παραγόμενου από περιορισμό:

```
<xs:simpleType name="negativeInteger" base="xsi:integer">
      <xs:maxInclusive value="-1"/>
</xs:simpleType>
```

Σ' αυτή τη δήλωση, έχει ονομαστεί ένας νέος απλός τύπος δεδομένων το negativeInteger, και ορίστηκε σαν να παράγεται από το built-in τύπο integer.

Εφόσον οι ακέραιοι είναι άπειροι, δεν χρειάζεται να οριστεί ένα κάτω όριο αλλά το άνω όριο είναι η ουσία αυτής της δήλωσης. Αυτό το περιοριστικό facet θα μπορούσε να εκφραστεί και ως εξής:

```
<xs:maxExclusive value="0"/>
```

Παρατίθεται ένα πιο πολύπλοκο simple datatype το οποίο χρησιμοποιεί το facet *pattern.* Οι αριθμοί τηλεφώνου στη Νότια Αμερική είναι δεκαψήφιοι και από ένα τριψήφιο κωδικό περιοχής(Area Code), ένα τριψήφιο τοπικό αριθμό (local exchange) και έναν τετραψήφιο αριθμό(local number). Ο κωδικός περιοχής και ο τοπικός αριθμός έχουν κάποιους επιπρόσθετους περιορισμούς. Το πρώτο ψηφίο είναι πάντα 0 ή 1, εφόσον αυτοί οι αριθμοί χρησιμοποιούνται για υπεραστικά τηλεφωνήματα.

Κάποιες φορές οι κωδικοί περιοχής απαιτούν ένα δεύτερο ψηφίο να περιορίζεται σε 0 ή 1, σαν ένας τρόπος για να ξεχωρίζουν τα τοπικά τηλεφωνήματα (7 ψηφία) από τα υπεραστικά (10 ψηφία). Αυτό το ξεχωριστό στυλ του κωδικού περιοχής μπορεί να καθοριστεί ως εξής:

```
<xs:simpleType name="AreaCode" base="xsi:string">
      <xs:minLength value="3"/>
      <xs:maxLength value="3"/>
      <xs:pattern value="[2-9] [0-1][0-9]" />
</xs:simpleType>
```

Ένα παράδειγμα ενός derived datatype στην περίπτωση ενός εγγράφου θα μπορούσε να είναι:

```
<TelephoneNumber>
      <AreaCode>312</AreaCode>
      <Exchange>555</Exchange>
      <Number>1212</Number>
</TelephoneNymber>
```

Εφόσον τα παλιά δίκτυα έχουν αντικατασταθεί με ψηφιακά αυτός ο περιορισμός δε χρειάζεται πια. Οπότε θα μπορούσε να δηλωθεί ένας λιγότερο περιορισμένος κωδικό περιοχής πιο απλά σαν έναν ακέραιο μεταξύ 200 – 999:

```
<simpleType name="AreaCode" base="xsi:integer">
      <minInclusive value="200"/>
```

```
      <maxInclusive value="999"/>
</simpleType>
```

Το νέο αυτό στυλ θα επέτρεπε ένα κωδικό περιοχής που παλιότερα δε θα ήταν έγκυρος:

```
<AreaCode>925</AreaCode>
```

Εξαιτίας του ότι στο καινούργιο αυτό στυλ οι κωδικοί των περιοχών είναι λιγότερο περιορισμένοι από τους αρχικούς, δε μπορεί να εξαχθεί ο καινούργιος τύπος απ' τον παλιό – η παραγωγή από επέκταση περιορίζεται στους complex datatypes. Άλλος ένας λόγος που δε μπορεί να εξαχθεί ο νέος απ' το παλιό είναι ότι έχει αλλάξει το base type από string σε integer.

Εξαγωγή από λίστα

Οι τύποι δεδομένων που εξάγονται από λίστα περιέχουν μια λευκή-κενή προκαθορισμένη λίστα τιμών η οποία συμμορφώνεται στις αρχές της base type. Ένας list τύπος δεδομένων πρέπει να εξάγεται από έναν atomic τύπο δεδομένων. Δε μπορεί να εξαχθεί από έναν άλλο τύπο λίστας. Ο ατομικός τύπος πρέπει επίσης να είναι κατάλληλος για να συγκαταλεχθεί σε μια λίστα – τα φυσικά strings περιέχουν whitespace τα οποία δημιουργούν προβλήματα όταν προσπαθούμε να φτιάξουμε μια λίστα από strings.

Για παράδειγμα θα μπορούσε να δημιουργηθεί μια απλή λίστα από floating-point αριθμούς:

```
<simpleType name="listOfFloats" base="xsi:float" derivedBy="list">
```

Ένα παράδειγμα αυτού του τύπου δεδομένων στην περίπτωση ενός εγγράφου θα μπορούσε να είναι:

```
<ListOfFloats>-INF -1.02E01  -0.42e1 0 </ListOfFloats>
```

Αυτό το παράδειγμα έχει μήκος 4. Αν χρειαζόταν να περιοριστεί ακόμα περισσότερο ο τύπος δεδομένων ώστε να απαιτεί να έχει μήκος 4, θα μπορούσε να γίνει χρησιμοποιώντας το facet length. Η δήλωση του νέου τύπου θα ήταν κάπως έτσι:

```
<xs:simpleType name="listOfFloats" base="xsi:float" derivedBy="list">
      <xs:restrictions>
            <xs:length value="1"/>
      </xs:restrictions>
</xs:simpleType>
```

Αυτό περιορίζει τον τύπο στο να είναι μια λίστα με ακριβώς 4 αντικείμενα. Αν χρειαζόταν να δημιουργηθεί μια λίστα με κυμαινόμενο μήκος, αλλά περιορισμένο ανάμεσα σε κάποιες συγκεκριμένες τιμές, τότε θα οριζόταν ένα βελησνεκές:

```
<xs:simpleType name="listOfFloats" base="xsi:float" derivedBy="list">
      <xs:restrictions>
            <xs:minLength value="1" />
            <xs:maxLength value="100" />
      </xs:restrictions>
</simpleType>
```

Αυτή η δήλωση επιτρέπει κάθε λίστα που να περιέχει από 1 έως 100 αντικείμενα.

Δημιουργώντας Απλούς Τύπους χρησιμοποιώντας Facets

Με τη χρησιμοποίηση των facets ορίζονται οι τύποι δεδομένων που μπορούν να έχουν οι απλοί τύποι. Για παράδειγμα, έστω ότι δημιουργείται ένα simple type ονομαζόμενο dayOfMonth, που μπορεί να κρατήσει μόνο τιμές μεταξύ 1 έως 31. Στην περίπτωση αυτή,

μπορεί να ορισθεί με τον τρόπο αυτό, με τη χρήση δύο facets, των minInclusive και maxInclusive:

```
<xs:simpleType name=”dayOfMonth” base=”xs:integer”>
      <xs:restrictions>
            <xs:minInclusive value=”1”/>
            <xs:maxInclusive value=”31”/>
      </xs:restrictions>
</xs:simpleType>
```

Τώρα που έχει δημιουργηθεί αυτός ο νέος τύπος, μπορούν να οριστούν στοιχεία και attributes αυτού του τύπου.

Ο τύπος catalogID που ορίζεται παρακάτω, είναι ακόμα πιο δυνατός από τον απλό τύπο dayOfMonth. Ο απλός τύπος catalogID χρησιμοποιεί την ιδιότητα facet για να ορίσει μια regular expression, που πρέπει να ικανοποιεί τιμές string κειμένων.

```
<xs:simpleType name=”dayOfMonth” base=”xs:integer”>
      <xs:restrictions>
            <xs:pattern value=”\d{3}-\d{4}-\d{3}”/>
      </xs:restrictions>
</xs:simpleType>
```

Στην περίπτωση αυτή, το κείμενο στον τύπο simpleType πρέπει να ταιριάζει με την απλή έκφραση “\d{3}-\d{4}-\d{3}”, που ορίζει τρία ψηφία, μια παύλα, τέσσερα ψηφία, μια ακόμα παύλα και τρία ψηφία.

Ο τύπος catalogID είναι ο τύπος της ιδιότητας bookID του στοιχείου <book>. Έτσι, μπορεί να συγκεκριμενοποιηθούν οι τιμές bookID στο book.xml, ταιριάζοντας την απλή έκφραση που χρησιμοποιήθηκε για αυτήν την ιδιότητα:

```
<book bookID=”123-4567-890”>
      <bookTitle>Earthquakes for Breakfast</bookTitle>
      <pubDate>2001-10-20</pubDate>
      <replacementValue>15.95</replacementValue>
      <maxDaysOut>14</maxDaysOut>
</book>
```

Με το facet enumeration μπορεί να οριστεί μια απαρίθμηση τιμών, όπως γίνεται με τα DTDs. Με τη χρήση μιας απαρίθμησης μπορούν να περιοριστούν οι δυνατές τιμές ενός simple type σε μια λίστα τιμών που είναι ήδη ορισμένη.

Για παράδειγμα, για να οριστεί ένας simple type ονομαζόμενος weekday, του οποίου οι τιμές μπορούν να είναι: “Sunday”, “Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday” και “Saturday”, ο τύπος αυτός θα οριστεί ως εξής:

```
<xs:simpleType name="weekday">
     <xs:restriction base="xs:string">
       <xs:enumeration value="Sunday"/>
       <xs:enumeration value="Monday"/>
       <xs:enumeration value="Tuesday"/>
       <xs:enumeration value="Wednesday"/>
       <xs:enumeration value="Thursday"/>
```

```
        <xs:enumeration value="Friday"/>
        <xs:enumeration value="Saturday"/>
      </xs:restriction>
   </xs:simpleType>
```

Χρησιμοποιώντας Ανώνυμους Τύπους Ορισμού (Anonymous Type Definitions)

Μπορεί να χρησιμοποιηθεί ο ανώνυμους τύπους ορισμού (anonymous type definition) για την αποφυγή ορισμού ενός νέου τύπου από την αρχή. Η χρησιμοποίηση του anonymous type definition απλά σημαίνει την ταυτόχρονη χρησιμοποίηση των στοιχείων <xs:simpleType> και <xs:complexType> μέσα στη δήλωση του στοιχείου <xs:element>. Στην περίπτωση αυτή, δεν ορίζεται μια συγκεκριμένη τιμή στην ιδιότητα type στο στοιχείο <xs:element>, γιατί ο anonymous type που χρησιμοποιείται δεν έχει όνομα.

Στο παράδειγμα που ακολουθεί θα χρησιμοποιηθεί ένα anonymous type definition για το στοιχείο <book>. Το στοιχείο αυτό «κρατά» τα στοιχεία <bookTitle>, <pubDate>, <replacementValue> και <maxDaysOut>. Θα έχει ακόμα ένα attribute που θα ονομάζεται bookID, ώστε να φαίνεται καλό για να δημιουργηθεί από ένα complex type. Όμως, αντί να δηλωθεί ένας ξεχωριστός σύνθετος τύπος, θα εισαχθεί το στοιχείο <xs:complexType> μέσα στο <xs:element> lement, που θα ορίζει το <book>:

```
<xs:complexType>element inside the <xs:element> element that declares
<book>:

<xs:element name=”book” minOccurs=”0” maxOccurs=”10”>
      <xs:complexType>
      .
      .
      .
      </xs:complexType>
</xs:element>
```

Τώρα, προστίθενται ελεύθερα τα στοιχεία που είναι μέσα στο <book> element- χωρίς να ορίζεται κανένα όνομα ή ξεχωριστός complex type:

```
 <xs:element name=”book” minOccurs=”0” maxOccurs=”10”>
      <xs:complexType>
            <xs:sequence>
                  <xs:element name=”bookTitle” type=”xs:string”/>
                  <xs:element name=”pubDate” type=”xs:date” minOccurs=”0”/>
                  <xs:element name=”replacementValue” type=”xs:decimal”/>
                  </xs:sequence>
      </xs:complexType>
</xs:element>
```

Μπορούν να χρησιμοποιηθούν ακόμα και απλοί anonymous types. Για παράδειγμα, το στοιχείο <maxDaysOut> κρατά τον αριθμό των περισσότερων ημερών που έχει εκδοθεί ένα βιβλίο. Για να ορισθεί ότι οι περισσότερες ημέρες που έχει εκδοθεί ένα βιβλίο είναι 14, χρησιμοποιείται ένας νέος τύπος simple anonymous type, για να μπορεί να χρησιμοποιηθεί το facet maxExclusive, έτσι:

```
<xs:element name=”book” minOccurs=”0” maxOccurs=”10”>
      <xs:complexType>
```

```
            <xs:sequence>
                  <xs:element name="bookTitle" type="xs:string"/>
                  <xs:element name="pubDate" type="xs:date" minOccurs="0"/>
                  <xs:element name="replacementValue" type="xs:decimal"/>
                  <xs:element name="maxDaysOut">
                        <xs:simpleType base="xs:integer">
                              <xs:restrictions>
                                    <xs:maxExclusive value="14"/>
                              </xs:restrictions>
                        </xs:simpleType>
                  </xs:element>
                  .
                  .
                  .
            </xs:sequence>
      </xs:complexType>
</xs:element>
```

Ακόμη, μπορεί να περιληφθεί η δήλωση attribute σε anonymous type definitions, έτσι:

```
<xs:element name="book" minOccurs="0" maxOccurs="10">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="bookTitle" type="xs:string"/>
                  <xs:element name="pubDate" type="xs:date" minOccurs="0"/>
                  <xs:element name="replacementValue" type="xs:decimal"/>
                  <xs:element name="maxDaysOut">
                        <xs:simpleType base="xs:integer">
                              <xs:restrictions>
                                    <xs:maxExclusive value="14"/>
                              </xs:restrictions>
                        </xs:simpleType>
                  </xs:element>
            </xs:sequence>
            <xs:attribute name="bookID" type="catalogID"/>
      </xs:complexType>
</xs:element>
```

Σύνθετοι Τύποι (Complex Types)

Ορισμός Σύνθετων Τύπων

Όπως έχει αναφερθεί, τα στοιχεία δηλώνονται χρησιμοποιώντας το <xs:element> και οι ιδιότητες το <xs:attribute>. Στη συνέχεια καθορίζεται το μοντέλο περιεχομένου του στοιχείου <photographer>.

```
<xs:element name=”photographer” type= “PhotographerType”>
<xs:complexType name=”PhotographerType”>
      <xs:attribute name=”firstName” type=”xs:string”/>
      <xs:attribute name=”lastName” type=”xs:string”/>
      <xs:attribute name=”BirthDay” type=”xs:date”/>
      <xs:attribute name=”dayTakenPhoto” type=”xs:date”/>
</xs:complexType>
</xs:element>
```

Άρα όποτε χρησιμοποιείται ένα στοιχείο Photographer, πρέπει να έχει τέσσερις ιδιότητες, δύο από τις οποίες πρέπει να έχουν τιμή κειμένου και ο άλλες δύο τιμή date. Όμως αν η παρουσία των στοιχείων παιδιά ήταν υποχρεωτική, θα έπρεπε να είχαν δηλωθεί έτσι:

```
<xs:element name="photographer">
     <xs:complexType>
       <xs:sequence>
          <xs:element name="name" type="xs:string"/>
          <xs:element name="lastName" type="xs:string"/>
          <xs:element name="birthDay" type="xs:date"/>
          <xs:element name="dayTakenPhoto" type="xs:date"/>
       </xs:sequence>
     </xs:complexType>
  </xs:element>
```

Εδώ έχει προστεθεί το στοιχείο <sequence>, το οποίο σημαίνει πως ότι τα στοιχεία πρέπει να εμφανίζονται με τη σειρά που έχουν δηλωθεί.

Για να ενισχυθεί η έννοια των τύπων, τα firstName, lastName, birthday και dayTakenPhoto ιδιότητες και elements σ’ αυτό το παράδειγμα περιέχουν το simple type “string” και “date”, ενώ το PhotographerType είναι complexType (περιγράφει το μοντέλο περιεχομένου). Παρόλα αυτά, και τα δύο χρησιμοποιούν την ιδιότητα type για να προσδιορίσουν τον τύπο.

Στοιχειώδης τύπος (Primitive Type)

Οι Στοιχειώδεις τύποι αποτελούν τη βάση για όλους τους υπόλοιπους τύπους, και δε μπορούν να καθοριστούν από μικρότερες συνιστώσες. Εξ’ ορισμού, δε μπορούν να περιέχουν στοιχεία ή ιδιότητες, αφού είναι οι βασικοί τύποι από τους οποίους αντλούνται οι άλλοι.

Οι Στοιχειώδεις τύποι μπορούν να χρησιμοποιηθούν για τιμές στοιχείων ή ιδιοτήτων, αλλά δε μπορούν να περιέχουν παιδιά στοιχεία ή ιδιότητες. Οι primitive types είναι πάντα ενσωματωμένοι.

Ένα σύνολο χαρακτήρων είναι ένας κοινός στοιχειώδης τύπος δεδομένων γνωστός σαν string datatype.

Μπορεί να φαίνεται πως ο διαχωρισμός μεταξύ των primitive και των derived datatypes να είναι λίγο αυθαίρετος, γι’ αυτό η διάταξη των Schemas δεν ανταποκρίνεται απαραίτητα σε αυτούς μια συγκεκριμένης γλώσσας προγραμματισμού. Όπως ακριβώς ορίζεται ένα string ή ένα float στη Java και στη C++, έτσι ορίζεται και στα XML Schemas.

Στοιχειώδης τύπος	**Περιγραφή**
String	Αναπαριστά χαρακτήρες τύπου string στην XML

Boolean	Αναπαριστά true ή false.
Float	Standard των πραγματικών αριθμών που παρουσιάζονται με ακρίβεια σε 32-bit point τύπου float
Double	Standard των πραγματικών αριθμών που παρουσιάζονται με ακρίβεια σε 64-bit point τύπου float
Decimal	Αναπαριστά δεκαδικούς αριθμούς
TimeDuration	Αναπαριστά χρονική διάρκεια
RecurringDuration	Αναπαριστά ένα συγκεκριμένο χρονικό διάστημα το οποίο επαναλαμβάνεται με συγκεκριμένη συχνότητα, ξεκινώντας από μια δεδομένη χρονική στιγμή
Binary	Αναπαριστά αυθαίρετα δυαδικά δεδομένα
UriReference	Αναπαριστά ένα URI.

Στοιχειώδης τύπος	**Περιγραφή**
ID	Αναπαριστά την ιδιότητα ταυτότητα(ID) όπως αυτή ορίζεται στην σύσταση XML 1.0
IDREF	Αναπαριστά την ιδιότητα IDREF όπως αυτή ορίζεται στην σύσταση XML 1.0
ENTITY	Αναπαριστά την ιδιότητα οντότητα (ENTITY) όπως αυτή ορίζεται στην σύσταση XML 1.0
Qname	Αναπαριστά τα έγκυρα XML ονόματα αυτά ορίζονται στην σύσταση XML 1.0

Απορρέοντες Τύποι (Derived Type)

Ένας απορρέοντας τύπος δεδομένων ορίζεται σε σχέση με ένα υπάρχον τύπο δεδομένων (γνωστό σαν το based type του). Οι απορρέοντες τύποι δεδομένων μπορεί να έχουν ιδιότητες, και μπορεί να έχουν στοιχεία ή να είναι πολύπλοκου περιεχομένου.

Περιπτώσεις των απορρέοντων τύπων δεδομένων μπορεί να περιέχουν κάθε καλογραμμένο XML το οποίο είναι έγκυρο σύμφωνα με τον προσδιορισμό του τύπου δεδομένων. Μπορεί να είναι ενθετημένα ή να εξάγονται από το χρήστη.

Νέοι τύποι μπορούν να εξάγονται είτε από ένα απορρέοντα τύπο δεδομένων είτε από ένα άλλο απορρέοντα τύπο δεδομένων. Γι' αυτό οι ακέρραιοι τύποι των XML Σχημάτων εξάγονται από τον τύπο καθορισμού δεκαδικών αριθμών, ο οποίος είναι ο βασικός τύπος του. Μπορούμε εναλλακτικά να εξάγουμε ένα ακόμα πιο περιορισμένο τύπο ακεραίου. Αυτό είναι ένα παράδειγμα δομής ενός XML Σχήματος γνωστό ως ορισμός απλού τύπου. Χρησιμοποιεί το στοιχείο <simpleType> για να περιγράψει έναν εξαγχθέντα ακέρραιο τύπο δεδομένων ο οποίος είναι περιορισμένος σε αρνητικές τιμές.

```
<xs:simpleType name="negative integer">
      <xs:restriction base="xs:integer">
            <xs:minInclusive value="bounded"/>
            <xs:minInclusive value="-1"/>
      </xs:restriction>
</xs:simpleType>
```

Όπως και για τους απορρέοντες τύπους δεδομένων η W3C έχει ορίσει ένα σύνολο ενθετημένων απορρέοντων τύπων δεδομένων. Αυτοί οι τύποι ήταν τόσο κοινοί που

θεωρήθηκε πως θα έπρεπε να είναι ένα αναπόσπαστο μέρος της σύστασης των XML Σχημάτων.

Προκαθορισμένοι απορρέοντες τύποι

Απορρέον Τύπος	Περιγραφή	Base Type
CDATA	Αναπαριστά whitespace κανονικοποιημένα strings	string
token	Αναπαριστά strings με σύμβολα	Token
language	Αναπαριστά φυσική γλώσσα identifiers	Token
NMTOKEN	XML 1.0 NMTOKEN	Token
NMTOKENS	XML 1.0 NMTOKENS	NMTOKEN
Name	Αναπαριστά ονόματα XML	Token
NCName	Αναπαριστά non-colonized ονόματα XML	Name
integer	Έναν ακέραιο αριθμό	Integer
long	Εξάγεται από το integer θέτοντας σαν μέγιστη τιμή το 9223372036854775807 και σαν ελάχιστη το -9223372036854775808	Integer
short	Εξάγεται από το integer θέτοντας μέγιστη τιμή το 32767 και ελάχιστη το -32768	Integer που εξάγεται από το long
byte	Εξάγεται από το short θέτοντας σαν μέγιστη τιμή το 127 και σαν ελάχιστη το –128	Short
data	Αναπαριστά μια συγκεκριμένη ημερομηνία	Χρονική περίοδος που εξάγεται από το recurringDuration

Ατομικός Τύπος (Atomic Type)

Ένας ατομικός τύπος έχει τιμή που δε μπορεί να διαιρεθεί, τουλάχιστον όχι μέσα σε ένα XML Σχήμα περιβάλλον. Θα πρέπει να σημειωθεί πως οι ατομικοί δεν είναι ανάλογοι των απορρεόντων.

Οι αριθμοί και τα κείμενα είναι ατομικοί τύποι εφόσον οι τιμές τους δεν μπορούν να περιγραφούν χρησιμοποιώντας μικρότερα κομμάτια. Αυτό είναι προφανές, αλλά το κείμενο (string) δε μπορεί να διαιρεθεί σε μικρότερα κομμάτια, τους χαρακτήρες; Αν και αυτό είναι αληθές με μια αφαιρετική λογική, στα XML Σχήματα δεν υπάρχει η έννοια του χαρακτήρα σαν τύπος δεδομένων άρα το string είναι ένας ατομικός απορρέοντας τύπος

```
<atom>This string is as fine as we can slice textual data – there are no
character atoms in XML Schema.</atom>
```

Το δεύτερο παράδειγμα είναι ένας ατομικός απορρέοντας τύπος. Είναι μια ημερομηνία η οποία θα μπορούσε να εξαχθεί από ένα απορρέοντα τύπο κειμένου, όμως στο XML Σχήμα προέρχεται έμμεσα από τρεις άλλους τύπους, αλλά δε μπορεί να διαιρεθεί:

```
<atom2>1927-01-16</atom2>
```

Το τρίτο παράδειγμα είναι επίσης ένας ατομικός απορρέοντας τύπος. Ένας ακέραιος προέρχεται από έναν float απορρέοντα τύπο.

```
<atom3>469557</atom3>
```

Τύποι Λίστας (List Type)

Ένας τύπος λίστας έχει τιμή που αποτελείται από μια ακολουθία ατομικών τιμών πεπερασμένου μήκους. Αντίθετα με άλλες γλώσσες προγραμματισμού, ένας τύπος δεδομένων λίστας του XML Σχήματος δε μπορεί να φτιαχτεί από άλλες λίστες. Αυτός ο τύπος μπορεί να θεωρηθεί σαν ειδική περίπτωση του πιο γενικού συνόλου ή συλλογής τύπου δεδομένων.

Οι τύποι λίστας είναι πάντα απορρέοντες τύποι, οι οποίοι πρέπει να είναι οριοθετημένοι από λευκούς χαρακτήρες, όπως οι IDREFS ή οι NMTOKENS τύποι ιδιοτήτων, που δηλώνονται στην XML 1.0. Όμως, ένας τύπος λίστας πρέπει να επιτρέπει την παρουσία κενών, αλλά δε μπορεί να χρησιμοποιηθεί κανένα white space μέσα στις ιδιαίτερες τιμές των αντικειμένων μιας λίστας.

Μια σημαντική θεώρηση για ένα list type είναι το μήκος του – αυτή η περιγραφική τιμή είναι πάντα ο αριθμός των αντικειμένων μιας λίστας, και δεν έχει σχέση με τον αριθμό των χαρακτήρων που αναπαριστούν τα αντικείμενα.

Για παράδειγμα, μπορεί να οριστεί ένας απλός derived list type για το χρήστη για γενικά μεγέθη:

```
<xs:simpleType name="sizes">
      <xs:list itemType="size"/>
</xs:simpleType>
```

Αυτός ο καινούργιος τύπος δεδομένων θα μπορούσε να χρησιμοποιηθεί μετά μέσα σε ένα πιο συγκεκριμένο element σε ένα έγγραφο:

```
<dressSizes type="sizes">small medium large xlarge </dressSizes>
```

Απ' την άλλη πλευρά, το να χρησιμοποιείται whitespace σαν σημαία δημιουργεί προβλήματα όταν θελήσουμε να δημιουργήσουμε λίστες ή strings. Μέσα σ' αυτό το πλαίσιο, το ακόλουθο παράδειγμα δεν είναι αληθές – είναι μια λίστα έξι αντικειμένων:

```
<bad_list>this is a single list item</bad_list>
```

Οι list types μπορούν να φανούν ένας πολύ χρήσιμος τρόπος για να αναπαραστηθούν συλλογές αριθμών, ή standard attribute types της XML 1.0, όπως IDREFs ή NMTOKENS. Απ' την άλλη, όλα αυτά είναι άχρηστα για χειρισμό δεδομένων κειμένου τα οποία περιέχουν οτιδήποτε άλλο εκτός από λέξεις – προτάσεις και μεγαλύτερα κομμάτια κειμένου δε μπορούν να αναπαρασταθούν με ένα list type.

Τύποι Ενώσεως (Union Type)

Οι Union types επιτρέπουν να δημιουργηθεί ένα στοιχείο ή η τιμή μιας ιδιότητας από atomic ή/ και list types.Με τη θεώρηση πως υπάρχουν μικρές συντομεύσεις για τα μεγέθη στα ρούχα (s, m , l, xl) που ονομάζονται shortSize, θα μπορούσε να δημιουργηθεί ένας τύπος που θα επέτρεπε τη χρησιμοποίηση είτε του πλήρους ονόματος του μεγέθους είτε τη συντόμευσή του. Μπορεί να δημιουργηθεί ένας τύπος και να οριστεί με την ιδιότητα memberType:

```
      <xs:simpleType name="deliveryUnion">
```

```
            <Union memberTypes="sizes shortSize" />
      </xs:simpleType>
```

3.4.8 Δημιουργία νέων τύπων δεδομένων

Περιοριστικά facets

Τα περιοριστικά facets είναι εκείνα που οριοθετούν το πεδίο τιμών ενός απορέοντα τύπου. Οι προωτογενείς τύποι δεν έχουν περιοριστικά facets, αλλά μπορούν να προστεθούν δημιουργώντας ένα derived τύπο ο οποίος εξάγεται από περιορισμό.

Υπάρχουν πολλοί τύποι περιοριστικών facets, που χρησιμοποιούνται σε τρία group απλών τύπων:

- Facets που χρησιμοποιούνται μόνο σε απλούς τύπους που είναι ordered
  length, minLength, maxLength
  pattern
  enumeration
- Facets που χρησιμοποιούνται μόνο σε απλούς τύπους που ορίζουν δεσμούς (bounds)
  minExclusive, maxExclusive, minInclusive, maxInclusive
  precision, scale
  encoding
- Facets που χρησιμοποιούνται σε προσωρινούς απλούς τύπους
  duration, period

Τα facets length, minLength, maxLength

Αυτά τα τρία facets ασχολούνται με τον αριθμό των μονάδων μήκους ενός τύπου δεδομένων, η τιμή των οποίων πρέπει να είναι πάντα ένας μη αρνητικός ακέραιος. Η φύση των μονάδων θα ποικίλει, εξαρτώμενη από τη βάση του τύπου δεδομένων:

- Από αυτούς που εξάγονται από το string type, το length, είναι ένας αριθμός από το Unicode code points και είναι σημαντικό να θυμόμαστε ότι κάθε χαρακτήρας Unicode μπορεί να είναι 8,16 ή 32-bits long, ή ακόμα ποικίλων ακολουθιών μηκών των 8-bit.
- Τύποι που εξάγονται από το binary type measure length όπως ο αριθμός των οκτάδων των δυαδικών δεδομένων.
- List types καθορίζουν το length όπως ο αριθμός των αντικειμένων στη λίστα.

Τα minLength και maxLength facets είναι ο ελάχιστος και μέγιστος επιτρεπόμενος αριθμός μονάδων για τον datatype. Για παράδειγμα μπορούν να χρησιμοποιηθούν αυτά για τον περιορισμό ορισμένων datatype να είναι πάντα 3-digit αριθμοί:

```
<xs:simpleType name="AreaCode">
      <xs:restriction base="xs:string">
            <xs:minLength value="3"/>
            <xs:maxLength value="3"/>
      </xs:restriction>
</xs:simpleTYpe>
```

Το facet pattern

Αυτό το facet είναι ένας περιορισμός στο lexical space του datatype, το οποίο εμμέσως περιορίζει το value space. Ένα pattern είναι μια κανονική έκφραση (regex) ώστε η lexical αναπαράσταση ενός datatype θα πρέπει να ταιριάζει ώστε το τελευταίο literal να θεωρείται έγκυρο. Η regex γλώσσα που χρησιμοποιείται στα XML Schema είναι ίδια με αυτή που έχει δηλωθεί για την Perl γλώσσα προγραμματισμού.

Το facet whitespace

Αυτό το facet αποτελεί έναν περιορισμό όσον αφορά το αν επιτρέπεται ένα whitespace ή όχι.

Το facet enumeration

Αυτό το facet μοιάζει πολύ με το specification ενός DTD για μια λίστα στοιχείων τύπου choice ή για τις αριθμήσιμες τιμές ενός attribute. Το enumeration περιορίζει ένα χώρο τιμών για ένα συγκεκριμένο σετ τιμών – εάν η τιμή δεν προσδιορίζεται στο schema, δεν είναι έγκυρη.

Τα Facet minExclusive, maxExclusive, minInclusive, maxInclusive

Όλα αυτά τα facets μπορούν μόνο να αναφερθούν σε ένα datatype το οποίο έχει ένα order relation.

- Τα δύο "min" facets ορίζουν το ελάχιστο του χώρου τιμών.
- Τα δύο "max" facets ορίζουν το μέγιστο.
- Ένα exclusive όριο σημαίνει ότι η οριακή τιμή δεν περιλαμβάνεται στο χώρο τιμών (εννοώντας ότι για όλες τις τιμές V στο χώρο τιμών ισχύει,minExclusive<V<maxExclusive)
- Ένα inclusive όριο είναι ένα που περιλαμβάνεται μέσα στο χώρο τιμών.
  Βέβαια, αυτοί οι δύο τύποι ορίων δεν συσχετίζονται – ένα κάτω όριο μπορεί να είναι exclusive, ενώ ένα άνω όριο μπορεί να είναι inclusive. Σαφώς, πρέπει να επιλεγεί ένας από τους δύο τύπους ορίων για κάθε περίπτωση – είναι αδύνατο ένα όριο να είναι ταυτόχρονα inclusive και exclusive!

Εάν το στοιχείο minOccurs είναι 0, πρέπει να σιγουρευτείτε ότι η στήλη που το αντιπροσωπεύει στη βάση μπορεί να περιλαμβάνει null τιμές.

Τα facet precision, scale

Αυτά τα δύο facets χρησιμοποιούνται σε όλους τους datatypes που προκύπτουν από το decimal type. precision είναι ο μέγιστο αριθμός των decimal digits που επιτρέπεται γιο όλο τον αριθμό και scale είναι ο μέγιστος αριθμός ψηφίων στο κλασματικό μέρος του αριθμού.

Το facet encoding

Αυτό το facet είναι περιοριστικό για το lexical space ενός datatype που προκύπτει από τη δυαδική base type. Η τιμή αυτού του facet μπορεί να είναι είτε hex είτε base64.

Εάν η τιμή είναι hex, τότε κάθε δυαδικό byte είναι κωδικοποιημένο σαν ένα διψήφιο δεκαεξαδικό αριθμό, χρησιμοποιώντας τα 10 ψηφία ASCII και τα γράμματα A – F. Για παράδειγμα, το hex-κωδικοποιημένο string "312D322D33" είναι η κωδικοποιημένη έκδοση του ASCII string "1-2-3".

Εάν η τιμή είναι base64, τότε ολόκληρη η δυαδική ακολουθία είναι κωδικοποιημένη με την ευρεία χρήση για το internet standard Base64 Content-Transfer-Encoding μεθόδου.

Τα facet duration, period

Αυτά τα facets χρησιμοποιούνται μόνο για τον recurringDuration datatype και τους datatypes που προκύπτουν από αυτόν. Η τιμή πρέπει να είναι πάντα timeDuration.

- Το duration facet είναι η διάρκεια των τιμών recurringDuration.
- Το period facet είναι η συχνότητα εμφάνισης αυτών των τιμών.

Περιοριστικά facets σε μια βάση δεδομένων

Όταν δημιουργείται ένας εξαγόμενος datatype, πρέπει ακόμα να δημιουργηθεί ένας από το χρήστη οριζόμενος datatype για το ανταποκρινόμενο πεδίο της βάσης δεδομένων που ταιριάζει με τους περιορισμούς που υπάρχουν στο XML datatype.

3.5 Κανόνες επιλογής μεταξύ στοιχείων και ιδιοτήτων

Δεν υπάρχει κανένας ευδιάκριτος τρόπος για να αποφασίσει κάποιος πότε είναι καταλληλότερη η χρήση των στοιχείων (elements) ή των ιδιοτήτων (attributes) σε ένα αρχείο DTD ή XML Schema. Και τα στοιχεία και οι ιδιότητες μπορούν να χρησιμοποιηθούν για την αναπαράσταση της πληροφορίας. Πολλές φορές η επιλογή μεταξύ ενός στοιχείου και μιας ιδιότητας γίνεται πολύ αυθαίρετα, και είναι σχεδόν θέμα στιλ. Οι κύριοι παράγοντες απόφασης είναι η προσωπική προτίμηση του καθενός και εάν το σύνολο εργαλείων που χρησιμοποιείται παρέχει την υποστήριξη για τις ιδιότητες.

Ενώ η επιλογή μπορεί πράγματι να είναι αυθαίρετη σε μερικές περιπτώσεις, οι «χαρακτηριστικοί» ρόλοι των στοιχείων και των ιδιοτήτων και ο σκοπός για τον οποίο έχει σχεδιαστεί ένα μοντέλο περιεχομένου εγγράφου, καθώς και οι περιορισμοί που περιέχουν αυτά τα δύο μπορεί να θέσουν την προτίμηση του χρήστη προς κάποια κατεύθυνση. Παρακάτω αναφέρονται τα πλεονεκτήματα και τα μειονεκτήματα των στοιχείων και των ιδιοτήτων, ώστε να είναι δυνατό να αποφασιστεί σε ποιες περιπτώσεις είναι κατάλληλη η χρήση του καθενός.

Ένας τρόπος για να διαχωριστεί η χρήση των στοιχείων και των ιδιοτήτων είναι τα στοιχεία να περιέχουν τα «πραγματικά» δεδομένα, ενώ οι ιδιότητες να λειτουργούν ως υπόμνημα για τα στοιχεία προσθέτοντας τους πρόσθετες πληροφορίες και περιγράφοντας το περιεχόμενο τους. Στην περίπτωση των κενών στοιχείων οι ιδιότητες παρέχουν τις επιπρόσθετες πληροφορίες και εξηγούν τον λόγο για τον οποίο αυτά είναι παρόν. Στα στοιχεία που δεν είναι κενά, οι ιδιότητες μπορούν ενδεχομένως να χρησιμοποιηθούν για να περιγράψουν το περιεχόμενο που αντιπροσωπεύουν τα στοιχεία. Έτσι η παραδοσιακή πρακτική σύνταξης είναι να τίθεται το πραγματικό κείμενο (αυτό που θα μπορούσε να τυπωθεί ή να διαβαστεί) ως το περιεχόμενο των δεδομένων του κειμένου, και να διατηρηθούν τα μεταδεδομένα (πληροφορίες σχετικές με το κείμενο) στις ιδιότητες, από όπου μπορούν να απομονωθούν ευκολότερα για την ανάλυση ή την εξειδικευμένη επεξεργασία τους.

Οι «εγγενής» διαφορές μεταξύ των στοιχείων και των ιδιοτήτων στην XML 1.0 τείνουν να καθορίσουν τα όρια για το που μπορεί να χρησιμοποιηθεί το καθένα. Η πιο θεμελιώδης διαφορά είναι ότι τα στοιχεία μπορούν να περιέχουν στοιχεία παιδιά (child elements), καθώς επίσης και το ότι το περιεχόμενο τους δεν περιορίζεται μόνο στο κείμενο όπως συμβαίνει με τις ιδιότητες. Εάν είναι όλο και πιο πιθανό ότι θα πρέπει να διασπαστεί η πληροφορία που αποθηκεύεται, τότε θα πρέπει η πληροφορία να περιέχεται σε στοιχεία.

Αλλά από την πλευρά των συστημάτων υπολογιστών, δεν υπάρχει τίποτα λανθασμένο με την αποθήκευση των δεδομένων σε ιδιότητες και το αντίθετο, ειδικά όπου ο όγκος των δεδομένων του κειμένου είναι κατά περίπτωση σχετικά μικρός:

```
<line speaker="Portia" text="The quality of mercy is not
strain">184</line>
```

Πολλά εξαρτώνται από το τι είναι επιθυμητό να γίνει με τις πληροφορίες και ποια τμήματα από αυτές προσεγγίζονται ευκολότερα με κάθε μέθοδο. Μια εμπειροτεχνική μέθοδος για τα συμβατικά έγγραφα κειμένων είναι ότι εάν η σύνταξη σήμανσης απομακρυνόταν, το κείμενο που θα έμενε θα έπρεπε ακόμα να ήταν αναγνώσιμο και χρησιμοποιήσιμο, ακόμα κι αν ήταν ανομοιόμορφο και δύσχρηστο. Για την παραγωγή βάσεων δεδομένων, όμως, ή άλλα machine-generated έγγραφα όπως οι συναλλαγές ηλεκτρονικού εμπορίου, η ανθρώπινη ανάγνωση μπορεί να μην είναι σημαντική, και έτσι να είναι δυνατόν να υπάρξουν έγγραφα όπου όλα τα δεδομένα να αποθηκεύονται σε ιδιότητες, και το έγγραφο να μην περιέχει καθόλου κείμενο στο μοντέλο περιεχομένου του.

3.5.1 Πλεονεκτήματα της χρήσης των στοιχείων

Η χρήση των στοιχείων για την περιγραφή του περιεχομένου είναι πιο σαφής από αυτή των ιδιοτήτων. Ένα έγγραφο με στοιχεία είναι γενικά ευκολότερο να διαβάσει από ένα

έγγραφο με πολλές ιδιότητες. Τα στοιχεία υποστηρίζουν την παρεμβολή μεγάλων τμημάτων κειμένων μεταξύ τους, ενώ μπορούν να περιλαμβάνονται κενά διαστήματα και κενές γραμμές στο κείμενο, ενώ στις ιδιότητες αυτό δεν είναι δυνατό. Η χρήση των στοιχείων πρέπει να γίνεται όταν:

- Τα δεδομένα μπορεί να είναι πολύ εκτενή. Το XML έγγραφο δε θα ήταν εύκολα αναγνώσιμο εάν τα δεδομένα εμφανίζονταν σε μια ιδιότητα.
- Για την αποθήκευση σχολίων που χρησιμοποιούνται για τη διευκόλυνση των χρηστών, όπως ένα τμήμα κειμένου HTML, ένα μήνυμα λάθους, κ.λπ.
- Αναπαριστώνται πίνακες. Κατά την αναπαράσταση των πινάκων, δημιουργείται ένα νέο στοιχείο. Το στοιχείο μπορεί να εμφανιστεί πολλές φορές.
- Μια τιμή έχει τη δομή όπως σε μια γλώσσα προγραμματισμού - δημιουργία ενός νέου στοιχείου που περιέχει ιδιότητες.
- Η σειρά είναι σημαντική - δημιουργία νέων στοιχείων στα οποία μπορεί να επιβληθεί η σειρά τοποθέτησης τους.
- Υπάρχει η αίσθηση ότι μια απλή αλφαριθμητική σειρά θα πρέπει να «διασπασθεί» σε πολλαπλά τμήματα σε κάποιο χρονικό σημείο στη συνέχεια
- Χρειάζεται να γίνει η μοντελοποίηση των δεδομένων ώστε να επανα-χρησιμοποιηθούν μέσα σε ένα έγγραφο ή σε διάφορα έγγραφα

Η χρησιμοποίηση των στοιχείων για την αποθήκευση του περιεχομένου και ο περιορισμός των ιδιοτήτων ως σχολιασμός σε αυτά έχουν όμως μερικά μειονεκτήματα. Η σήμανση των στοιχείων με τις ετικέτες αρχής και τέλους είναι πιο εκτενή από ότι η σήμανση των ιδιοτήτων με τη χρήση ενός ονόματος και μερικών εισαγωγικών. Τα στοιχεία-παιδιά μπορούν να παρέχουν περισσότερη ευελιξία, αλλά αυτή η ευελιξία δεν είναι πάντα απαραίτητη. Σε μερικές καταστάσεις, όπως στην ανταλλαγή μεγάλων ποσοτήτων μικρών τμημάτων πληροφορίας μεταξύ βάσεων δεδομένων, οι ιδιότητες μπορεί να είναι αποδοτικότερες για την αναπαράσταση της πληροφορίας.

3.5.2 Πλεονεκτήματα της χρήσης των ιδιοτήτων

Οι ιδιότητες πρέπει να χρησιμοποιηθούν κυρίως για τις ιδιότητες ID και IDREF. Τα ID και IDREF παρέχουν χρήσιμη λειτουργία, επειδή ένα εργαλείο επικύρωσης DTD μπορεί να ελέγξει την ισχύ των αναφορών παραπομπής ID (αυτή η λειτουργία δεν είναι διαθέσιμη στα στοιχεία).

Η χρησιμοποίηση των ιδιοτήτων έχει ως αποτέλεσμα να παράγονται μικρότερα αρχεία. Είναι επίσης ευκολότερο να διαβαστούν οι ιδιότητες με τη χρήση του DOM. Πρέπει να γίνεται χρήση των ιδιοτήτων όταν είναι δυνατό, και χρήση των στοιχείων όταν αυτό απαιτείται. Η χρήση των ιδιοτήτων πρέπει να γίνεται όταν:

- Παρουσιάζεται μια μονότιμη ιδιότητα σε ένα αντικείμενο - ένα όνομα, μια ημερομηνία, κ.λπ.
- Η ιδιότητα έχει έναν απλό τύπο δεδομένων (datatype) και καμία εσωτερική δομή.
- Μια διεύθυνση, παραδείγματος χάριν, πρέπει να διαιρεθεί σε οδό, πόλη, κράτος, κωδικό περιοχής. Υπάρχουν τόσα πολλά στοιχεία για μια διεύθυνση που θα ήταν πρακτικότερο η διεύθυνση να διαμορφωθεί απλά ως ένα στοιχείο και τα υπόλοιπα στοιχεία της να παρέχονται ως ιδιότητες.

  ```
  <Address city="San Mateo" State="CA" zip="94401" street1="123 Ellsworth" street2="Apt. 13"/>
  ```

- Το μέγεθος της αποθήκευσης είναι σημαντικό. Δεδομένου ότι οι ιδιότητες περιέχουν μόνο κείμενο, απαιτούν πολύ λιγότερο «βάρος δεδομένων» (overhead) από αυτό των στοιχείων τα οποία επιτρέπουν την αναπαράσταση κενών γραμμών και διαστημάτων.

3.6 Σύγκριση μεταξύ των τεχνολογιών μοντελοποίησης DTD και XMLSchema

Τα XML Schemas, όπως αναφέρθηκε και παραπάνω, παρέχουν μια αντικειμενοστρεφή προσέγγιση για τον καθορισμό της δομής ενός XML εγγράφου και με αυτά μπορούν επίσης να καθοριστούν επανα-χρησιμοποιήσιμοι τύποι δεδομένων (όπως για παράδειγμα ένας αριθμός ISBN) με βάση ένα ευρύ φάσμα προκαθορισμένων τύπων που υπάρχουν στα Schemas. Αν και τα XML Schemas υπερτερούν των DTDs υπάρχουν στην πραγματικότητα αρκετοί λόγοι για τους οποίους προτιμάται η χρήση των DTDs αντί των XML Schemas για τη μοντελοποίηση του περιεχομένου των εγγράφων XML.

Ένας από τους πιο σημαντικούς λόγους είναι ότι το XML Schema είναι μια νέα τεχνολογία. Αυτό έχει ως αποτέλεσμα να υπάρχουν περισσότερα προγράμματα επεξεργασίας των XML εγγράφων (parsers) που υποστηρίζουν το DTD. Επίσης οι οργανισμοί που χρησιμοποιούν για μεγάλο χρονικό διάστημα το διαδίκτυο για τη μεταφορά αρχείων μέσα από αυτό μπορεί να έχουν αναπτύξει τη μοντελοποίηση των εγγράφων τους με βάση το DTD.

Το DTD είναι ένα καθιερωμένο πρότυπο μοντελοποίησης XML εγγράφων, και υπαρχουν πολλά παραδείγματα XML εγγράφων στο διαδίκτυο που είναι ελεύθερα διαθέσιμα για επαναχρησιμοποίηση και η δομή τους περιγράφεται από ένα DTD. Αυτά τα αρχεία DTD μπορούν να χρησιμοποιηθούν από κάθε χρήστη για να δημιουργήσει με βάση αυτά τα δικά του αρχεία DTD γρηγορότερα από ότι εάν επιχειρούσε να αναπτύξει από την αρχή ένα ολόκληρο DTD.

Τέλος, είναι σημαντικό να αναφερθεί ότι το XML Schema είναι και αυτό ένα έγγραφο XML. Περιέχει ένα XML Namespace που αναφέρεται σε αυτό το έγγραφο και ένα Schema που καθορίζει τη δομή του. Αυτά τα στοιχεία είναι πρόσθετη πληροφορία (overhead) για το XML έγγραφο. Όταν ένας αναλυτής (parser) εξετάζει ένα XML έγγραφο που η δομή του περιγράφεται με ένα XML Schema, τότε πρέπει να συνδέσει όλα τα παραπάνω στροιχεία σε ένα έγγραφο, να ερμηνεύσει το Schema, να φορτώσει το namespace, και να επικυρώσει το Schema, και όλα αυτά προτού ακόμα αναλύσει το πραγματικό XML έγγραφο. Εάν χρησιμοποιείται η XML ως πρωτόκολλο επικοινωνίας μεταξύ δύο συστημάτων, η χρήση των οποίων είναι συχνή, και είναι απαραίτητη η γρήγορη επεξεργασία των δεδομένων, τότε όλη αυτή η επιπλέον πληροφορία μπορεί να υποβαθμίσει σημαντικά την απόδοση αυτών των συστημάτων.

Επιπλέον με τα XML Schemas μπορούν να αναπτυχθούν κοινά λεξιλόγια, με αποτέλεσμα να επιτρέπεται στις μηχανές να πραγματοποιήσουν διάφορους κανόνες που καθορίζονται από τους χρήστες. Παρέχουν τρόπους για τον προσδιορισμό της δομής, του περιεχομένου και της σημασιολογίας των XML εγγράφων. Εν τέλει, τα XML Schemas είναι πλουσιότερα και ισχυρότερα για την περιγραφή και τη μοντελοποίηση των πληροφοριών από ότι είναι τα DTDs.

3.7 Ασκήσεις

Άσκηση 1 (Θέμα 3-δ εξετάσεων Φεβρουαρίου 2002)

Να καθοριστεί νέος τύπος δεδομένων σε ένα XMLSchema, ο οποίος να δέχεται τις ακέραιες τιμές από το –100 έως το 500

Λύση

Το XML Schema παρέχει τη δυνατότητα δημιουργίας νέων τύπων δεδομένων. Το γεγονός αυτό κάνει τα XML Schemas πιο ευέλικτα από τα DTD. Οι τύποι δεδομένων των στοιχείων που ορίζονται σε ένα XML Schema είναι δύο ειδών, simpleType και complexType.

Οι τιμές των στοιχείων τύπου simpleType είναι απλά strings που δεν περιέχουν ενθετημένα άλλα στοιχεία, αλλά μπορούν να περιοριστούν ώστε να περιέχουν αριθμούς ή κάποια άλλη συγκεκριμένη μορφή.

```
<element name = "FirstName" type = "string">
```

Τα στοιχεία τύπου complexType μπορούν να περιέχουν ενθετημένα και άλλα στοιχεία ή να περιέχουν attributes.

```
<element name = "Customer">
<complexType>
   <sequence>
      <element name = "FirstName" type = "string"/>
      <element name = "LastName" type = "string"/>
   </sequence>
</complexType>
</element>
```

Προκειμένου να δημιουργήσουμε έναν τύπο δεδομένων για να περιορίσουμε το εύρος των τιμών το οποίο θα εισάγεται (συγκεκριμένα ακεραίους αριθμούς από -100 ως 500) θα πρέπει να ορίσουμε ένα simpleType στοιχείο.

```
<simpleType name = "new_dt">
   <restriction base = "integer">
      <minInclusive value = "-100"/>
      <maxInclusive value = "500"/>
   <restriction>
</simpleType>
```

Το στοιχείο restriction χρησιμοποιείται για να περιοριστεί το simpleType με τη χρήση περιοριστικών στοιχείων. Συγκεκριμένα η τιμή "integer" της ιδιότητας base του στοιχείου restriction καθορίζει ότι ο νέος τύπος δεδομένων θα δέχεται μόνο ακεραίους αριθμούς και τα ενθετημένα στοιχεία minInclusive και maxInclusive, μέσω της ιδιότητας value, καθορίζουν το ελάχιστο και το μέγιστο ακέραιο που θα δέχεται το στοιχείο αυτό. Στη συνέχεια παρατίθεται ένα πλήρες XML Schema, όπου ορίζεται ο παραπάνω τύπος δεδομένων new_dt και ένα στοιχείο number όπου είναι τύπου new_dt.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:simpleType name="new_dt">
      <xs:restriction base="xs:integer">
         <xs:minInclusive value="-100"/>
         <xs:maxInclusive value="500"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:element name="number" type="new_dt"/>
</xs:schema>
```

Άσκηση 2

Με βάση το δενδρικό διάγραμμα που αναπτύχθηκε στο υποερώτημα (α) της Άσκησης 2 του Κεφαλαίου 2:

(α) Να καθοριστεί DTD που περιγράφει πλήρως την δενδρική δομή

(β) Να δοθεί παράδειγμα XML αρχείου με βάση το καθορισμένο DTD

(γ) Να δοθεί το XMLSchema της παραπάνω δομής

Λύση

(α) Το DTD που περιγράφει την δενδρική δομή είναι:

```
<!ELEMENT Μάθημα (Τίτλος_Μαθήματος, Κωδικός_Μαθήματος,
                    Πανεπιστήμιο,Τμήμα,Καθηγητής,Έτος,Εξάμηνο,Βιβλίο)>
<!ELEMENT Τίτλος_Μαθήματος (#PCDATA)>
<!ELEMENT Κωδικός_Μαθήματος (#PCDATA)>
<!ELEMENT Πανεπιστήμιο (#PCDATA)>
<!ELEMENT Τμήμα (#PCDATA)>
<!ELEMENT Καθηγητής (Ονοματεπώνυμο,email)+>
<!ELEMENT Έτος (#PCDATA)>
<!ELEMENT Εξάμηνο (#PCDATA)>
<!ELEMENT Βιβλίο (Τίτλος,Συγγραφέας,Εκδότης,Έτος_Έκδοσης)>
<!ELEMENT Ονοματεπώνυμο (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT Τίτλος  (#PCDATA)>
<!ELEMENT Συγγραφέας (#PCDATA)>
<!ELEMENT Εκδότης (#PCDATA)>
<!ELEMENT Έτος_Έκδοσης (#PCDATA)>
```

Έστω ότι το dtd αυτό αποθηκεύεται τοπικά με το όνομα Lesson.dtd

(β) Ακολουθεί ένα παράδειγμα ενός XML εγγράφου το οποίο βασίζεται στο προηγούμενο DTD. Το XML αρχείο θεωρείται ότι είναι αποθηκευμένο στην ίδια θέση με το dtd αρχείο.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Μάθημα SYSTEM "Lesson.dtd">
<Μάθημα>
      <Τίτλος_Μαθήματος>ΑΠΟΘΗΚΕΣ ΚΑΙ ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ</Τίτλος_Μαθήματος>
      <Κωδικός_Μαθήματος>ΤΕ256</Κωδικός_Μαθήματος>
      <Πανεπιστήμιο>Πανεπιστήμιο Πειραιώς</Πανεπιστήμιο>
      <Τμήμα>Διδακτική της Τεχνολογίας και Ψηφιακών Συστημάτων</Τμήμα>
      <Καθηγητής>
            <Ονοματεπώνυμο>Δρ. Δημήτριος Σάμψων</Ονοματεπώνυμο>
            <email>sampson@iti.gr</email>
```

```
      </Καθηγητής>
      <Έτος>3</Έτος>
      <Εξάμηνο>5</Εξάμηνο>
      <Βιβλίο>
            <Τίτλος>Οδηγός της XML με παραδείγματα</Τίτλος>
            <Συγγραφέας>Benoit Marchal</Συγγραφέας>
            <Εκδότης>Β. Γκιούρδας Εκδοτική</Εκδότης>
            <Έτος_Έκδοσης>2000</Έτος_Έκδοσης>
      </Βιβλίο>
</Μάθημα>
```

(γ) Ακολουθεί το αντίστοιχο XML Schema που αναπαριστά την παραπάνω δενδρική δομή

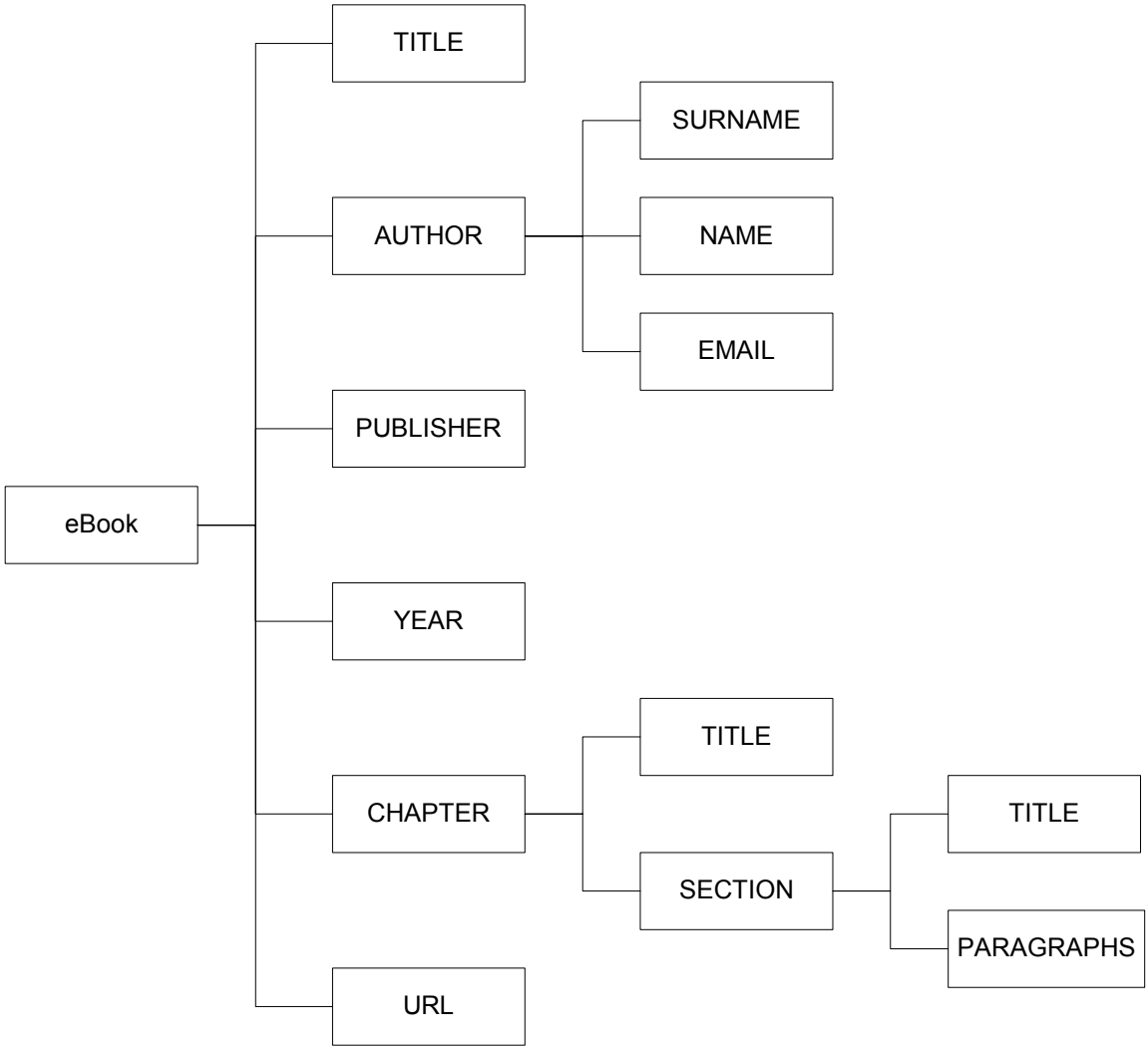
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="Τίτλος_Μαθήματος" type="xs:string"/>
      <xs:element name="Κωδικός_Μαθήματος" type="xs:string"/>
      <xs:element name="Πανεπιστήμιο" type="xs:string"/>
      <xs:element name="Τμήμα" type="xs:string"/>
      <xs:element name="Έτος" type="xs:string"/>
      <xs:element name="Εξάμηνο" type="xs:string"/>
      <xs:element name="Καθηγητής">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="Ονοματεπώνυμο" type="xs:string"
maxOccurs=”unbounded”/>
                        <xs:element name="email" type=”xs:string”
maxOccurs=”unbounded”/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="Βιβλίο">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="Τίτλος" type="xs:string"/>
                        <xs:element name="Συγγραφέας" type="xs:string"/>
                        <xs:element name="Εκδότης" type="xs:string"/>
                        <xs:element name="Έτος_Έκδοσης" type="xs:string"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="Μάθημα">
```

```
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="Τίτλος_Μαθήματος"/>
                        <xs:element ref="Κωδικός_Μαθήματος"/>
                        <xs:element ref="Πανεπιστήμιο"/>
                        <xs:element ref="Τμήμα"/>
                        <xs:element ref="Καθηγητής"/>
                        <xs:element ref="Έτος"/>
                        <xs:element ref="Εξάμηνο"/>
                        <xs:element ref="Βιβλίο"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Προκειμένου να ορίσουμε ένα στοιχείο που έχει ενθετημένα άλλα στοιχεία χρησιμοποιούμε την ίδια δομή που χρησιμοποιήσαμε για να ορίσουμε τα στοιχεία "Καθηγητής" "Βιβλίο" "Μάθημα". Με το στοιχείο sequence, δηλώνεται ότι τα στοιχεία που καθορίζονται μέσα στο στοιχείο αυτό πρέπει να εμφανίζονται με την ίδια σειρά μέσα στα XML έγγραφα. Ο ελάχιστος και μέγιστος αριθμός εμφάνισης του κάθε στοιχείου καθορίζεται με τις ιδιότητες minOccurs και maxOccurs αντίστοιχα. Η προκαθορισμένη (default) τιμή των ιδιοτήτων αυτών είναι η τιμή "1". Σε περίπτωση δηλαδή που δεν δηλώνονται οι ιδιότητες αυτές κατά τη δήλωση ενός στοιχείο σημαίνει ότι το στοιχείο αυτό πρέπει να εμφανίζεται υποχρεωτικά μία φορά. Η τιμή "unbounded" της ιδιότητας maxOccurs χρησιμοποιείται όταν δεν θέλουμε να ορίσουμε ένα άνω όριο εμφάνισης κάποιου στοιχείου. Τέλος η ιδιότητα ref του στοιχείου element χρησιμοποιείται για να κάνει αναφορά σε ένα ήδη δηλωμένο στοιχείο.

Άσκηση 3 (Θέμα 3 εξετάσεων Σεπτεμβρίου 2002)

Η δενδρική αναπαράσταση η οποία περιγράφει πλήρως ηλεκτρονικά βιβλία που υπάρχουν δημοσιευμένα στο Internet, φαίνεται σχηματικά παρακάτω:



(α) Καθορίστε το XML Schema που καθορίζει την παραπάνω δομή για περιγραφή των ηλεκτρονικών βιβλίων. Να περιγραφούν όλες οι παραδοχές που έγιναν για τον τύπο των δεδομένων καθώς και για τη συχνότητα εμφάνισης του κάθε πεδίου.

(β) Να δοθεί παράδειγμα XML αρχείου το οποίο να περιγράφει ένα ηλεκτρονικό βιβλίο με βάση το καθορισμένο XML Schema.

Λύση

(α)Όσον αφορά το τύπο δεδομένων, για λόγους απλότητας θεωρούμε ότι ο τύπος κάθε στοιχείου είναι string. Όσον αφορά τη συχνότητα εμφάνισης κάθε στοιχείου θεωρούμε ότι όλα τα πεδία είναι υποχρεωτικά. Όσον αφορά τα πεδία AUTHOR, CHAPTER, SECTION και URL θεωρούμε ότι εμφανίζονται στα XML έγγραφα πάνω από μια φορά. Για παράδειγμα ένα βιβλίο μπορεί να έχει πολλούς συγγραφείς, να είναι διαθέσιμο από πολλές

ηλεκτρονικές διευθύνσεις, αποτελείται από πολλά κεφάλαια, όπου κάθε ένα κεφάλαιο αποτελείται από πολλά τμήματα (sections).

Συνεπώς το XML Schema που υλοποιεί την παραπάνω δενδρική αναπαράσταση, θα είναι ως εξής:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="TITLE" type="xs:string"/>
	<xs:element name="PUBLISHER" type="xs:string"/>
	<xs:element name="YEAR" type="xs:string"/>
	<xs:element name="URL" type="xs:string"/>
	<xs:element name="AUTHOR">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="SURNAME" type="xs:string"/>
				<xs:element name="NAME" type="xs:string"/>
				<xs:element name="EMAIL" type="xs:string"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="SECTION">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="TITLE"/>
				<xs:element name="PARAGRAPHS"
maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="CHAPTER">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="TITLE"/>
				<xs:element ref="SECTION" maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="eBook">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="TITLE"/>
```

```
                        <xs:element ref="AUTHOR" maxOccurs="unbounded"/>
                        <xs:element ref="PUBLISHER"/>
                        <xs:element ref="YEAR"/>
                        <xs:element ref="CHAPTER" maxOccurs="unbounded"/>
                        <xs:element ref="URL" maxOccurs="unbounded"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Υποθέτουμε ότι το Schema αυτό αποθηκεύεται τοπικά σε ένα αρχείο με όνομα Exercise.xsd.

(β) Ακολουθεί ένα παράδειγμα XML εγγράφου που θα χρησιμοποιηθεί για την περιγραφή ενός ηλεκτρονικού βιβλίου και θα βασίζεται στο παραπάνω XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<eBook xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Exercise.xsd">
      <TITLE>MASTERING XML</TITLE>
      <AUTHOR>
            <SURNAME>White</SURNAME>
            <NAME>Chuck</NAME>
            <EMAIL>chuck@tumeric.net</EMAIL>
      </AUTHOR>
      <AUTHOR>
            <SURNAME>Wall</SURNAME>
            <NAME>David</NAME>
            <EMAIL>david@davidwall.com</EMAIL>
      </AUTHOR>
      <PUBLISHER>SYBEX</PUBLISHER>
      <YEAR>2001</YEAR>
      <CHAPTER>
            <TITLE>Creating XML Documents</TITLE>
            <SECTION>
                  <TITLE>Creating an XML Document</TITLE>
                  <PARAGRAPHS>4</PARAGRAPHS>
            </SECTION>
            <SECTION>
                  <TITLE>Two Kinds of legal XML</TITLE>
                  <PARAGRAPHS>3</PARAGRAPHS>
            </SECTION>
      </CHAPTER>
```

```
      <CHAPTER>
            <TITLE>An Introduction to XML Schemas</TITLE>
            <SECTION>
                  <TITLE>What is an XML Schema</TITLE>
                  <PARAGRAPHS>2</PARAGRAPHS>
            </SECTION>
            <SECTION>
                  <TITLE>When to use XML Schemas</TITLE>
                  <PARAGRAPHS>4</PARAGRAPHS>
            </SECTION>
      </CHAPTER>
      <URL>www.sybex.com</URL>
      <URL>www.amazon.com</URL>
</eBook>
```

Με την ιδιότητα noNamespaceSchemaLocation (χρησιμοποιείται το πρόθεμα xsi που αντιστοιχίζεται στο URI http://www.w3.org/2001/XMLSchema-instance) δηλώνεται σε κάποιον XML αναλυτή πού μπορεί να αναζητήσει το XML Schema και ότι το λεξικό που καθορίζεται από το schema, δεν ανήκει σε κάποιον χώρο ονοματοδοσίας. Η τιμή της ιδιότητας αυτής περιγράφει τη θέση του XML Schema. Το Schema αυτό είναι αποθηκευμένο στην ίδια θέση με το XML έγγραφο και για το λόγο αυτό δεν περιγράφεται κάποιο πλήρες path που να δείχνει που είναι αποθηκευμένο.

ΣΗΜΕΙΩΣΗ

Σε περίπτωση όπου θα θέλαμε να ορίσουμε ένα XML Schema στο οποίο να δηλώνεται ένας χώρος ονοματοδοσίας (έστω "http://www.mynamespaces.gr/xsd") όπου θα ανήκει το λεξικό που θα καθορίζεται από το XML Schema, το root element του XML Schema θα έχει την εξής μορφή

```
<xs:schema targetNamespace=" http://www.mynamespaces.gr/xsd"
xmlns="http://www.mynamespaces.gr/xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

και το αντίστοιχο root element ενός XML εγγράφου όπου θα χρησιμοποιεί ως μοντέλο δομής το παραπάνω schema, θα έχει την εξής μορφή

```
<eBook xmlns="http://www.mynamespaces.gr/xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mynamespaces.gr/xsd Exercise.xsd">
```

Άσκηση 4

Με βάση το δενδρικό διάγραμμα που αναπτύχθηκε στο υποερώτημα (α) της Άσκησης 4 του Κεφαλαίου 2:

(α) Να σχεδιαστεί το DTD που υλοποιεί την δενδρική αναπαράσταση και ένα παράδειγμα XML εγγράφου.

Λύση

Με βάση τις παραδοχές που έγιναν για την τον τύπο δεδομένων των πεδίων (Άσκηση 4, Κεφάλαιο 2), παρατίθενται δύο ίδιες εκδόσεις του ίδιου DTD. Στο πρώτο (Exercise5a.dtd) τα πεδία Κωδικός_Χρήστη και Ταξινομικός_Αριθμός αναπαρίστανται σαν στοιχεία και στο δεύτερο (Exercise5b.dtd) ως ιδιότητες.

```
<!ELEMENT Δελτίο (Βιβλίο,Χρήστης)>
<!ELEMENT Βιβλίο (Ταξινομικός_Αριθμός,Συγγραφέας+,Τίτλος,Ημερομηνία)>
<!ELEMENT Χρήστης (Κωδικός_Χρήστη)>
<!ELEMENT Ταξινομικός_Αριθμός (#PCDATA)>
<!ELEMENT Συγγραφέας (#PCDATA)>
<!ELEMENT Τίτλος (#PCDATA)>
<!ELEMENT Ημερομηνία (#PCDATA)>
<!ELEMENT Κωδικός_Χρήστη (#PCDATA)>
```

Exercise5a.dtd

```
<!ELEMENT Δελτίο (Βιβλίο,Χρήστης)>
<!ELEMENT Βιβλίο (Συγγραφέας+,Τίτλος,Ημερομηνία)>
<!ATTLIST Βιβλίο Ταξινομικός_Αριθμός ID #REQUIRED>
<!ELEMENT Συγγραφέας (#PCDATA)>
<!ELEMENT Τίτλος (#PCDATA)>
<!ELEMENT Ημερομηνία (#PCDATA)>
<!ELEMENT Χρήστης EMPTY>
<!ATTLIST Χρήστης  Κωδικός_Χρήστη ID #REQUIRED>
```

Exercise5b.dtd

Θα πρέπει να σημειωθεί ότι στο δεύτερο DTD, οι ιδιότητες `Ταξινομικός_Αριθμός` και `Κωδικός_Χρήστη` ορίστηκαν ως τύπου ID, δηλαδή ως ιδιότητες των οποίων οι τιμές είναι μοναδικά καθορισμένες.

Ένα παράδειγμα XML εγγράφου που θα βασίζεται στο πρώτο DTD (Exercise5a.dtd) θα έχει την εξής μορφή:

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

```
<!DOCTYPE Δελτίο SYSTEM "Exercise5a.dtd">
<Δελτίο>
      <Βιβλίο>
            <Ταξινομικός_Αριθμός>Π278</Ταξινομικός_Αριθμός>
            <Συγγραφέας>Micheline Kamber</Συγγραφέας>
            <Συγγραφέας>Jiawei Han</Συγγραφέας>
            <Τίτλος>Data Mining: Concepts and Techniques</Τίτλος>
            <Ημερομηνία>2000</Ημερομηνία>
      </Βιβλίο>
      <Χρήστης>
            <Κωδικός_Χρήστη>Ε00023</Κωδικός_Χρήστη>
      </Χρήστης>
</Δελτίο>
```

Ένα παράδειγμα ενός XML εγγράφου που θα βασίζεται στο δεύτερο DTD (Exercise5b.dtd) θα έχει την εξής μορφή:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Δελτίο SYSTEM "Exercise5b.dtd">
<Δελτίο>
 <Βιβλίο Ταξινομικός_Αριθμός="Π278">
      <Συγγραφέας>Micheline Kamber</Συγγραφέας>
      <Συγγραφέας>Jiawei Han</Συγγραφέας>
      <Τίτλος>Data Mining: Concepts and Techniques</Τίτλος>
      <Ημερομηνία>2000</Ημερομηνία>
 </Βιβλίο>
 <Χρήστης Κωδικός_Χρήστη="Ε00023"/>
</Δελτίο>
```

Στο τελευταίο παράδειγμα αυτό φαίνεται πως τα πεδία Ταξινομικός_Αριθμός και Κωδικός_Χρήστη πέρασαν σαν ιδιότητες στα στοιχεία Βιβλίο και Χρήστης αντίστοιχα.

4 XML Χώροι Ονοματοδοσίας

Μια από τις δυσκολίες που ανακύπτουν κατά την εργασία με XML έγγραφα που ανακτώνται από πολλαπλές πηγές, είναι ότι η έννοια των ετικετών της κάθε εφαρμογής, μπορεί να γίνει πολλές φορές ασαφής. Για παράδειγμα, αν μια χρηματιστηριακή εφαρμογή, συγχώνευε δύο XML έγγραφα, ένα από μια Καναδέζική τράπεζα και ένα από μια Αμερικάνικη τράπεζα σε ένα ενιαίο έγγραφο, κατά την επεξεργασία της ετικέτας <dollars>, θα ήταν δύσκολο να προσδιορίσει αν το περιεχόμενο αυτού του στοιχείου αναφέρεται σε Καναδέζικα ή σε Αμερικάνικα Δολάρια. Το πρόβλημα αυτό επιλύουν οι XML χώροι ονοματοδοσίας, όπου επιτρέπουν στους χρήστες να εξαλείψουν την ασάφεια της έννοιας των XML στοιχείων και ιδιοτήτων. Οι XML χώροι ονοματοδοσίας επιτρέπουν στους χρήστες να προσδιορίσουν ονόματα στοιχείων και ιδιοτήτων, συσχετίζοντας τα με χώρους ονοματοδοσίας που συνδέονται με URI αναφορές. Οι χώροι ονοματοδοσίας ορίζονται στην σύσταση "Namespaces in XML" (REC-xml-namespaces-1999-01-14) της W3C.

4.1 Τι είναι ο χώρος ονοματοδοσίας

Σε γενικές γραμμές οι χώροι ονοματοδοσίας μπορούν να περιγραφτούν ως εξής:

Η κατά W3C σύσταση χώρων ονοματοδοσίας (W3C XML Namespaces Recommendation) καθορίζει έναν τρόπο για την διάκριση διπλών ονομάτων στοιχείων και χαρακτηριστικών.

Ένας χώρος ονοματοδοσίας στην XML είναι μια συλλογή από ονόματα στοιχείων και χαρακτηριστικών. Ο χώρος ονοματοδοσίας αντιστοιχεί σε ένα μοναδικό όνομα, που είναι ένα URI. Κατά συνέπεια, ένα όνομα στοιχείου ή ιδιότητας σε ένα XML χώρο ονοματοδοσίας μπορεί να ταυτιστεί μοναδικά από ένα όνομα αποτελούμενο από δυο μέρη:Το όνομα του χώρου ονοματοδοσίας και του τοπικού του ονόματος(local name). Αυτού του είδους συστήματος ονοματοδοσίας είναι το μόνο που καθορίζεται από την σύσταση των χώρων ονοματοδοσίας.

Οι χώροι ονοματοδοσίας στην XML δηλώνονται με τη χρήση της ιδιότητας xmlns, μέσω της οποίας συσχετίζεται ένα πρόθεμα με ένα χώρο ονοματοδοσίας. Η δήλωση είναι εντός εμβέλειας για το στοιχείο που περιέχει την ιδιότητα και όλους τους απογόνους του. Για παράδειγμα:

```
<!—Εδώ δηλώνονται δύο XML χώροι ονοματοδοσίας. Στην εμβέλεια τους περιέχονται τα στοιχεία A και B-->
<A xmlns:foo="http://www.foo.org/" xmlns="http://www.bar.org/">
     <B>abcd</B>
  </A>
```

Εάν μια δήλωση ενός χώρου ονοματοδοσίας περιέχει ένα πρόθεμα, η αναφορά στα ονόματα των στοιχείων και των ιδιοτήτων που ανήκουν στο χώρο αυτό γίνεται με τη χρήση του προθέματος αυτούαναφέρεται κάποιος στα ονόματα των στοιχείων και των χαρακτηριστικών σε αυτόν τον χώρο ονοματοδοσίας με το πρόθεμα αυτό.

```
<!-- Τα στοιχεία A και B περιέχονται στο χώρο ονοματοδοσίας http://www.foo.org και η αναφορά σε αυτά γίνεται μέσω του προθέματος foo που είναι συσχετισμένο με αυτά -->
<foo:A xmlns:foo="http://www.foo.org/">
     <foo:B>abcd</foo:B>
```

```
</foo:A>
```

Εάν μια δήλωση ενός χώρου ονοματοδοσίας δεν περιέχει πρόθεμα, τότε ο χώρος ονοματοδοσίας είναι ο προκαθορισμένος χώρος ονοματοδοσίας και δεν απαιτείται η χρήση προθέματος για να γίνει η αναφορά στα στοιχεία που περιέχονται στο χώρο ονοματοδοσίας. Για παράδειγμα:

```
<!—Το παράδειγμα είναι ίδιο με το προηγούμενο με τη διαφορά ότι δηλώνεται ένας προκαθορισμένος χώρος ονοματοδοσίας -->
<A xmlns="http://www.foo.org/">
      <B>abcd<B>
</A>
```

4.2 Προβλήματα που επιλύουν οι χώροι ονοματοδοσίας

Η XML παρέχει τη δομή για την περιγραφή των δεδομένων μέσα σε μια εφαρμογή, καθώς και το μέσο για τη μεταφορά της πληροφορίας μεταξύ των εφαρμογω. Οποτεδήποτε χρησιμοποιούνται αυτές οι πολλαπλές πήγες πληροφοριών, υπάρχει η πιθανότητα ονοματολογικής σύγκρουσης. Η ονοματολογική σύγκρουσης συμβαίνει όταν δύο κομμάτια πληροφοριών, τα οποία έχουν διαφορετικούς τύπους ή διαφορετική σημασιολογική έννοια αναφέρονται με τα ίδια ονόματα στοιχείων ή ιδιοτήτων.

Τα προβλήματα στα οποία απευθύνονται οι χώροι ονοματοδοσίας αφορούν κυρίως την ονοματολογική σύγκρουση, την ασάφεια δηλαδή που δημιουργείταόταν δύο κομμάτια πληροφοριών, τα οποία έχουν διαφορετικούς τύπους ή διαφορετική σημασιολογική έννοια αναφέρονται με τα ίδια ονόματα στοιχείων ή ιδιοτήτων. Η ονοματολογική σύγκρουση μπορεί να προκύψει για διάφορους λόγους, όπως για παράδειγμα στην περίπτωση όπου τμήματα XML εγγράφων συγχωνεεύονται σε ένα. Για παράδειγμα, ονοματολογική σύγκρουση προκύπτει κατά τη συνένωση των εξής δύο XML εγγράφων:

```
<?xml version="1.0" ?>
<Address>
     <Street>Wilhelminenstr. 7</Street>
     <City>Darmstadt</City>
     <State>Hessen</State>
     <Country>Germany</Country>
     <PostalCode>D-64285</PostalCode>
</Address>
```

και:

```
<?xml version="1.0" ?>
<Server>
     <Name>OurWebServer</Name>
     <Address>123.45.67.8</Address>
</Server>
```

Στο πρώτο έγγραφο το στοιχείο `Address` χρησιμοποιείται για την περιγραφή της φυσικής διεύθυνσης ενός προσώπου, και περιέχει ενθετημένα διάφορα στοιχεία, όπως για παράδειγμα το όνομα της οδού και της πόλης που διανέμει το πρόσωπο αυτό. Αντίθετα, στο δεύτερο XML έγγραφο, το στοιχείο `Address` χρησιμοποιείται για την περιγραφή της IP διεύθυνσης ενός εξυπηρετητή. Είναι φανερό ότι η ονοματολογική σύγκρουση εμφανίζεται, αφού συγχωνευθούν τα δύο αυτά έγγραφα σε ένα και κατά την επεξεργασία του στοιχείου `Address`.

Ένα άλλο πρόβλημα στο οποίο απευθύνονται οι χώροι ονοματοδοσίας είναι η ανάγκη για κοινούς, συμφωνημένους ορισμούς για τους όρους λεξιλογίου που μπορούν να

χρησιμοποιηθούν από δικτυακές εφαρμογές κυρίως σε τομείς της βιομηχανίας. Το πρόβλημα αυτό, είναι παρόμοιο με αυτό ονοματολογικών συγκρούσεων που είναι το αποτέλεσμα από συγχωνευμένα δεδομένα. Παραδείγματος χάριν, σκεφτείτε τις ερωτήσεις, που ένας παράγοντας της αγοράς πρέπει να λάβει υπόψη του όταν συγκρίνει τις τιμές των ειδών. Πρώτον, ποια είναι η ακριβής έννοια του όρου "price'' όταν αυτή βρίσκεται σε ένα έγγραφο; Δεύτερον, εννοεί το ποσό των χρημάτων που πρέπει να υποβληθεί, προκειμένου να αγοραστεί ένα προϊόν, ή ίσως κάτι ελαφρώς διαφορετικό όπως για παράδειγμα αν περιλαμβάνεται στη τιμή ο φόρος. Τρίτον, εάν μια εφαρμογή βρίσκει μια τιμή των $100, κωδικοποιημένη ως <dollars>100</dollars>, αναφέρεται αυτή η τιμή στα δολάρια των ΗΠΑ, ή ίσως στα Καναδικά δολάρια;

Ο διαμοιρασμός της πληροφορίας και η επαναχρησιμοποίηση δεδομένων θα αποτελεί μια κενή εργασίας στο Διαδίκτυο του μέλλοντος, και η XML θα διαδραματίσει έναν κεντρικό ρόλο για να γίνει αυτό. Σε αυτόν τον μελλοντικό κόσμο, οι παραγωγοί δεδομένων και οι δημιουργοί των εφαρμογών πρέπει να εξασφαλίσουν ότι η έννοια των δεδομένων τους (π.χ., τι αντιπροσωπεύουν οι ετικέτες) είναι πάντα σαφής, και τα XML namespaces παρέχουν στους χρήστες ακριβώς αυτήν την δυνατότητα. Όπως σημειώσαμε ήδη, οι τομείς της βιομηχανίας θα τυποποιήσουν πιθανώς τα XML λεξιλόγια, και τέτοια τυποποιημένα λεξιλόγια θα επιτρέψουν στους συνεργάτες των επιχειρήσεων να εκτελέσουν ακριβείς συναλλαγές. Παραδείγματος χάριν, χρησιμοποιώντας ένα namespace το οποίο αναφέρεται σε ένα κατάλληλο URI (όπως xmlns:usa=http://AmericaTheBeautiful.org), $100 Αμερικάνικα δολάρια θα μπορούσαν να εκφραστούν ως <usa:dollars>100</usa:dollars>.

4.3 Χρησιμοποιώντας τους χώρους ονοματοδοσίας στην XML

4.3.1 Δηλώνοντας τους χώρους ονοματοδοσίας

Οι χώροι ονοματοδοσίας δηλώνονται στην XML με τους εξής δύο τρόπους:

```
<πρόθεμα:όνομα_στοιχείου xmlns:πρόθεμα= "URI">
```

και

```
<όνομα_στοιχείου xmlns= "URI">
```

Με τον πρώτο τρόπο χρησιμοποιείται η ιδιότητα `xmlns:πρόθεμα` που συσχετίζει ένα πρόθεμα με το χώρο ονοματοδοσίας που αντιστοιχεί στο URI που δηλώνεται ως τιμή στην ιδιότητα αυτή. Η δεύτερη μορφή (xmlns) δηλώνει ότι ο χώρος ονοματοδοσίας που καθορίζεται, είναι ο προκαθορισμένος (default) χώρος ονοματοδοσίας. Ένα παράδειγμα δήλωσης ενός χώρου ονοματοδοσίας με χρήση προθέματος είναι το εξής:

```
<?xml version="1.0"?>
<addr:Address xmlns:addr="http://www.tu-darmstadt.de/ito/addresses">
        <addr:Street>Wilhelminenstr. 7</addr:Street>
        <addr:City>Darmstadt</addr:City>
        <addr:State>Hessen</addr:State>
        <addr:Country>Germany</addr:Country>
        <addr:PostalCode>D-64285</addr:PostalCode>
</addr:Address>
```

Στο παράδειγμα αυτό, χρησιμοποιείται το πρόθεμα addr, για να προσδιοριστεί ο χώρος ονοματοδοσίας που περιγράφεται από το URI (στη συγκεκριμένη περίπτωση είναι μια URL διεύθυνση) "`http://www.tu-darmstadt.de/ito/addresses`". Αυτό το είδος της

δήλωσης χρησιμοποιείται για να προσδιορίζει και τα στοιχεία και τις ιδιότητες των XML εγγράφων. Το ίδιο XML έγγραφο, στο οποίο ο χώρος ονοματοδοσίας ορίζεται ως προκαθορισμένος, θα έχει την εξής μορφή:

```
<?xml version="1.0"?>
<Address xmlns="http://www.tu-darmstadt.de/ito/addresses">
      <Street>Wilhelminenstr. 7</Street>
      <City>Darmstadt</City>
      <State>Hessen</State>
      <Country>Germany</Country>
      <PostalCode>D-64285</PostalCode>
</Address>
```

Μέσω της προκαθορισμένης δήλωσης αποφεύγεται η χρήση του ίδιου προθέματος σε όλα τα στοιχεία του εγγράφου. Ωστόσο, η προκαθορισμένη δήλωση των χώρων ονοματοδοσίας επιτρέπει τον προσδιορισμό μόνο των στοιχείων ενός XML εγγράφου και όχι των ιδιοτήτων του.

Χρήση προκαθορισμένων χώρων ονοματοδοσίας έναντι χρήσης προθεμάτων

Το πότε θα χρησιμοποιηθεί ο προκαθορισμένος χώρος ονοματοδοσίας αντί των προθεμάτων είναι καθαρά θέμα επιλογής, αν και η επιλογή αυτή μπορεί να επηρεάσει την αναγνωσιμότητα του εγγράφου. Όταν στοιχεία των οποίων όλα τα ονόματα ανήκουν σε ένα μοναδικό XML χώρο ονοματοδοσίας και είναι ομαδοποιημένα, η χρήση ενός προκαθορισμένου χώρου ονοματοδοσίας κάνει το έγγραφο πιο ευανάγνωστο. Για παράδειγμα:

```
<!-- Τα στοιχεία A, B, C, και G περιέχονται στον χώρο ονοματοδοσίας που περιγράφεται από το εξής URI, http://www.foo.org/. -->
<A xmlns="http://www.foo.org/">
      <B>abcd</B>
      <C>efgh</C>
<!-- Τα στοιχεία D, E και F περιέχονται στο χώρο ονοματοδοσίας που περιγράφεται από το ακόλουθο. URI http://www.bar.org/. -->
      <D xmlns="http://www.bar.org/">
            <E>1234</E>
            <F>5678</F>
      </D>
<!-- Το στοιχείο G περιέχεται στο χώρο ονοματοδοσίας http://www.foo.org/ -->
            <G>ijkl</G>
</A>
      <A xmlns="http://www.foo.org/">
      <B xmlns="http://www.bar.org/">abcd</B>
      <C xmlns="http://www.foo.org/">efgh</C>
      <D xmlns="http://www.bar.org/">
            <E xmlns="http://www.foo.org/">1234</E>
            <F xmlns="http://www.bar.org/">5678</F>
```

```
      </D>
      <G xmlns="http://www.foo.org/">ijkl</G>
</A>
```

Σε μερικές περιπτώσεις, οι προκαθορισμένοι χώροι ονοματοδοσίας μπορούν να επεξεργαστούν πιο γρήγορα απ' ότι τα προθέματα χώρων ονοματοδοσίας, αλλά η διαφορά είναι αμελητέα σε σύγκριση με τον συνολικό χρόνο επεξεργασίας.

4.3.2 Εμβέλεια των χώρων ονοματοδοσίας

Ένας χώρος ονοματοδοσίας μπορεί να δηλωθεί σε οποιοδήποτε στοιχείο σε ένα XML έγγραφο. Τα στοιχεία που είναι ενθετημένα σε αυτά που δηλώνεται ο χώρος ονοματοδοσίας είναι εντός της εμβέλειας του χώρου αυτού, εκτός και αν ο χώρος ονοματοδοσίας έχει αντικατασταθεί ή έχει καταργηθεί. Για παράδειγμα, στο ακόλουθο τμήμα ενός XML εγγράφου, η εμβέλεια της δήλωσης του χώρου ονοματοδοσίας http://www.foo.org/ περιλαμβάνει το στοιχείο A και τους απογόνους του (στοιχεία B και C). Η εμβέλεια της δήλωσης του χώρου ονοματοδοσίας http://www.bar.org/ περιλαμβάνει μόνο το στοιχείο C.

```
<foo:A xmlns:foo="http://www.foo.org/">
      <foo:B>
            <bar:C xmlns:bar="http://www.bar.org/" />
      </foo:B>
</foo:A>
```

Η εμβέλεια ενός χώρου ονοματοδοσίας περιλαμβάνει το στοιχείο στο οποίο έχει δηλωθεί. Για παράδειγμα, στο ακόλουθο τμήμα ενός XML εγγράφου, τα ονόματα B και C είναι στον χώρο ονοματοδοσίας http://www.bar.org/ και όχι στον http://www.foo.org/ .Αυτό ισχύει γιατί η δήλωση που σχετίζει το πρόθεμα foo με τον χώρο ονοματοδοσίας http://www.bar.org/ συμβαίνει στο στοιχείο B αντικαθιστώντας(overriding) την δήλωση στο στοιχείο A που το σχετίζει με τον χώρο ονοματοδοσίας http://www.foo.org/ .

```
<foo:A xmlns:foo="http://www.foo.org/">
      <foo:B xmlns:foo="http://www.bar.org/">
            <foo:C>abcd</foo:C>
      </foo:B>
</foo:A>
```

Ομοίως, στο ακόλουθο παράδειγμα, τα ονόματα B και C ανήκουν στον http://www.bar.org/ χώρο ονοματοδοσίας και όχι στον http://www.foo.org/ γιατί η δήλωση του http://www.bar.org/ σαν προκαθορισμένο χώρο ονοματοδοσίας , αντικαθιστά την δήλωση στο στοιχείο A.

```
<A xmlns="http://www.foo.org/">
      <B xmlns="http://www.bar.org/">
            <C>abcd</C>
      </B>
</A>
```

Ένα τελικό παράδειγμα, είναι αυτό στο οποίο το όνομα της ιδιότητας D ανήκει στον χώρο ονοματοδοσίας http://www.bar.org/ .

```
<foo:A xmlns:foo="http://www.foo.org/">
      <foo:B foo:D="http://www.bar.org/"
```

```
                    xmlns:foo="http://www.bar.org/">
            <C>abcd</C>
      </foo:B>
</foo:A>
```

Εάν κάποιο στοιχείο ή μια ιδιότητα είναι στην εμβέλεια μιας δήλωσης χώρου ονοματοδοσίας, το όνομά του δεν ανήκει απαραίτητα σε αυτό το χώρο ονοματοδοσίας. Όταν κάποιο στοιχείο ή μια ιδιότητα είναι στην εμβέλεια μιας δήλωσης χώρου ονοματοδοσίας, το όνομα του στοιχείου ή της ιδιότητας ελέγχεται για να διαπιστωθεί εάν έχει κάποιο πρόθεμα που ταιριάζει με το πρόθεμα στην δήλωση. Το κατά πόσο το όνομα είναι στον χώρο ονοματοδοσίας εξαρτάται από το κατά πόσο το πρόθεμα ταιριάζει. Για παράδειγμα, στο ακόλουθο, τα A, B και D ονόματα τύπου στοιχείων και τα C και E ονόματα χαρακτηριστικών είναι στην εμβέλεια της δήλωσης του χώρου ονοματοδοσίας που αντιστοιχεί στο URI http://www.foo.org/. Ενώ, τα ονόματα A,B και C είναι σ' αυτόν τον χώρο ονοματοδοσίας, τα ονόματα D και E δεν είναι.

```
<foo:A xmlns:foo="http://www.foo.org/">
      <foo:B foo:C="foo"/>
      <bar:D bar:E="bar"/>
</foo:A>
```

Όταν μια δήλωση ενός χώρου ονοματοδοσίας τοποθετείται εκτός εμβέλειας, παύει να ισχύει. Για παράδειγμα, στο ακόλουθο έγγραφο, η δήλωση του χώρου ονοματοδοσίας που αντιστοιχεί στο URI http://www.foo.org/, δεν ισχύει στο στοιχείο C, γιατί αυτό είναι εκτός της εμβέλειας του. Αυτό σημαίνει ότι ισχύει το τέλος του B στοιχείου, στο οποίο έχει δηλωθεί ο χώρος ονοματοδοσίας που αντιστοιχεί στο URI http://www.foo.org/.

```
<!--το B είναι στον http://www.foo.org/ χώρο ονοματοδοσίας, ενώ το C δεν
περιέχεται σε κανένα χώρο ονοματοδοσίας -->
<A>
      <B xmlns="http://www.foo.org/">abcd</B>
      <C>efgh</C>
</A>
```

Κατά τον τερματισμό της ισχύος ενός χώρου ονοματοδοσίας, επανέρχεται εντός εμβελείας ο χωρός ονοματοδοσίας που είχε αντικατασταθεί. Για παράδειγμα, στο ακόλουθο XML έγγραφο, ο χώρος ονοματοδοσίας που αντιστοιχεί στο URI http://www.foo.org/, βρίσκεται πάλι εντός εμβέλειας μετά το τέλος του στοιχείου B. Αυτό γίνεται γιατί είχε αντικατασταθεί στο B στοιχείο από την δήλωση του χώρου ονοματοδοσίας που αντιστοιχεί στο URI http://www.bar.org/.

```
<!—Τα στοιχεία A και C περιέχονται στο χώρο ονοματοδοσίας που περιγράφεται
από το URI http://www.foo.org/ namespace. Το στοιχείο B ανήκει στο χώρο
ονοματοδοσίας που περιγράφεται απ΄ το URI http://www.bar.org/ namespace. --
>
<A xmlns="http://www.foo.org/">
      <B xmlns="http://www.bar.org/">abcd</B>
      <C>efgh</C>
</A>
```

Πολλές δηλώσεις χώρων ονοματοδοσίας μπορούν να είναι εντός εμβέλειας την ίδια στιγμή, εφόσον δεν χρησιμοποιούν τα ίδια προθέματα και τουλάχιστον ένας έχει οριστεί ως ο προκαθορισμένος χώρος ονοματοδοσίας. Για παράδειγμα, στο ακόλουθο παράδειγμα,

οι χώροι ονοματοδοσίας http://www.foo.org/ και http://www.bar.org/, είναι εντός εμβέλειας για όλα τα στοιχεία.

```
<A xmlns:foo="http://www.foo.org/"
      xmlns:bar="http://www.bar.org\
      foo:B>abcd</foo:B>
      <bar:C>efgh</bar:C>
</A>
```

Ένα επακόλουθο αυτού είναι ότι μπορούν να τοποθετηθούν όλοι οι χώροι ονοματοδοσίας στο στοιχείο ρίζα και έτσι θα είναι όλοι εντός εμβέλειας για όλα τα στοιχεία. Αυτός είναι ο απλούστερος τρόπος να χρησιμοποιηθούν οι χώροι ονοματοδοσίας.

Πολλαπλές δηλώσεις χώρων ονοματοδοσίας

Πολλές δηλώσεις χώρων ονοματοδοσίας μπορούν να είναι εντός εμβέλειας την ίδια στιγμή, εφόσον δεν χρησιμοποιούν τα ίδια προθέματα και τουλάχιστον ένας έχει οριστεί ως ο προκαθορισμένος χώρος ονοματοδοσίας. Για παράδειγμα, στο ακόλουθο, ο http://www.foo.org/ και ο http://www.bar.org/ χώροι ονοματοδοσίας είναι εντός εμβέλειας για όλα τα στοιχεία.

```
<A xmlns:foo="http://www.foo.org/" xmlns:bar="http://www.bar.org/">
      <foo:B>abcd</foo:B>
      <bar:C>efgh</bar:C>
</A>
```

Ένα επακόλουθο αυτού είναι ότι μπορούν να τοποθετηθούν όλοι οι χώροι ονοματοδοσίας στο στοιχείο ρίζα και έτσι θα είναι όλοι εντός εμβέλειας για όλα τα στοιχεία. Αυτός είναι ο απλούστερος τρόπος να χρησιμοποιηθούν οι χώροι ονοματοδοσίας. ο επόμενο παράδειγμα

```
<Department
         xmlns:addr="http://www.tu-darmstadt.de/ito/addresses"
         xmlns:serv="http://www.tu-darmstadt.de/ito/servers">
      <Name>DVS1</Name>
      <addr:Address>
                  <addr:Street>Wilhelminenstr. 7</addr:Street>
                  <addr:City>Darmstadt</addr:City>
                  <addr:State>Hessen</addr:State>
                  <addr:Country>Germany</addr:Country>
                  <addr:PostalCode>D-64285</addr:PostalCode>
            </addr:Address>
            <serv:Server>
                  <serv:Name>OurWebServer</serv:Name>
                  <serv:Address>123.45.67.8</serv:Address>
            </serv:Server>
</Department>
```

4.3.3 Μέθοδοι αντικατάστασης – κατάργησης του χώρου ονοματοδοσίας

Αντικατάσταση του χώρου ονοματοδοσίας που αντιστοιχεί σε ένα συγκεκριμένο πρόθεμα

Για να αντικαταστήσει κάποιος το πρόθεμα που χρησιμοποιείται σε μια δήλωση χώρου ονοματοδοσίας, απλά πρέπει να δηλώσει έναν άλλο χώρο ονοματοδοσίας με το ίδιο πρόθεμα. Για παράδειγμα, στο XML έγγραφο που ακολουθεί, το πρόθεμα foo σχετίζεται με τον http://www.foo.org/ χώρο ονοματοδοσίας και προσδιορίζει τα στοιχεία A και B καθώς και με τον http://www.bar.org/ χώρο ονοματοδοσίας, όπου προσδιορίζει τα στοιχεία C και D. Αυτό σημαίνει ότι τα A και B ανήκουν στον http://www.foo.org/ χώρο ονοματοδοσίας και τα C και D ανήκουν στον http://www.bar.org/ χώρο ονοματοδοσίας.

```
<foo:A xmlns:foo="http://www.foo.org/">
      <foo:B>
            <foo:C xmlns:foo="http://www.bar.org/">
                  <foo:D>abcd</foo:D>
            </foo:C>
      </foo:B>
</foo:A>
```

Αντικατάσταση ενός προκαθορισμένου χώρου ονοματοδοσίας

Προκειμένου να αντικαταστήσει μια νέα δήλωση την τρέχουσα προκαθορισμένη δήλωση, θα πρέπει να δηλωθεί ως η προκαθορισμένη. Για παράδειγμα, στο XML έγγραφο που ακολουθεί, ο προκαθορισμένος χώρος ονοματοδοσίας για τα στοιχεία A και B είναι ο http://www.foo.org/ και για τα στοιχεία C και D ο http://www.bar.org/ . Αυτό σημαίνει ότι τα A και B είναι στον http://www.foo.org/ χώρο ονοματοδοσίας και τα C και D στον http://www.bar.org/ :

```
<A xmlns="http://www.foo.org/">
      <B>
            <C xmlns="http://www.bar.org/">
                  <D>abcd</D>
            </C>
      </B>
</A>
```

Κατάργηση του προθέματος ενός χώρου ονοματοδοσίας

Το πρόθεμα ενός XML χώρου ονοματοδοσίας δεν είναι δυνατόν να καταργηθεί. Παραμένει εντός εμβέλειας ως το τέλος του στοιχείου στο οποίο δηλώθηκε εκτός και αν έχει αντικατασταθεί. Η προσπάθεια κατάργησης ενός προθέματος συσχετίζοντας το με ένα χώρο ονοματοδοσίας του οποίου το όνομα είναι κενό οδηγεί σε συντακτικά λάθη. Για παράδειγμα:

```
<foo:A xmlns:foo="http://www.foo.org/">
      <foo:B>
            <foo:C xmlns:foo="">   <!-- ==== Αυτό είναι λάθος == -->.
                  <foo:D>abcd</foo:D>
```

```
            </foo:C>
      </foo:B>
</foo:A>
```

Κατάργηση ενός προκαθορισμένου χώρου ονοματοδοσίας

Για να καταργηθεί ένας προκαθορισμένος χώρος ονοματοδοσίας, αρκεί να δηλωθεί ένας προκαθορισμένος χώρος ονοματοδοσίας με κενό όνομα. Εντός της εμβέλειας αυτής της δήλωσης, τα ονόματα των στοιχείων που δεν περιέχουν πρόθεμα δεν ανήκουν σε κανένα χώρο ονοματοδοσίας. Για παράδειγμα, στο XML έγγραφο που ακολουθεί, ο προκαθορισμένος χώρος ονοματοδοσίας για τα A και B είναι ο http://www.foo.org/ και δεν υπάρχει προκαθορισμένος χώρος ονοματοδοσίας για τα C και D. Συνεπώς τα A και B ανήκουν στον http://www.foo.org/ και τα C,D δεν ανήκουν σε κανένα χώρο ονοματοδοσίας:

```
<A xmlns="http://www.foo.org/">
      <B>
            <C xmlns="">
                  <D>abcd</D>
            </C>
      </B>
</A>
```

4.3.4 Χώροι ονοματοδοσίας και DTD

Προκειμένου ένα έγγραφο στο οποίο έχει δηλωθεί ένας χώρος ονοματοδοσίας να είναι έγκυρο ως προς ένα DTD που περιγράφει τη δομή του, θα πρέπει στο ίδιο το DTD να καθορίζεται επ' ακριβώς το όνομα των στοιχείων καθώς και των ιδιοτήτων που απαιτούνται για τον καθορισμό του χώρου ονοματοδοσίας. Στο επόμενο XML έγγραφο χρησιμοποιείται ένα εσωτερικό DTD για να καθορίζονται τα ονόματα των ιδιοτήτων και των στοιχείων όπως επίσης και τα URI's που περιγράφουν ένα συγκεκριμένο χώρο ονοματοδοσίας

```
<?xml version="1.0"?>
<!DOCTYPE f:A
[
      <!ELEMENT f:A(f:B)>
      <!ATTLIST f:A xmlns:f CDATA #FIXED "http://www.f.org">
      <!ATTLIST f:A f:C CDATA #IMPLIED>
      <!ELEMENT f:B (#PCDATA)>
]>
<f:A xmlns:f= "http://www.f.org" f:C= "value">
  <f:B>abcd</f:B>
</f:A>
```

Το DTD ενός έγκυρου XML έγγραφου, που δηλώνεται ένας προκαθορισμένος χώρος ονοματοδοσίας θα έχει την εξής μορφή

```
<?xml version="1.0"?>
<!DOCTYPE A
```

```
[
      <!ELEMENT A(B)>
      <!ATTLIST A xmlns CDATA #FIXED "http://www.f.org">
      <!ELEMENT B (#PCDATA)>
]>
<A xmlns= "http://www.f.org">
  <B>abcd</B>
</A>
```

Το επόμενο XML έγγραφο δεν είναι έγκυρο, καθότι τα στοιχείο `A` δεν έχει οριστεί στο DTD

```
<?xml version="1.0"?>
<!DOCTYPE foo:A
[
      <!ELEMENT foo:A (#PCDATA)>
      <!ATTLIST foo:A
      xmlns:foo CDATA #FIXED "http://www.foo.org/"
      xmlns CDATA #FIXED "http://www.foo.org/"
]>
<A xmlns=http://www.foo.org/>abcd</A>
```

4.4 Παραδείγματα XML Χώρων Ονοματοδοσίας

Παράδειγμα 1

Το υποθετικό αυτό παράδειγμα αναφέρεται στα αποτελέσματα της αναζήτησης στο website www.amazon.com για την εύρεση cd, dvd και βιβλίων που περιέχουν το όνομα Notting Hill.
Ορίζονται τρείς χώροι ονοματοδοσίας για να προσδιοριστούν οι ετικέτες `title`, `price`, και `rating` έτσι ώστε να εξαλειφθεί η προφανής ασάφεια.

```
<?xml version="1.0" encoding="UTF-8"?>
<search xmlns:cd="http://www.amazon.com/cd/def"
xmlns:dvd="http://www.amazon.com/dvd/def"
xmlns:book="http://www.amazon.com/book/def">
```

Αποτέλεσμα της Αναζήτηση του Στοιχείου: Notting Hill

```
</search>
  <cd:result>
    <cd:title>Nottimg Hill</cd:title>
    <cd:price>$12</cd:price>
    <cd:rating>3</cd:rating>
  </cd:result>
  <dvd:result>
    <dvd:title>Nottimg Hill</dvd:title>
    <dvd:price>$24</dvd:price>
    <dvd:rating>5</dvd:rating>
  </dvd:result>
```

```
   <book:result>
      <book:title>Nottimg Hill</book:title>
      <book:price>$8</book:price>
      <book:rating>4</book:rating>
   </book:result>
</search>
```

Παράδειγμα 2

Ένα έγγραφο MathML που περιέχει τους πρώτους τρεις αριθμούς Fibonacci.

```
<?xml version="1.0" encoding="UTF-8"?>
<mathml:math xmlns:mathml="http://www.w3.org/1998/Math/MathML">
   <mathml:mrow>
      <mathml:mi>f(1)</mathml:mi>
      <mathml:mo>=</mathml:mo>
      <mathml:mn>1</mathml:mn>
   </mathml:mrow>
   <mathml:mrow>
      <mathml:mi>f(2)</mathml:mi>
      <mathml:mo>=</mathml:mo>
      <mathml:mn>1</mathml:mn>
   </mathml:mrow>
   <mathml:mrow>
      <mathml:mi>f(3)</mathml:mi>
      <mathml:mo>=</mathml:mo>
      <mathml:mn>2</mathml:mn>
   </mathml:mrow>
</mathml:math>
```

Παράδειγμα 3

Ένα έγγραφο XHML στο οποίο οι χώροι ονοματοδοσίας καθορίζονται στο στοιχείο ρίζας του εγγράφου:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:xlink="http://www.w3.org/1999/xlink">
   <head>
      <title>Three Namespaces</title>
   </head>
   <body>
      <h1 align="center">An Ellipse and a Rectangle</h1>
      <svg xmlns="http://www.w3.org/2000/svg" width="12cm" eight="10cm">
         <ellipse rx="110" ry="130"/>
         <rect x="4cm" y="1cm" width="3cm" height="6cm"/>
      </svg>
      <p xlink:type="simple" xlink:href="ellipses.html">More about
ellipses</p>
      <p xlink:type="simple" xlink:href="γ">More about rectangles</p>
      <hr/>
```

```
    <p>Last Modified May 13, 2000</p>
  </body>
</html>
```

4.5 Ασκήσεις

Άσκηση 1

Με βάση το δενδρικό διάγραμμα που αναπτύχθηκε στην άσκηση 4 του Κεφαλαίου 2, να αναπτυχθούν δυο έγγραφα XML Schemas. Το πρώτο, θα περιγράφει πληροφορία που αφορά το δανειζόμενο βιβλίο, και το δεύτερο, θα περιγράφει πληροφορία που αφορά το χρήστη. Συγκεκριμένα, τον κωδικό, όνομα, επώνυμο, email, διεύθυνση κατοικίας. Να δοθεί ένα παράδειγμα ενός XML αρχείου όπου θα συνδυάζει, μέσω των χώρων ονοματοδοσίας, τα λεξικά που παρέχονται από τα δύο XML Schemas για να περιγράψει τη φόρμα δανεισμού βιβλίων, όπως αυτή συμπληρώθηκε από ένα συγκεκριμένο χρήστη.

Λύση

Ακολουθούν δύο εκδόσεις του XML Schema που περιγράφει το δανειζόμενο βιβλίο. Στην πρώτη έκδοση τα πεδία `Κωδικός_Χρήστη` και `Ταξινομικός_Αριθμός` αναπαρίστανται σαν στοιχεία και στο δεύτερο ως ιδιότητες των οποίων οι τιμές είναι μοναδικά ορισμένες.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.abook.gr/xsd" xmlns="http://www.abook.gr/xsd">
      <xs:element name="Βιβλίο">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="Ταξινομικός_Αριθμός"
type="xs:string"/>
                        <xs:element name="Συγγραφέας" type="xs:string"
maxOccurs="unbounded"/>
                        <xs:element name="Τίτλος" type="xs:string"/>
                        <xs:element name="Ημερομηνία" type="xs:string"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Exercise5a.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.abook.gr/xsd" xmlns="http://www.abook.gr/xsd">
      <xs:element name="Βιβλίο">
            <xs:complexType>
                  <xs:sequence>
```

```
                        <xs:element name="Συγγραφέας" type="xs:string"
maxOccurs="unbounded"/>
                        <xs:element name="Τίτλος" type="xs:string"/>
                        <xs:element name="Ημερομηνία" type="xs:string"/>
                  </xs:sequence>
                  <xs:attribute name="Ταξινομικός_Αριθμός" type="xs:ID"
use="required"/>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Exercise5b.xsd

Στο έγγραφο Exercise5b.xsd το πεδίο `Ταξινομικός_Αριθμός` έχει εισαχθεί σαν attribute στο πεδίο Βιβλίο και είναι τύπου `ID`. Η υποχρεωτική χρήση της ιδιότητας `Ταξινομικός_Αριθμός` καθορίζεται από την τιμή required της ιδιότητας use. Και στις δυο περιπτώσεις ορίζεται ένας χώρος ονοματοδοσίας (http://www.abook.gr/xsd) όπου θα ανήκει το λεξικό όπως αυτό ορίζεται από τα XML Schemas.

Με την ίδια λογική ακολουθούν δύο εκδόσεις του XML Schema, που αφορά την περιγραφή του χρήστη

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.user.gr/xsd" xmlns="http://www.user.gr/xsd">
      <xs:element name="Χρήστης">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="Κωδικός_Χρήστη"
                        type="xs:string"/>
                        <xs:element name="Όνομα" type="xs:string"/>
                        <xs:element name="Επώνυμο" type="xs:string"/>
                        <xs:element name="Διεύθυνση" type="xs:string"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Exercise5_1.xml

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.user.gr/xsd" xmlns="http://www.user.gr/xsd">
      <xs:element name="Χρήστης">
            <xs:complexType>
```

```
                  <xs:sequence>
                        <xs:element name="Όνομα" type="xs:string"/>
                        <xs:element name="Επώνυμο" type="xs:string"/>
                        <xs:element name="Διεύθυνση" type="xs:string"/>
                  </xs:sequence>
                  <xs:attribute name="Κωδικός_Χρήστη" type="xs:ID"
                  use="required"/>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Exercise5_2.xml

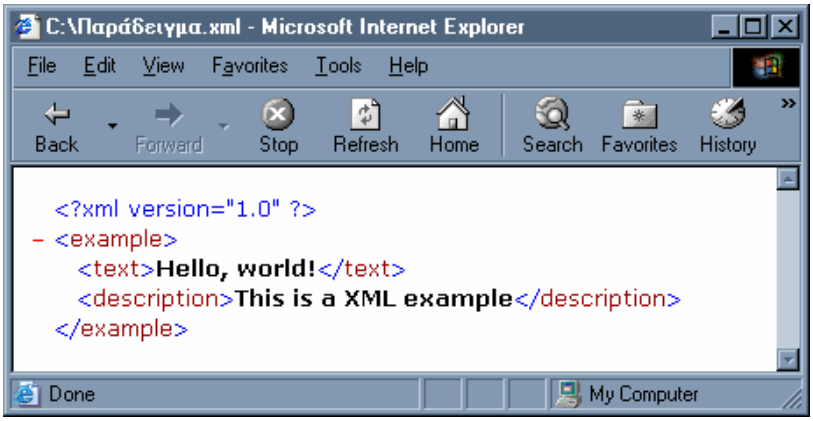
Ένα παράδειγμα XML εγγράφου που θα περιγράφει τη πληροφορία του δελτίου κάνοντας χρήση των λεξικών, μέσω των χώρων ονοματοδοσίας, όπως αυτά ορίστηκαν από τα XML Schemas Exercise5a.xsd και Exercise5_1.xsd, θα έχει την εξής μορφή:

```
<?xml version="1.0" encoding="UTF-8"?>
<δελτίο>
      <bo:Βιβλίο xmlns:bo="http://www.abook.gr/xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.abook.gr/xsd Exercise5a.xsd" >
            <bo:Ταξινομικός_Αριθμός>Π278</bo:Ταξινομικός_Αριθμός>
            <bo:Συγγραφέας>Micheline Kamber</bo:Συγγραφέας>
            <bo:Συγγραφέας>Jiawei Han</bo:Συγγραφέας>
            <bo:Τίτλος>Data Mining: Concepts and Techniques</bo:Τίτλος>
            <bo:Ημερομηνία>2000</bo:Ημερομηνία>
            </bo:Βιβλίο>
      <us:Χρήστης xmlns:us="http://www.user.gr/xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.abook.gr/xsd Exercise5_1.xsd">
            <us:Κωδικός_Χρήστη>Π377</us:Κωδικός_Χρήστη>
            <us:Όνομα>Κώστας</us:Όνομα>
            <us:Επώνυμο>Καστραντάς</us:Επώνυμο>
            <us:Διεύθυνση>Χείρωνος 5</us:Διεύθυνση>
      </us:Χρήστης>
</δελτίο>
```

5 Μετασχηματισμός XML εγγράφων με χρήση της τεχνολογίας XSLT

5.1 Εισαγωγή στο μετασχηματισμό εγγράφων

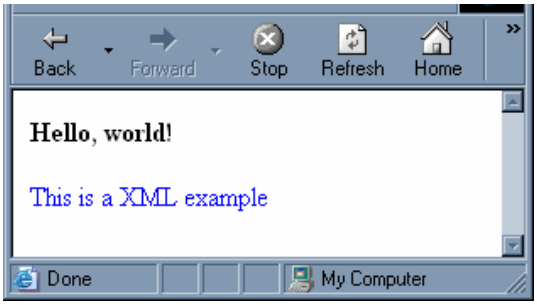
Η μορφοποίηση ενός XML εγγράφου καθορίζει τον τρόπο με τον οποίο παρουσιάζεται το περιεχόμενο του εγγράφου. Στην εικόνα που ακολουθεί απεικονίζεται ο τρόπος με τον οποίο εμφανίζονται τα XML έγγραφα στον φυλλομετρητή Microsoft Internet Explorer



Εικόνα 5-1 Αναπαράσταση XML εγγράφου

Προκειμένου να μορφοποιηθεί το XML έγγραφο έτσι ώστε μόνο το περιεχόμενό του να είναι ορατό από τον χρήστη και να αποκρύπτονται οι ετικέτες, τότε γίνεται χρήση των μετασχηματισμών. Ο μετασχηματισμός ενός XML εγγράφου είναι μια διαδικασία κατά την οποία ανασυντάσσονται τμήματα του εγγράφου ώστε να δημιουργηθεί έγγραφο διαφορετικής μορφής. Για παράδειγμα τα XML έγγραφα τα οποία δεν είναι αναγνώσιμα από έναν παλαιό περιηγητή που αναγνωρίζει μόνο έγγραφα τύπου HTML, είναι δυνατόν να μετασχηματιστούν με απλές διαδικασίες σε HTML. Το περιεχόμενο του εγγράφου παραμένει το ίδιο ενώ αλλάζουν μόνο οι ετικέτες σήμανσης. Οι μετασχηματισμοί μπορούν επίσης να χρησιμοποιηθούν για να «φιλτράνουν» ένα XML έγγραφο και να διατηρήσουν μόνο ένα τμήμα του αρχικού.

Τα XML έγγραφα μετασχηματίζονται με τη βοήθεια της γλώσσας μετασχηματισμού XSLT (eXtensible Stylesheet Language for Transformations). Η γλώσσα μετασχηματισμού συντάσσεται σε ένα ξεχωριστό έγγραφο το οποίο αποτελεί το φύλο στυλ (stylesheet) και αυτό με τη σειρά του εφαρμόζεται στο έγγραφο XML μέσω μιας μηχανής μετασχηματισμού. Η εικόνα που ακολουθεί απεικονίζει το αποτέλεσμα του μετασχηματισμού .



Εικόνα 5-2 Αποτέλεσμα μετασχηματισμού

Η XSLT γλώσσα μετασχηματισμού χρησιμοποιείται για τη διαχείριση XML εγγράφων και πιο συγκεκριμένα επιτρέπει:

- Την προσθήκη ειδικών συστατικών για την εμφάνιση, όπως η προσθήκη του λογότυπου ή της διεύθυνσης του αποστολέα σε ένα τιμολόγιο XML
- Τη δημιουργία νέου περιεχομένου από ένα ήδη υπάρχον, όπως η δημιουργία ενός πίνακα περιεχομένων
- Την παρουσίαση της πληροφορίας με το κατάλληλο για τον αναγνώστη επίπεδο λεπτομέρειας, όπως για παράδειγμα η χρήση φύλλου στυλ για την παρουσίαση πληροφοριών υψηλού επιπέδου σε κάποιο πρόσωπο της διοίκησης ενώ παράλληλα χρησιμοποιείται άλλο φύλλο στυλ για την παρουσίαση λεπτομερέστερων τεχνικών πληροφοριών στο υπόλοιπο προσωπικό
- Το μετασχηματισμό XML εγγράφων σε HTML για προς τα πίσω συμβατότητα με υπάρχοντες φυλλομετρητές (browsers)

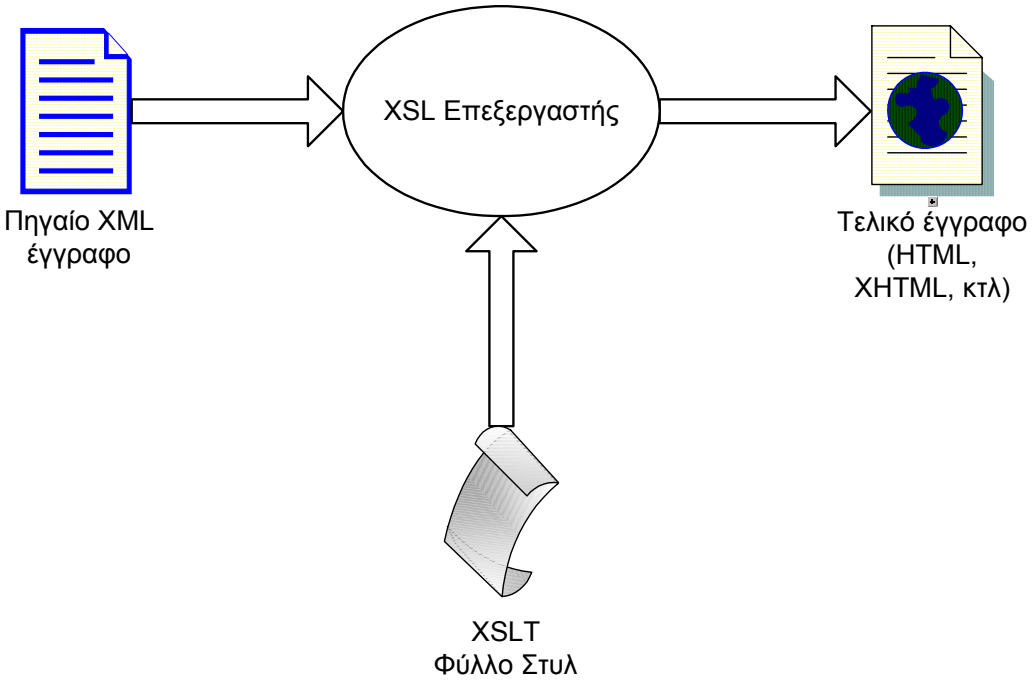
Το κεφάλαιο αυτό θα ασχοληθεί με το μετασχηματισμό XML εγγράφων σε HTML έγγραφα, καθότι αυτού του είδους ο μετασχηματισμός συναντάται πιο συχνά.

Προκειμένου να επιτευχθεί ένας XSLT μετασχηματισμός, απαιτούνται δύο έγγραφα, το προς μετασχηματισμό έγγραφο (που ονομάζεται και έγγραφο εισόδου ή πηγαίο έγγραφο) καθώς και το φύλλο στυλ που περιέχει τις οδηγίες μετασχηματισμού. Ο XSLT επεξεργαστής μετατρέπει το αρχείο εισόδου με βάση τις οδηγίες μετασχηματισμού που περιέχονται στο φύλλο στυλ και παράγει το τελικό έγγραφο, που ονομάζεται και έγγραφο εξόδου. Θα πρέπει να σημειωθεί ότι τόσο το πηγαίο XML έγγραφο, όσο και το XSLT φύλλο στυλ είναι καλά ορισμένα XML έγγραφα, ακολουθούν δηλαδή τους συντακτικούς κανόνες της XML. Σχηματικά, η διαδικασία μετασχηματισμού απεικονίζεται στην Εικόνα 5-1.

Το έγγραφο που ακολουθεί (με) είναι ένα καλά ορισμένο XML έγγραφο και περιέχει πληροφορία που σχετίζεται με βιβλία, όπως τίτλος βιβλίου, συγγραφέας, χρονιά έκδοσης κτλ. Στο έγγραφο αυτό θα εφαρμοστούν όλα τα είδη των μετασχηματισμών που θα αναφερθούν στις επόμενες παραγράφους. Προκειμένου ένας XSLT επεξεργαστής να μετασχηματίσει ένα XML έγγραφο, θα πρέπει στο έγγραφο αυτό να εμφανίζεται η οδηγία επεξεργασίας `<?xml-stylesheet?>` όπου θα υποδεικνύει στον επεξεργαστή ποιο XSLT φύλλο στυλ να χρησιμοποιήσει. Συγκεκριμένα μέσω της ιδιότητας `href` καθορίζεται το URI του XSLT φύλλου στυλ (στο συγκεκριμένο παράδειγμα είναι το `library_books.xsl`), ενώ η ιδιότητα `type` χρησιμοποιείται για τον καθορισμό του περιεχομένου του εγγράφου όπως αυτό προτείνεται από το πρότυπο MIME και παίρνει την τιμή `text/xml`.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="library_books.xsl"?>
<LIBRARY>
      <BOOK>
            <TITLE>Learning XML</TITLE>
            <AUTHOR>Erik T. Ray</AUTHOR>
            <PUBLISHER>O'REILLY</PUBLISHER>
            <PRICE>55</PRICE>
            <YEAR>2001</YEAR>
            <ISBN>0-596-00046-4</ISBN>
      </BOOK>
      <BOOK>
            <TITLE>Inside XML</TITLE>
```

```
            <AUTHOR>Steven Holzner</AUTHOR>
            <PUBLISHER>New Riders</PUBLISHER>
            <PRICE>49.99</PRICE>
            <YEAR>2001</YEAR>
            <ISBN>0-7357-1020-1</ISBN>
      </BOOK>
      <BOOK>
            <TITLE>Metadata Management</TITLE>
            <AUTHOR>Guy Tozer</AUTHOR>
            <PUBLISHER>Artech House Publishers</PUBLISHER>
            <PRICE>50</PRICE>
            <YEAR>2001</YEAR>
            <ISBN>0-89006-280-3</ISBN>
      </BOOK>
</LIBRARY>
```



Εικόνα 5-3 Η διαδικασία μετασχηματισμού ενός XML εγγράφου μέσω XSLT φύλλων στυλ

Η διαδικασία μετασχηματισμού των XML εγγράφων πραγματοποιείται με έναν από τους εξής τρεις τρόπους:

- **Μετασχηματισμός στην πλευρά του εξυπηρετητή (server)**. Ένα λογισμικό που εκτελείται στον εξυπηρετητή, όπως για παράδειγμα ένα Java servlet, μπορεί να χρησιμοποιήσει ένα φύλλο στυλ για να μετασχηματίσει ένα έγγραφο και να στείλει τη τελική του μορφή στο πελάτη
- **Μετασχηματισμός στην πλευρά του πελάτη (client)**. Ένα πρόγραμμα που εκτελείται στον πελάτη, όπως για παράδειγμα ένας φυλλομετρητής (browser),

εκτελεί το μετασχηματισμό, διαβάζοντας το φύλλο στυλ όπως αυτό καθορίζεται από την οδηγία επεξεργασίας `<?xml-stylesheet?>`.

- **Μετασχηματισμός μέσω ανεξάρτητων εφαρμογών**. Ένας μεγάλος αριθμός από (μη-διαδικτυακές) εφαρμογές, συνήθως ανεπτυγμένες σε Java, παρέχουν τη δυνατότητα XSLT μετασχηματισμών

5.2 Δημιουργία XSLT φύλλων στυλ

Οι μετασχηματισμοί XSLT δέχονται ως είσοδο ένα έγγραφο σε δενδροειδή μορφή και ως έξοδο παράγουν ένα άλλο έγγραφο σε δενδροειδή μορφή. Η XSLT διαχειρίζεται τα έγγραφα ως δένδρα που αποτελούνται από κόμβους. Η XSLT αναγνωρίζει εφτά είδη κόμβων:

Κόμβος	**Περιγραφή**
Ρίζα εγγράφου	Ο κόμβος αυτός αναπαριστά την αρχή του εγγράφου. Ο κόμβος αυτός δεν πρέπει να σχετίζεται με το στοιχείο ρίζας του εγγράφου. Ο κόμβος αυτός περιέχει το στοιχείο ρίζας του εγγράφου και ότι οτιδήποτε άλλο εκτός της XML δήλωσης.
Στοιχείο	Ένας κόμβος-στοιχείο μπορεί να περιέχει άλλα στοιχεία, και οποιοδήποτε άλλο είδος κόμβου εκτός από τον κόμβο ρίζας
Ιδιότητα	Μια ιδιότητα παρόλο που περιέχεται σε ένα στοιχείο, περιγράφει πληροφορία, άρα μπορεί να αναπαρασταθεί σαν κόμβος. Ο κόμβος που αναπαριστά μια ιδιότητα καλείται κόμβος-φύλλο διότι στην δενδροειδή αναπαράσταση δεν περιέχει απογόνους
Σχόλιο	Περιέχει το κείμενο ενός σχολίου, χωρίς τους χαρακτήρες `<!--` και `-->`
Χώρος Ονοματοδοσίας	Κόμβος που περιέχει το URI ενός χώρου ονοματοδοσίας.
Οδηγίες επεξεργασίας	Περιέχει το κείμενο μιας οδηγίας επεξεργασίας, εκτός των χαρακτήρων `<?` και `?>`
Κείμενο	Περιέχει το κείμενο ενός κόμβου

Η XSLT παρέχει μια σειρά από μεθόδους προκειμένου να αντιστοιχηθούν ή να επιλεγούν οι κόμβοι που πρόκειται να επεξεργαστούν. Για παράδειγμα, ο χαρακτήρας / αντιστοιχεί στον κόμβο ρίζα. Στο παράδειγμα που ακολουθεί, περιγράφεται ο τρόπος ανάπτυξης ενός απλού XSLT εγγράφου στο οποίο ο κόμβος ρίζα, και συνεπώς όλο το έγγραφο, αντικαθίσταται με μια HTML σελίδα.

Τα XSLT έγγραφα είναι και αυτά καλά ορισμένα XML έγγραφα, οπότε κάθε έγγραφο πρέπει να ξεκινάει με μια XML δήλωση. Στη συνέχεια χρησιμοποιείται το στοιχείο `stylesheet` στο οποίο δηλώνεται ο χώρος ονοματοδοσίας που αντιστοιχεί στο URI `http://www.w3.org/1999/XSL/Transform`.

```
<?xml version="1.0"?>
```

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Η επεξεργασία των κόμβων γίνεται με τη χρήση *προτύπων* (*templates*). Κατά την αντιστοίχηση ή την επιλογή των κόμβων, το πρότυπο ενημερώνει τον XSLT επεξεργαστή πώς να μετασχηματίσει ένα συγκεκριμένο κόμβο. Στην περίπτωση του παραδείγματος, χρησιμοποιείται το στοιχείο <xsl:templates> στο οποίο η τιμή του ορίσματος match καθορίζει τον κόμβο που πρέπει να αντιστοιχηθεί.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
.
.
.
    </xsl:template>
</xsl:stylesheet>
```

Αφού αντιστοιχηθεί ο κόμβος ρίζας, το πρότυπο εφαρμόζεται στον κόμβο αυτό. Εδώ χρειάζεται να αντικατασταθεί ο κόμβος ρίζας με ένα HTML έγγραφο, συνεπώς

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <HTML>
            <HEAD>
                <TITLE>
                    Ένας απλός μετασχηματισμός
                </TITLE>
            </HEAD>
            <BODY>
                Ο μετασχηματισμός αυτός αντικαθιστά
                όλο το έγγραφο
            </BODY>
        </HTML>
    </xsl:template>
</xsl:stylesheet>
```

Η χρήση του στοιχείου <xsl:template> θέτει έναν κανόνα στο Φύλλο Στυλ. Όταν ο XSL επεξεργαστής αναλύει ένα έγγραφο, ο πρώτος κόμβος που συναντά είναι ο κόμβος ρίζα. Ο κανόνας που θέτει το πρότυπο εφαρμόζεται στον κόμβο ρίζα με αποτέλεσμα ο XSL επεξεργαστής να τον αντικαθιστά με ένα HTML έγγραφο, παράγοντας το παρακάτω αποτέλεσμα

```
<HTML>
      <HEAD>
            <TITLE>
                  Ένας απλός μετασχηματισμός
            </TITLE>
      </HEAD>
      <BODY>
```

```
            Ο μετασχηματισμός αυτός αντικαθιστά
            όλο το έγγραφο
      </BODY>
</HTML>
```

Εξαγωγή της τιμής ενός κόμβου με χρήση του στοιχείου <xsl:value-of>

Προκειμένου να εξαχθεί η τιμή ενός συγκεκριμένου κόμβου ώστε να εισαχθεί στο εξαγόμενο έγγραφο, χρησιμοποιείται το στοιχείο <xsl:value-of>.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
      <html>
            <BODY> Τίτλος Βιβλίου <HR/>
               <b><xsl:value-of select="LIBRARY/BOOK/TITLE"/><b><BR/>
            </BODY>
      </html>
  </xsl:template>
</xsl:stylesheet>
```

Στο παράδειγμα αυτό ζητείται να επιστραφεί ο τίτλος του βιβλίου που βρίσκεται πρώτο σε σειρά στο XML έγγραφο. Η επιλογή του κατάλληλου κόμβου γίνεται με τη χρήση της ιδιότητας select η οποία είναι παρόμοια με την ιδιότητα match του στοιχείου <xsl:template>, με τη διαφορά ότι μέσω της ιδιότητας select οι κόμβοι επιλέγονται με τη χρήση της γλώσσας XPath η οποία περιγράφεται παρακάτω. Στη συγκεκριμένη περίπτωση μέσω της XPath ορίζεται ένα μονοπάτι που ξεκινάει από το στοιχείο ρίζας (LIBRARY) και καταλήγει στον επιθυμητό κόμβο, το εξαγόμενο έγγραφο θα έχει την εξής μορφή:

```
<HTML>
      <BODY>
            Τίτλος Βιβλίου <HR/>
            <b> Learning XML</b><BR/>
      </BODY>
</HTML>
```

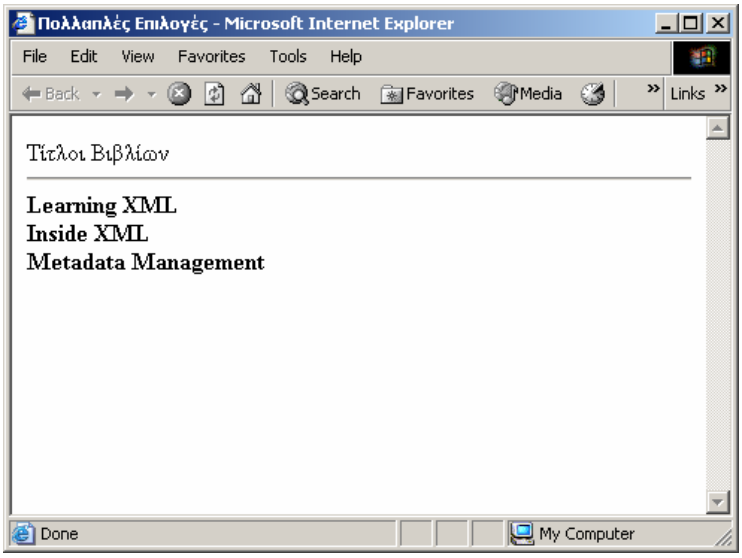
Διαχείριση πολλαπλών επιλογών με χρήση του στοιχείου <xsl:for-each>

Μέσω της ιδιότητας select επιλέγεται μόνο ο πρώτος κόμβος που εκπληρώνει το κριτήριο επιλογής. Στην περίπτωση ύπαρξης πολλών κόμβων που εκπληρώνουν το κριτήριο, χρησιμοποιείται το στοιχείο <xsl:for-each>. Στην περίπτωση δηλαδή που θέλουμε να επιστραφούν όλοι οι τίτλοι των βιβλίων που περιγράφονται στο έγγραφο books.xml, το XSL έγγραφο θα έχει την εξής μορφή

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="
http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
   <html>
     <head>
       <title>Πολλαπλές Επιλογές</title>
     </head>
     <body>Τίτλοι Βιβλίων <HR/>
       <xsl:for-each select="LIBRARY/BOOK">
            <b><xsl:value-of select="TITLE"/> </b><BR/>
```

```
        </xsl:for-each>
      </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Το αποτέλεσμα του μετασχηματισμού φαίνεται στην παρακάτω εικόνα



Το στοιχείο <xsl:apply-templates>

Ένας εναλλακτικός τρόπος για την εφαρμογή προτύπων στα παιδιά ενός συγκεκριμένου κόμβου, είναι με τη χρήση του στοιχείου `<xsl:apply-templates>`.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
       <html>
            <BODY> Τίτλος Βιβλίου <HR/>
             <b><xsl:apply-templates select="LIBRARY/BOOK"/></b><BR/>
            </BODY>
       </html>

  <xsl:template match="LIBRARY/BOOK">
       <tr><td>
          <xsl:value-of select="TITLE"/>
       </td></tr>
  </xsl:template>
</xsl:stylesheet>
```

Στο παραπάνω XSL έγγραφο ζητείται να επιστραφεί ο τίτλος του πρώτου βιβλίου που περιγράφεται στο έγγραφο `books.xml`.

5.2.1 Επιλογή κόμβων με χρήση της τεχνολογίας XPath

Προκειμένου να γίνει οποιοδήποτε είδος επεξεργασίας ενός XML εγγράφου, μέσω του XSLT, θα πρέπει να παρέχεται η δυνατότητα εύκολης μετακίνησης σε ένα οποιοδήποτε κόμβο του εγγράφου, η δυνατότητα αναγνώρισης του τρέχοντα κόμβου και η δυνατότητα επιλογής μιας ομάδας κόμβων για περαιτέρω επεξεργασία. Αυτές οι ικανότητες πλοήγησης

σε διάφορα σημεία του XML εγγράφου γίνεται μέσω της τεχνολογίας XPath, που παρέχει μια εξεζητημένη γλώσσα για τη σήμανση τοποθεσιών και την επιλογή ενός συνόλου από κόμβους σε ένα έγγραφο.

Μονοπάτια Ελέγχου

Ο καθορισμός ενός κόμβου ή μιας ομάδας κόμβων, μέσω της XPath, γίνεται με τη χρήση των *διαδρομών ελέγχου* (*location paths*). Μια διαδρομή ελέγχου αποτελείται από ένα ή περισσότερα βήματα, τα οποία διαχωρίζονται με τη χρήση των χαρακτήρων "/" ή "//". Υπάρχουν δύο ειδών διαδρομών ελέγχου, οι απόλυτες και οι σχετικές διαδρομές. Οι απόλυτες διαδρομές καθορίζουν την έναρξη της διαδρομής από τον κόμβο-ρίζα και η περιγραφή της διαδρομής ξεκινάει με το χαρακτήρα "/". Η έναρξη των σχετικών διαδρομών καθορίζεται από τον εκάστοτε τρέχοντα κόμβο.

Μια διαδρομή ελέγχου αποτελείται από μια σειρά βημάτων, καθένα από τα οποία μεταφέρει τη διαδρομή στο τελικό της σημείο. Ένα βήμα αποτελείται από τρία μέρη: έναν άξονα που περιγράφει τη διεύθυνση της πορείας, ένα κόμβο ελέγχου που περιγράφει τι είδους κόμβοι είναι εφαρμόσιμοι, και ένα σύνολο από δηλώσεις που χρησιμοποιούν Boolean Τιμές 0 ή 1 (σωστό/λάθος).

Άξονες

Η XPath ορίζει ένα μεγάλο αριθμό αξόνων, οι οποίοι παρατίθενται στον πίνακα που ακολουθεί:

Είδη αξόνων	Αντιστοίχηση
`Ancestor`	Όλοι οι πρόγονοι του εκάστοτε τρέχοντα κόμβου. Περιλαμβάνονται τα στοιχεία στα οποία περιέχεται το στοιχείο αυτό, συμπεριλαμβανομένου και του κόμβου ρίζα.
`Ancestor-or-self`	Ο τρέχοντας κόμβος και όλοι οι πρόγονοί του.
`Attribute`	Περιέχει τις ιδιότητες του τρέχοντα κόμβου
`Child`	Τα παιδιά του τρέχοντα κόμβου.
`Descendant`	Οι απόγονοι του εκάστοτε τρέχοντα κόμβου
`Descendant -or-self`	Όπως και πάνω, αλλά περιέχει και τον εκάστοτε τρέχοντα κόμβο.
`Following`	Όλοι οι κόμβοι που ακολουθούν τον τρέχοντα κόμβο.
`Following-sibling`	Όλοι οι κόμβοι που ακολουθούν τον τρέχοντα κόμβο και βρίσκονται στο ίδιο επίπεδο με αυτό.
`Namespace`	Όλοι οι κόμβοι με το συγκεκριμένο χώρο ονοματοδοσίας.
`Parent`	Ο γονέας του τρέχοντα κόμβου
`Preceding`	Όλοι οι κόμβοι που έπονται του τρέχοντα κόμβου.
`Preceding-sibling`	Όλοι οι κόμβοι που έπονται του τρέχοντα κόμβου και βρίσκονται στο ίδιο επίπεδο

	με αυτό.
`Self`	Ο τρέχοντας κόμβος

Κόμβοι ελέγχου

Οι κόμβοι ελέγχου συνδέονται με τους άξονες μέσω των χαρακτήρων "::". Σε αρκετές περιπτώσεις αντί των όρων που παρέχουν οι κόμβοι έλεγχου (και παρατίθενται στον παρακάτω πίνακα) χρησιμοποιούνται ονόματα κόμβων ή ο χαρακτήρας "*" για την επιλογή ενός συνόλου κόμβων. Για παράδειγμα η δήλωση `child::*/child::TITLE` επιλέγει όλα τα στοιχεία `<TITLE>` που είναι εγγόνια του τρέχοντα στοιχείου. Οι κυριότεροι όροι των κόμβων ελέγχου περιέχονται στον πίνακα που ακολουθεί.

Όρος	**Αντιστοιχία**
/	Ο κόμβος ρίζας:όχι το στοιχείο ρίζας, αλλά ο κόμβος που περιέχει το στοιχείο της ρίζας και κάθε σχόλιο ή οδηγίες επεξεργασίας που προηγούνται αυτής
`node()`	Κάθε κόμβος εκτός της ρίζας και των χαρακτηριστικών.
*	Στον άξονα χαρακτηριστικών, κάθε χαρακτηριστικό. Στον άξονα του χώρου ονοματοδοσίας, κάθε χώρος ονοματοδοσίας. Σε όλους τους άλλους άξονες κάθε στοιχείο.
`text()`	Κάθε κόμβος κειμένου.
`Processing-instruction()`	Κάθε οδηγία επεξεργασίας.
`comment()`	Κάθε κόμβος σχολίου.
`@role`	Μια ιδιότητα με όνομα role.
.	Ο τρέχοντας κόμβος

Δηλώσεις

Το μέρος των δηλώσεων είναι και το πιο ενδιαφέρον, διότι παρέχει και τις περισσότερες δυνατότητες. Μερικοί από τους πιο σημαντικούς τύπους δηλώσεων είναι τα σύνολα κόμβων (node sets), οι Boolean δηλώσεις που επιστρέφουν τιμή 0 ή 1, και δηλώσεις αλφαριθμητικών (strings).

Μια δήλωση του τύπου `child::BOOK` επιστρέφει ένα σύνολο κόμβων που αποτελείται από όλα τα στοιχεία `<BOOK>`. Μερικές από τις πιο σημαντικές δηλώσεις συνόλων κόμβου είναι το `position()` που επιστρέφει τη θέση του τρέχοντα κόμβου στο σύνολο κόμβων που έχει επιλεχθεί (αρχίζοντας από το 1) και το `count()` που επιστρέφει τον αριθμό των κόμβων που περιέχεται σε ένα σύνολο κόμβων.

Στον πίνακα που ακολουθεί παρουσιάζονται οι λογικοί τελεστές που χρησιμοποιούνται στις Boolean δηλώσεις:

Τελεστής	**Περιγραφή**
!=	Διάφορο
<	Μικρότερο από
<=	Μικρότερο ή ίσο από

>	Μεγαλύτερο από
=>	Μεγαλύτερο ή ίσο από
=	Ισούται με

Στο ακόλουθο πρότυπο χρησιμοποιείται ο λογικός τελεστής >. Ο κανόνας αυτός εφαρμόζεται σε όλα τα στοιχεία <BOOK> που βρίσκονται μετά τη δεύτερη θέση.

```
<xsl:template match="BOOK[position() > 2]">
    <xsl:value-of select="."/>
</xsl:template>
```

Τέλος, η XPath παρέχει μια σειρά συναρτήσεων για την επεξεργασία αλφαριθμητικών, μερικές από τις οποίες παρατίθενται στον επόμενο πίνακα:

Συνάρτηση	**Περιγραφή**
`starts-with(string s1, string s2)`	Επιστρέφει true αν το πρώτο αλφαριθμητικό ξεκινάει με το δεύτερο
`contains(string s1, string s2)`	Επιστρέφει true αν το πρώτο αλφαριθμητικό περιέχει το δεύτερο

Παραδείγματα XPath δηλώσεων

Στη συνέχεια παρατίθενται μια σειρά παραδειγμάτων προκειμένου να γίνει πιο κατανοητή η χρήση της XPath κατά την επιλογή των κόμβων. Οι δηλώσεις XPath εφαρμόζονται στο αρχείο books.xml

Παράδειγμα	**Αποτέλεσμα**
`child::BOOK`	Επιστρέφει το στοιχείο <BOOK> που είναι παιδί του τρέχοντα κόμβου
`child::*`	Επιστρέφει όλα τα στοιχεία παιδιά του τρέχοντος κόμβου
`descendant::BOOK`	Επιστρέφει όλα τα στοιχεία BOOK που είναι απόγονοι του τρέχοντα κόμβου
`//BOOK`	Επιστρέφει όλα τα στοιχεία <BOOK>. Ισοδυναμεί με το `descendant::BOOK`
`@*`	Επιστρέφει όλα τις ιδιότητες του τρέχοντα κόμβου
`BOOK[3]`	Επιστρέφει το τρίτο στοιχείο <BOOK> που είναι παιδί του τρέχοντα κόμβου
`//BOOK/AUTHOR`	Επιστρέφει όλα τα στοιχεία <AUTHOR> που έχουν ως γονέα το στοιχείο <BOOK>
`BOOK[AUTHOR]`	Επιστρέφει όλα τα στοιχεία <BOOK> που περιέχουν το στοιχείο <AUTHOR> και που

	είναι παιδιά του τρέχοντος κόμβου
BOOK[AUTHOR="Erik T. Ray"]	Επιστρέφει τα στοιχεία <BOOK> των οποίων το κείμενο των στοιχείων παιδιών <AUTHOR> είναι ίσο με Erik T. Ray

Όπως φάνηκε στον παραπάνω πίνακα η δήλωση `descendant::BOOK` ισοδυναμεί με τη δήλωση `//BOOK`. Η γλώσσα XPath παρέχει μια σειρά συντομεύσεων που διευκολύνουν τη συγγραφή των δηλώσεων. Στον πίνακα που ακολουθεί περιέχονται οι κανόνες συντομογραφίας

Δήλωση	**Συντομογραφία**	**Περιγραφή**
self::node	.	Ο τρέχον κόμβος
parent::node	..	Ο γονέας του τρέχοντος κόμβου
child::childname	childname	Το στοιχείο childname που είναι παιδί του τρέχοντος κόμβου
attribute::childname	@childname	Η ιδιότητα childname του τρέχοντος κόμβου
/descendant-or-self::node()/	//	Επιστρέφει όλους τους συγκεκριμένους κόμβους που είναι απόγονοι του κόμβου ρίχα

Κάποια περαιτέρω παραδείγματα είναι τα εξής:

Επιλογή του πρώτου BBB παιδιού του AAA

Δήλωση: **/AAA/BBB[1]**

```
<AAA>
      <BBB/>
      <BBB/>
      <BBB/>
</AAA>
```

Επιλογή του τελευταίου στοιχείου BBB που είναι παιδί του στοιχείου AAA

Δήλωση: **/AAA/BBB[last()]**

```
<AAA>
     <BBB/>
     <BBB/>
     <BBB/>
     <BBB/>
</AAA>
```

Επιλογή των στοιχείων BBB που έχουν την ιδιότητα "id"

Δήλωση: **//BBB[@id]**

```
<AAA>
    <BBB id = "b1"/>
    <BBB id = "b2"/>
    <BBB name = "bbb"/>
    <BBB/>
</AAA>
```

Επιλογή όλων των στοιχείων που έχουν το στοιχείο CCC ανάμεσα στους προγόνους τους

Δήλωση: **//CCC/descendant::***

```
<AAA>
   <BBB>
      <DDD>
         <CCC>
            <DDD/>
            <EEE/>
         </CCC>
      </DDD>
   </BBB>
   <CCC>
      <DDD>
         <EEE>
            <DDD>
               <FFF/>
            </DDD>
         </EEE>
      </DDD>
   </CCC>
</AAA>
```

Επιλογή των γονέων όλων των στοιχείων DDD

Δήλωση: **//DDD/parent::***

```
<AAA>
    <BBB>
        <DDD>
          <CCC>
           <DDD/>
          <EEE/>
          </CCC>
        </DDD>
    </BBB>
    <CCC>
        <GGG>
            <EEE>
                <DDD>
                    <FFF/>
                </DDD>
            </EEE>
        </GGG>
    </CCC>
</AAA>
```

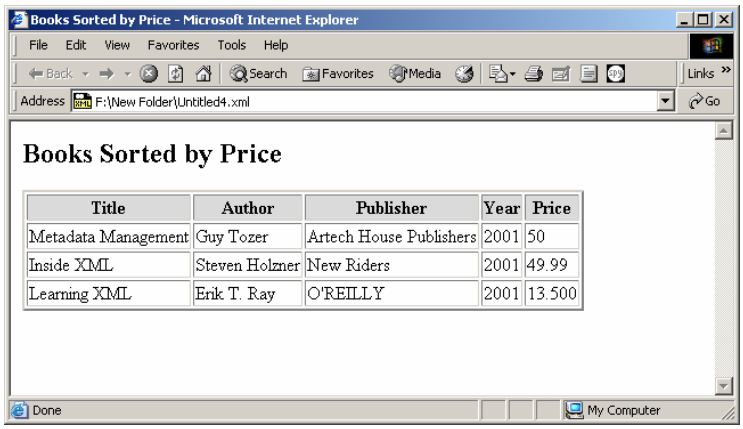
5.2.2 Ταξινόμηση στοιχείων

Προκειμένου να ταξινομηθεί ένα σύνολο κόμβων χρησιμοποιείται το στοιχείο <xsl:sort>. Το XSL έγγραφο που ακολουθεί ταξινομεί τα βιβλία που περιγράφονται στο έγγραφο books.xml κατά φθίνουσα σειρά των τιμών τους.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <head>
            <title>Books Sorted by Price</title>
         </head>
         <body>
            <h2>Books Sorted by Price</h2>
            <table border="2">
               <tr bgcolor="#DADADA">
                  <th>Title</th>
                  <th>Author</th>
                  <th>Publisher</th>
                  <th>Year</th>
                  <th>Price</th>
               </tr>
               <xsl:for-each select="LIBRARY/BOOK">
                  <xsl:sort select="PRICE" order="descending"/>
                  <tr>
                        <td>
                           <xsl:value-of select="TITLE"/>
                        </td>
                        <td>
                           <xsl:value-of select="AUTHOR"/>
                        </td>
                        <td>
                           <xsl:value-of select="PUBLISHER"/>
                        </td>

                        <td>
                           <xsl:value-of select="YEAR"/>
                        </td>
                        <td>
                           <xsl:value-of select="PRICE"/>
                        </td>
                  </tr>
               </xsl:for-each>
            </table>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

Το αποτέλεσμα του μετασχηματισμού φαίνεται στην παρακάτω εικόνα



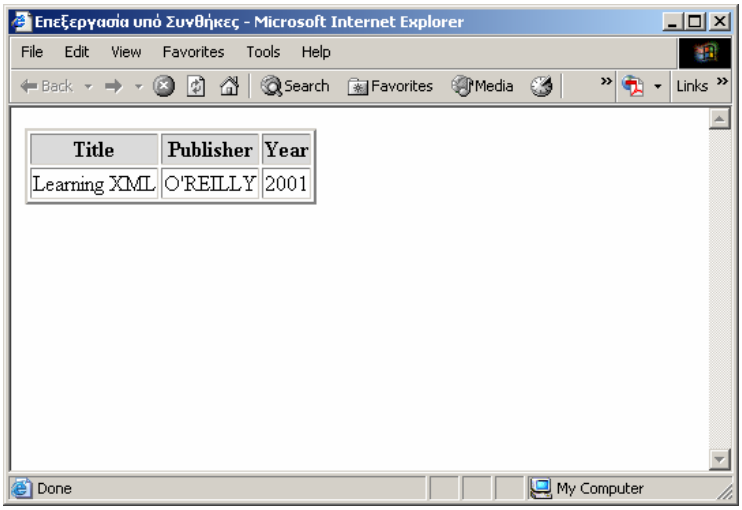
5.2.3 Επεξεργασία υπό συνθήκες

Το στοιχείο <xsl:if>

Ένα είδος επεξεργασίας υπό συνθήκες παρέχεται με τη χρήση του στοιχείου `<xsl:if>`. Το ακόλουθο XSL έγγραφο παρουσιάζει σε ένα πίνακα τον τίτλο, την τιμή και τον εκδότη του βιβλίου που ο συγγραφέας είναι ο `Erik T. Ray`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <head>
            <title>
               Επεξεργασία υπό Συνθήκες
            </title>
         </head>
         <body>
            <table border="2">
               <tr bgcolor="#DADADA">
                  <th>Title</th>
                  <th>Publisher</th>
                  <th>Year</th>
               </tr>
               <xsl:for-each select="LIBRARY/BOOK">
                  <xsl:if test="(AUTHOR='Erik T. Ray')">
                     <tr>
                        <td>
                           <xsl:value-of select="TITLE"/>
                        </td>
                        <td>
                           <xsl:value-of select="PUBLISHER"/>
                        </td>
                        <td>
                           <xsl:value-of select="YEAR"/>
                        </td>
                     </tr>
                  </xsl:if>
               </xsl:for-each>
            </table>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

Το αποτέλεσμα της επεξεργασίας φαίνεται στην παρακάτω εικόνα



Το στοιχείο <xsl:choose>

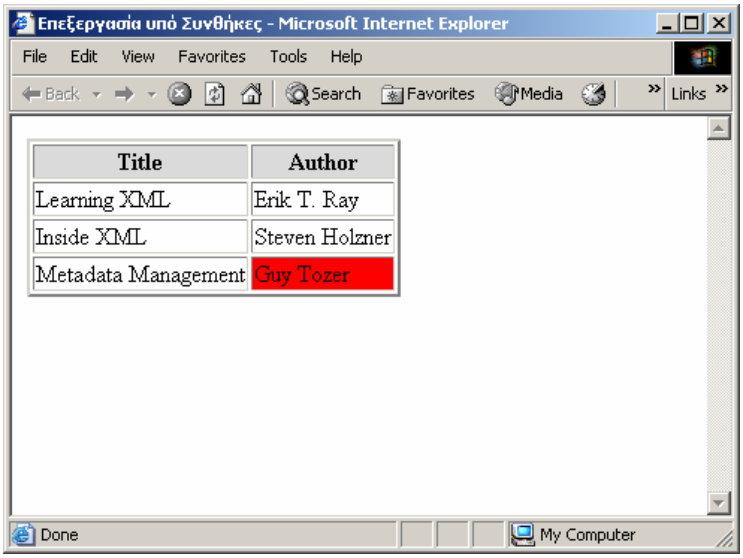
Με τη χρήση του στοιχείου `<xsl:choose>` δίνεται η δυνατότητα επιλογής από ένα πλήθος εναλλακτικών δράσεων. Το στοιχείο αυτό αντιστοιχεί στη δομή `if –then-else` (στην περίπτωση επιλογής από δύο εναλλακτικές δράσεις) ή στη δομή `switch` (στην περίπτωση επιλογής από περισσότερες από δύο εναλλακτικές λύσεις) που συναντώνται στις γλώσσες προγραμματισμού.

Το ακόλουθο XSL έγγραφο επιστρέφει σε ένα πίνακα τον τίτλο και τον συγγραφέα των βιβλίων που περιγράφονται στο έγγραφο books.xml. Το κελί του πίνακα που περιέχει το συγγραφέα `Guy Tozer` μαρκάρεται με διαφορετικό χρώμα από τα υπόλοιπα κελιά

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <head>
            <title>Επεξεργασία υπό Συνθήκες</title>
         </head>
         <body>
            <table border="2">
               <tr bgcolor="#DADADA">
                  <th>Title</th>
                  <th>Author</th>
               </tr>
               <xsl:for-each select="LIBRARY/BOOK">
                  <tr>
                     <td>
                        <xsl:value-of select="TITLE"/>
                     </td>
                     <xsl:choose>
                        <xsl:when test="(AUTHOR='Guy Tozer')">
                           <td bgcolor="red">
                              <xsl:value-of select="AUTHOR"/>
                           </td>
                        </xsl:when>
                        <xsl:otherwise>
                           <td>
                              <xsl:value-of select="AUTHOR"/>
```

```
                    </td>
                  </xsl:otherwise>
                </xsl:choose>
              </tr>
            </xsl:for-each>
          </table>
        </body>
      </html>
    </xsl:template>
</xsl:stylesheet>
```

Το αποτέλεσμα του μετασχηματισμού απεικονίζεται στην παρακάτω εικόνα



5.3 Ασκήσεις

Άσκηση 1

Να σχεδιαστεί ένα XSL έγγραφο που να μετασχηματίζει τα XML έγγραφα που αναπτύχθηκαν στην Άσκηση 3 του Κεφαλαίου 3.

Ένα XSL έγγραφο που θα αποθηκεύει τα πεδία TITLE, AUTHOR, PUBLISHER και YEAR σε ένα πίνακα μιας HTML σελίδας θα έχει την παρακάτω μορφή

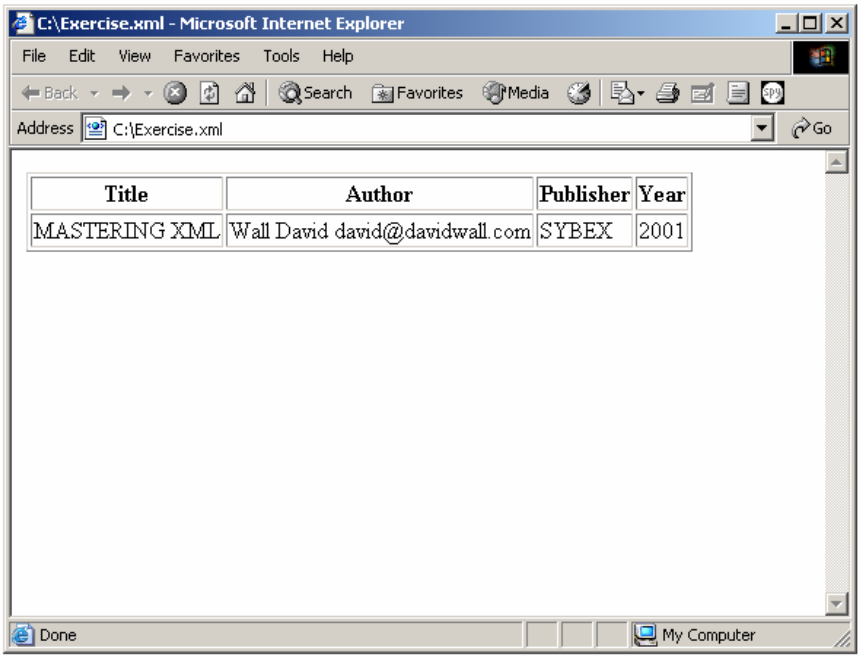
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <table  border="1">
          <tr>
            <th>Title</th>
            <th>Author</th>
            <th>Publisher</th>
            <th>Year</th>
          </tr>
```

```
            <xsl:for-each select="eBook">
               <tr>
                  <td><xsl:value-of select="TITLE"/></td>
                  <td><xsl:value-of select="AUTHOR/."/></td>
                  <td><xsl:value-of select="PUBLISHER"/></td>
                  <td><xsl:value-of select="YEAR"/></td>
               </tr>
            </xsl:for-each>
            </table>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

Προκειμένου να γνωρίζει ο parser, που θα επεξεργαστεί το XML έγγραφο, ποιο XSL έγγραφο χρησιμοποιεί για να μορφοποιηθεί, θα πρέπει να προστεθεί η γραμμή

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="display.xsl"?>
<eBook xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Exercise.xsd">
…
```

Display.xsl είναι το όνομα του XSL εγγράφου. Η παρουσίαση του παραπάνω XML εγγράφου σε συνδυασμό με το XSL έγγραφο φαίνεται στην παρακάτω εικόνα



6 XML και Βάσεις Δεδομένων

Σ' αυτό το κεφάλαιο θα εξετάσουμε ορισμένες μεθόδους μεταφοράς μιας ήδη υπάρχουσας ΒΔ σε XML.

Με την πληθώρα των επιχειρησιακών δεδομένων αποθηκευμένα σε σχεσιακές ΒΔ, υπάρχουν αρκετοί λόγοι για την έκθεσή τους ως XML.

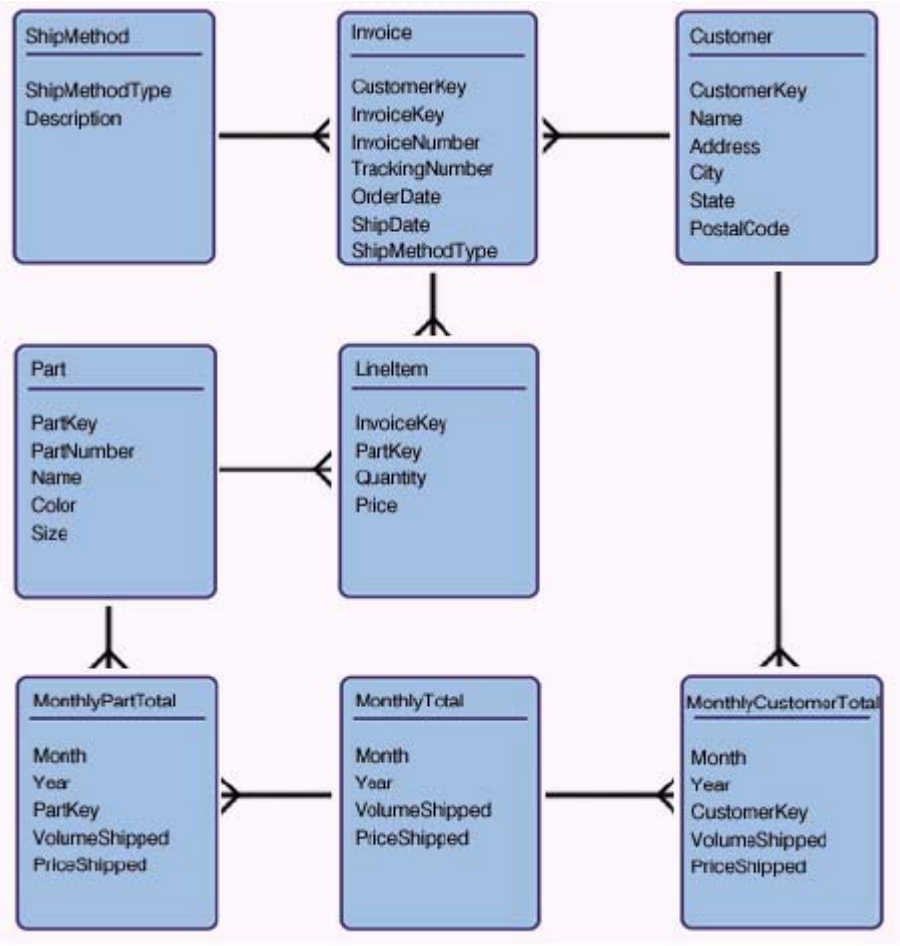
1) Διαμοιρασμός επιχειρησιακών δεδομένων με άλλα συστήματα.
2) Διαλειτουργικότητα με μη συμβατά συστήματα.
3) Εκθέτοντας τα στοιχεία κληρονομιών σε εφαρμογές που χρησιμοποιούν XML
4) Ενδοεπιχειρησιακές συναλλαγές.
5) Εμμονή αντικειμένου που χρησιμοποιεί XML.

Οι σχεσιακές βάσεις δεδομένων είναι μια ώριμη τεχνολογία και έχουν χρησιμοποιηθεί για την μοντελοποίηση πολύπλοκων σχέσεων μεταξύ δεδομένων που πρέπει να αποθηκευτούν. Στη συνέχεια θα δούμε πώς να μοντελοποιούμε ορισμένες από τις πολύπλοκες δομές ΒΔ και μετά θα δημιουργήσουμε μοντέλα περιεχομένου χρησιμοποιώντας XML DTDs. Θα δείξουμε επίσης ορισμένα παραδείγματα περιεχομένου για τα δεδομένα στην XML. Κατά τη διαδικασία θα δώσουμε ένα σύνολο από κατευθυντήριες γραμμές οι οποίες θα αποδειχθούν πολύ χρήσιμες όταν θα φτιάξουμε μοντέλα XML για σχεσιακά δεδομένα.

Σημειώστε ότι υπάρχουν ήδη ορισμένοι μηχανισμοί που παρέχουν προκαθορισμένο τρόπο για να αντληθεί XML από υπάρχουσες δομές σχεσιακών βάσεων δεδομένων όπως ο ADO 2,5 και ο SQL server 2000 ο οποίος παρέχει απευθείας εξαγωγή από joined δομές, όπως η XML. Παρόλα αυτά, αυτές οι τεχνολογίες ακόμα αναπτύσσονται και δεν μπορούν να χειριστούν πιο πολύπλοκες καταστάσεις όπως πολλά σε πολλά συσχετίσεις, οι οποίες πρέπει να αναπαρασταθούν με IDREF-ID δείκτες. Τώρα θα δούμε πώς οι δομές μπορούν να χειροτεχνηθούν για να αναπαραστήσουν σωστά τέτοιους τύπους συσχετίσεων. Θα ρυθμίσουμε τις δομές μας έτσι ώστε να μεγιστοποιήσουμε την επίδοση και να ελαχιστοποιήσουμε το μέγεθος του εγγράφου.

6.1 Μεταφέροντας μια Βάση Δεδομένων σε XML

Θα χρησιμοποιήσουμε ένα παράδειγμα για να δούμε πως οι κανόνες που φτιάχνουμε μπορούν να εφαρμοστούν σε πραγματικές καταστάσεις. Η δομή που θα μεταφερθεί σε XML θα είναι ένα τιμολόγιο παρακολούθησης και αναφοράς συστήματος και απεικονίζεται στην εικόνα που ακολουθεί.



Η ΒΔ του τιμολογίου περιέχει πληροφορίες για τιμολόγια υποβεβλημένα από πελάτες και τα κομμάτια που έχουν παραγγελθεί σ' αυτά τα τιμολόγια, καθώς και συγκεντρωτικά στοιχεία γι' αυτά τα κομμάτια. Θα δημιουργήσουμε την δομή XML για να περιέχει αυτές τις πληροφορίες.

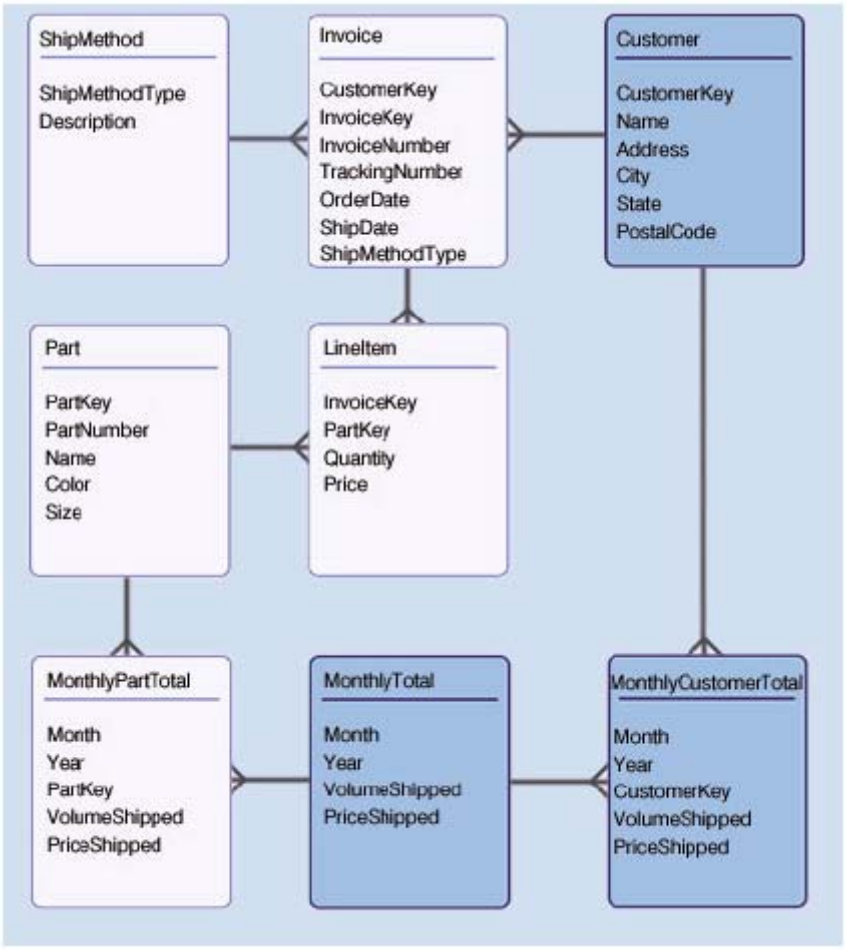
6.1.1 Εύρος του XML εγγράφου

Ο πρώτος κανόνας όταν σχεδιάζουμε μια δομή XML που θα περιέχει πληροφορίες είναι να αποφασίσουμε ποιο θα είναι το εύρος του εγγράφου. Το εύρος αναφέρεται στα δεδομένα και τις συσχετίσεις που θέλουμε να αναπαράγουμε όταν δημιουργούμε το XML έγγραφο – εξ' άλλου όταν εκθέτουμε το περιεχόμενο της ΒΔ μπορεί να μην χρειαστούμε όλα τα δεδομένα που περιέχει η βάση.

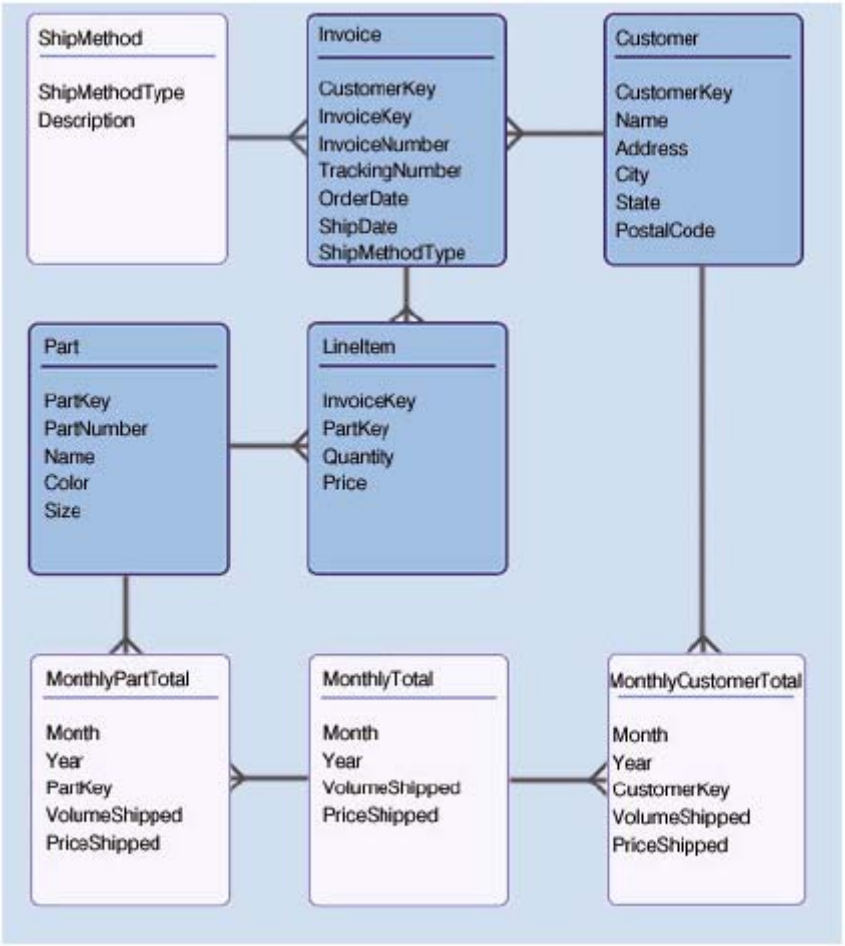
Αν σκεφτούμε να εκτελέσουμε μια αναζήτηση σε μία ΒΔ θα απαιτήσουμε μόνο ένα υποσύνολο απ' όλες τις πληροφορίες που περιέχει. Για παράδειγμα, ένας δικτυακός τόπος ηλεκτρονικού εμπορείου αποθηκεύει δεδομένα με σχέσεις που μοντελοποιούν όλα τα πράγματα που έχει αγοράσει ένας πελάτης στο παρελθόν καθώς και παρούσες παραγγελίες προς επεξεργασία. Εάν γράφαμε μια εφαρμογή CRM (Customer Relationship Management) δε θα έπρεπε απαραίτητα να ανασύρουμε όλες τις παρελθούσες αγορές αλλά μόνο αυτές που έγιναν πρόσφατα. Με λίγα λόγια το εύρος του εγγράφου που φτιάχνουμε καθορίζεται από τις επιχειρησιακές απαιτήσεις, για που θα χρησιμοποιηθούν τα δεδομένα και πώς και αυτές οι επιχειρησιακές απαιτήσεις μπορεί να διαφέρουν.

Για παράδειγμα οι επιχειρησιακές απαιτήσεις μας θα μπορούσαν να είναι να μεταδίδουμε πληροφορίες στο λογιστικό μας γραφείο για την πρόσθεση των μηνιαίων συνόλων των τιμολογίων, καθώς και πελάτη ανά πελάτη ανάλυση ώστε η χρέωση να μπορεί να

εκπληρωθεί. Σ' αυτή την περίπτωση ίσως να θέλουμε να αποσταλεί μόνο ένα συγκεκριμένο υποσύνολο των πληροφοριών στο λογιστικό μας γραφείο.



Μία εναλλακτική επιχειρησιακή απαίτηση μπορεί να είναι να μεταδώσουμε ένα XML αντίγραφο ενός τιμολογίου σ' ένα πελάτη κάθε φορά που ένα καινούργιο υποβάλλεται. Σ' αυτή την περίπτωση το υποσύνολο πληροφοριών προς μετάδοση θα είναι ως εξής:



Επιπρόσθετα μπορεί να θέλουμε να ελέγξουμε τις συγκεκριμένες στήλες που μεταδίδονται. Για παράδειγμα ας πούμε πως ο πελάτης μας ήθελε αναζητήσει ένα προϊόν το οποίο είχε παραγγείλει. Έχει τον αριθμό τιμολογίου για να αναγνωρίσει την αγορά, αλλά δεν νοιάζεται απαραίτητα για τον αριθμό παρακολούθησης τιμολογίου που χρησιμοποιεί η εφαρμογή μας εσωτερικά. Ο επιπλέον αριθμός μπορεί πράγματι να τον μπερδέψει περισσότερο. Αναγνωρίζοντας τους συγκεκριμένους πίνακες και στήλες που πρόκειται να μεταδοθούν αρχίζουμε να συνειδητοποιούμε το πώς ένα XML έγγραφο πρέπει να διαμορφωθεί. Αν τύχει να έχουμε πρόσβαση σ' ένα λογικό διάγραμμα δεδομένων της ΒΔ όπως ένα ERWIN model θα είναι πολύ βοηθητικό στο σχεδιασμό της XML.

ΚΑΝΟΝΑΣ 1

Επιλογή των δεδομένων που θα περιέχονται.

Βασιζόμενοι στις επιχειρησιακές απαιτήσεις που θα εκπληρώνει το XML έγγραφο αποφασίζουμε ποιοι πίνακες και στήλες της ΣΒΔ θα πρέπει να περικλείονται στο έγγραφο.

Για τους σκοπούς του παραδείγματος θα υποθέσουμε ότι όλες οι πληροφορίες στη δομή μας είναι σχετικές στη διαδικασία.

6.1.2 Δημιουργώντας το στοιχείο ρίζα

Απ' την στιγμή που έχουμε ξεκαθαρίσει το εύρος του εγγράφου που χρειαζόμαστε να δημιουργήσουμε, το οποίο μπορεί να καθοριστεί απ' τις επιχειρησιακές απαιτήσεις πρέπει να δημιουργήσουμε το στοιχείο ρίζα μέσα στο οποίο θα φωλιαστεί η XML αναπαράσταση των δεδομένων.

Για το παράδειγμά μας θα δημιουργήσουμε ένα στοιχείο ρίζα, ονομαζόμενο <SalesData >, να περικλείει τα άλλα στοιχεία που θα φτιάξουμε.

```
< SalesData >

….. τα λοιπά στοιχεία γράφονται εδώ.

< /SalesData >
```

Είναι επίσης πιθανό να θελήσουμε να προσθέσουμε ορισμένες πληροφορίες στο XML έγγραφο οι οποίες δεν είναι μέρος της ΣΒΔ Αυτές μπορούν να χρησιμοποιηθούν για να υποδεικνύουν μεταδιδόμενες δρομολογούμενες ή συμπεριφοριακές πληροφορίες. Για παράδειγμα μπορεί να θέλουμε να προσθέσουμε ένα χαρακτηριστικό πηγής έτσι ώστε η διαδικασία να μπορεί να αποφασίσει ποίος custom χειριστής πρέπει να τρέξει για να αναλυθεί το τρέχων έγγραφο.

Αν επιλέξουμε να προσθέσουμε πληροφορίες για το έγγραφο μ’ αυτόν τον τρόπο είναι πιο λογικό να τις προσθέσουμε ως χαρακτηριστικά του στοιχείου ρίζα που δημιουργούμε.

Πάντως πολλοί από τους XML εξυπηρετητές παρέχουν ένα τέτοιου είδους μηχανισμό γνωστό ως φάκελος.

Για το παράδειγμά μας θα προσθέσουμε ένα χαρακτηριστικό στο στοιχείο ρίζα να κατευθύνουμε τι πρέπει να κάνει η consuming διαδικασία με το έγγραφο που παραλαμβάνεται. Ειδικότερα θα προσθέσουμε ένα χαρακτηριστικό *Status*. Αυτό θα επιτρέπει στον επεξεργαστή να γνωρίζει αν οι πληροφορίες στ έγγραφο είναι καινούργιες αν είναι update σε ήδη υπάρχουσες ή αντίγραφο. Μέχρι στιγμής έχουμε την ακόλουθη δομή.

```
< !ELEMENT SalesData EMPTY>

< !ATTLIST SalesData

     Status (NewVersion | UpdatedVersion | CourtesyCopy) # REQUIERED >
```

ΚΑΝΟΝΑΣ 2

Δημιουργία του στοιχείου ρίζα για το έγγραφο

Πρόσθεση του στοιχείου ρίζα στο DTD και δήλωση όποιων χαρακτηριστικών του στοιχείου που είναι απαραίτητα για να περικλείουν επιπρόσθετες εννοιολογικές πληροφορίες, όπως rooting. Τα ονόματα των στοιχείων ρίζας πρέπει να προσδιορίζουν το περιεχόμενό τους.

6.1.3 Μοντελοποίηση των πινάκων

Έχοντας προσδιορίσει το στοιχείο ρίζας το επόμενο βήμα είναι να μοντελοποιήσουμε τους πίνακες τους οποίους έχουμε επιλέξει να συμπεριλάβουμε στο XML έγγραφο.

1) Πίνακες περιεχομένου : περιέχουν σύνολο εγγραφών π.χ. όλες οι διευθύνσεις πελατών μίας εταιρίας.
2) Πίνακες ευρετήρια : περιέχουν μια λίστα ζευγαριών ID περιγραφές που χρησιμοποιούνται για την περαιτέρω ταξινόμηση των πληροφοριών σε μια συγκεκριμένη σειρά ενός πίνακα, αποθηκεύοντας μια περιγραφή για κάθε ID του πίνακα περιεχομένων. Πίνακες όπως ο ShipMethod στο παράδειγμα μας είναι πίνακας ευρετήριο.

Για κάθε πίνακα περιεχομένου που έχουμε επιλέξει θα χρειαστεί να δημιουργούμε ένα στοιχείο στο DTD έγγραφο. Εφαρμόζοντας αυτό τον κανόνα στο παράδειγμά μας θα προσθέσουμε τα < Invoice > , < Part > , < MonthlyTotal > και άλλα στοιχεία στο DTD.

```
< ! ELEMENT SalesData EMPTY >

< ! ATTLIST SalesData

     Status (NewVersion | UpdatedVersion | CourtesyCopy) # REQUIERED >
```

```
< ! ELEMENT Invoice EMPTY >
< ! ELEMENT Customer EMPTY >
< ! ELEMENT Part EMPTY >
< ! ELEMENT MonthlyTotal EMPTY >
< ! ELEMENTMonthlyCustomerTotal EMPTY >
< ! ELEMENT MonthlyPartTotal EMPTY >
< ! ELEMENT LineItem EMPTY >
```

ΚΑΝΟΝΑΣ 3

Δημιουργία στοιχείου στο DTD για κάθε πίνακα περιεχομένου που έχουμε επιλέξει να μοντελοποιήσουμε. Δήλωση αυτών των στοιχείων ως empty για τη στιγμή.

6.1.4 Μοντελοποίηση των στηλών μη ξένων κλειδιών

Χρησιμοποιώντας αυτόν τον κανόνα θα δημιουργήσουμε χαρακτηριστικά στα στοιχεία τα οποία έχουμε ήδη καθορίσει να περιέχουν τις τιμές των στηλών της βάσης δεδομένων. Στο DTD αυτά τα χαρακτηριστικά εμφανίζονται στη δήλωση

!ATTLIST του στοιχείου που ανταποκρίνεται στον πίνακα στον οποίο εμφανίζονται οι στήλες. Εάν μία στήλη είναι εξωτερικό ξένο κλειδί σ' άλλον πίνακα να μην περιληφθεί σ' αυτό τον κανόνα – θα χειριστούμε τις στήλες εξωτερικών κλειδιών αργότερα όταν θα μοντελοποιήσουμε τις σχέσεις μεταξύ των στοιχείων που έχουμε δημιουργήσει.

Δήλωση κάθε χαρακτηριστικού που δημιουργείται με αυτό τον τρόπο με CDATA. Αν η στήλη ορίζεται στην ΒΔ να περιέχει τιμές Null τότε το αντίστοιχο χαρακτηριστικό είναι #REQUIRED, αλλιώς #IMPLIED.

Υπάρχουν τέσσερις επιλογές εδώ:

- `#FIXED` σημαίνει ότι το DTD παρέχει την τιμή
- `#REQUIRED` ότι πρέπει να εμφανίζεται στο έγγραφο
- `#IMPLIED` ότι μπορεί να εμφανιστεί ή όχι.

Τελικά μια τέτοια τιμή σημαίνει ότι ο επεξεργαστής πρέπει να υποκαταστήσει τη τιμή του χαρακτηριστικού εάν δεν περιέχεται στο έγγραφο, #IMPLIED είναι ο μόνος τρόπος νόμιμα να αφήσουμε κενή μια τιμή ενός χαρακτηριστικού.

Αν επιλέξουμε να αποθηκεύσουμε τις τιμές των στηλών του πίνακα ως περιεχόμενα των στοιχείων παρά σαν χαρακτηριστικά μπορούμε να έχουμε την ίδια προσέγγιση – δημιουργία ενός στοιχείου για κάθε για κάθε μονάδα δεδομένων και πρόσθεσή του στη λίστα περιεχομένου του στοιχείου του πίνακα στον οποίο εμφανίζεται η στήλη. Μη χρησιμοποίηση κατάληξης αν η στήλη δεν επιτρέπει nulls ή προαιρετικό πρόθεμα. Να γνωρίζεται ότι αν επιλεχθεί αυτή η προσέγγιση θα χρειαστεί να αναζητήσουμε πιθανές συγκρούσεις ονομάτων μεταξύ στηλών με το ίδιο όνομα σε διαφορετικούς πίνακες.

Δε τίθεται τέτοιο θέμα όταν χρησιμοποιούνται χαρακτηριστικά.

Επισκόπηση :

Επιτρέπεται η χρήση κενών;	**Στοιχεία**	**Χαρακτηριστικά**
Ναι	Κατάληξη με ?	Δήλωση ως #IMPLIED
Όχι	Χωρίς πρόθεμα	Δήλωση ως #REQUIRED

Για το παράδειγμά μας θυμηθείτε ότι θέλουμε να κρατήσουμε όλες τις στήλες μη ξένων κλειδιών με εξαίρεση τα system generated πρωτεύοντα κλειδιά.

```
<!ELEMENT SalesData EMPTY>
<!ATTLIST SalesData
  Status (NewVersion | UpdatedVersion | CourtesyCopy) #REQUIRED>
<!ELEMENT Invoice EMPTY>
<!ATTLIST Invoice
  InvoiceNumber CDATA #REQUIRED
  TrackingNumber CDATA #REQUIRED
  OrderDate CDATA #REQUIRED
  ShipDate CDATA #REQUIRED>
<!ELEMENT Customer EMPTY>
<!ATTLIST Customer
  Name CDATA #REQUIRED
  Address CDATA #REQUIRED
  City CDATA #REQUIRED
  State CDATA #REQUIRED
  PostalCode CDATA #REQUIRED>
<!ELEMENT Part EMPTY>
<!ATTLIST Part
  PartNumber CDATA #REQUIRED
  Name CDATA #REQUIRED
  Color CDATA #REQUIRED
  Size CDATA #REQUIRED>
<!ELEMENT MonthlyTotal EMPTY>
<!ATTLIST MonthlyTotal
  Month CDATA #REQUIRED
  Year CDATA #REQUIRED
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED>
<!ELEMENT MonthlyCustomerTotal EMPTY>
<!ATTLIST MonthlyCustomerTotal
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED>
<!ELEMENT MonthlyPartTotal EMPTY>
<!ATTLIST MonthlyPartTotal
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED>
<!ELEMENT LineItem EMPTY>
<!ATTLIST LineItem
  Quantity CDATA #REQUIRED
  Price CDATA #REQUIRED>
```

Σημειώστε ότι αφήσαμε απ' έξω τα Month ,Year απ' τις δομές < MonthlyPartTotal > και < MonthlySummaryTotal> από τη στιγμή που θα υπαγορευθούν απ' το στοιχείο < MonthlyTotal > που σχετίζεται μ' αυτά.

ΚΑΝΟΝΑΣ 4

Μοντελοποιώντας στήλες μη ξένα κλειδιά.

Δημιουργούμε χαρακτηριστικό για κάθε στήλη που έχουμε επιλέξει να συμπεριλάβουμε στο XML έγγραφο (εκτός foreign key columns). Αυτά τα χαρακτηριστικά θα πρέπει να εμφανιστούν στη δήλωση ! ATTLIST του στοιχείου που ανταποκρίνεται στον πίνακα στον οποίο εμφανίζονται. Δήλωση αυτών των χαρακτηριστικών ως CDATA και ακολούθως δήλωση ως #IMPLIED ή #REQUIRED αναλόγως αν η αρχική στήλη επιτρέπει κενές τιμές ή όχι.

6.1.5 Προσθέτοντας ID χαρακτηριστικά

Το επόμενο βήμα είναι να δημιουργήσουμε ένα ID για κάθε ένα από τα δομικά στοιχεία που έχουμε ορίσει μέχρι τώρα στην XML ΒΔ (με εξαίρεση το στοιχείο ρίζα.).

Αυτό για τη μοναδική αναγνώριση στοιχείων που χρειάζεται ν' αναφέρονται σ' αυτό άλλα στοιχεία.

Για το όνομα του χαρακτηριστικού χρησιμοποιούμε το όνομα του στοιχείου ακολουθούμενο από το ID. Αυτό υπάρχει περίπτωση να προκαλέσει ονοματολογικές συγκρούσεις μ' άλλα χαρακτηριστικά που έχουν ήδη προστεθεί στο XML και που τότε πρέπει να αλλάξουμε τα ονόματά. Αυτά μπορούν να ορισθούν ως τύπου ID και πρέπει να δηλωθούν ως #REQUIRED.

Όταν δημιουργούνται αυτές οι δομές, ένα μοναδικό ID θα πρέπει να δημιουργηθεί για κάθε περίπτωση (instance) ενός στοιχείου που γενικεύεται. Πρέπει να εξασφαλίσουμε ότι αυτά τα ID θα είναι μοναδικά όχι μόνο μεταξύ των όλων στοιχείων ενός συγκεκριμένου τύπου , αλλά μεταξύ όλων των στοιχείων μέσα στο έγγραφο. Ένας τρόπος για να γίνει αυτό με προγραμματισμό είναι να χρησιμοποιήσουμε το πρωτεύων κλειδί για την γραμμή που δημιουργείται βάζοντας σαν πρόθεμα το όνομα του πίνακα που εμφανίζεται για παράδειγμα για το πελάτη στη ΒΔ μας με ID 17 μπορούμε να χρησιμοποιήσουμε το string Customer17 για την τιμή του χαρακτηριστικού CustomerID του στοιχείου Customer. Αν έχουμε μη αριθμητικά κλειδιά στη ΒΔ μας , ή παρόμοια ονόματα πινάκων με αριθμητικές καταλήξεις (όπως Customer και Customer1) μπορεί να προκληθούν ονοματικές συγκρούσεις.

```
<!ELEMENT SalesData EMPTY>
<!ATTLIST SalesData
  Status (NewVersion | UpdatedVersion | CourtesyCopy) #REQUIRED>
<!ELEMENT Invoice EMPTY>
<!ATTLIST Invoice
  InvoiceID ID #REQUIRED
  InvoiceNumber CDATA #REQUIRED
  TrackingNumber CDATA #REQUIRED
  OrderDate CDATA #REQUIRED
  ShipDate CDATA #REQUIRED>
<!ELEMENT Customer EMPTY>
<!ATTLIST Customer
  CustomerID ID #REQUIRED
  Name CDATA #REQUIRED
  Address CDATA #REQUIRED
  City CDATA #REQUIRED
  State CDATA #REQUIRED
  PostalCode CDATA #REQUIRED>
<!ELEMENT Part EMPTY>
<!ATTLIST Part
  PartID ID #REQUIRED
  PartNumber CDATA #REQUIRED
  Name CDATA #REQUIRED
  Color CDATA #REQUIRED
  Size CDATA #REQUIRED>
<!ELEMENT MonthlyTotal EMPTY>
<!ATTLIST MonthlyTotal
  MonthlyTotalID ID #REQUIRED
  Month CDATA #REQUIRED
  Year CDATA #REQUIRED
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED>
<!ELEMENT MonthlyCustomerTotal EMPTY>
<!ATTLIST MonthlyCustomerTotal
  MonthlyCustomerTotalID ID #REQUIRED
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED>
<!ELEMENT MonthlyPartTotal EMPTY>
<!ATTLIST MonthlyPartTotal
  MonthlyPartTotalID ID #REQUIRED
```

```
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED>
<!ELEMENT LineItem EMPTY>
<!ATTLIST LineItem
  LineItemID ID #REQUIRED
  Quantity CDATA #REQUIRED
  Price CDATA #REQUIRED>
```

ΚΑΝΟΝΑΣ 5

Πρόσθεση ID χαρακτηριστικών στα στοιχεία που έχουμε δημιουργήσει στη δομή XML (εκτός στοιχείου ρίζας).

Χρησιμοποίηση του ονόματος του στοιχείου ακολουθούμενο από το ID για το όνομα του νέου χαρακτηριστικού. Δήλωση ως τύπου ID και #REQUIRED.

Διαχείριση ξένων κλειδιών

Σε δομές ΣΒΔ ο μόνος τρόπος για να δείξεις μια σχέση μεταξύ δεδομένων αποθηκευμένων σε διαφορετικούς πίνακες είναι μέσω ενός εξωτερικού κλειδιού. Υπάρχουν δύο τρόποι για να δείξεις αυτή τη σχέση στην XML. Μπορούμε να δημιουργήσουμε ιεραρχικές δομές οι οποίες μας επιτρέπουν να χρησιμοποιήσουμε containment για να δείξουμε σχέσεις μεταξύ δεδομένων (στο οποίο σχετικές πληροφορίες φωλιάζονται στο στοιχείο γονιός). Εναλλακτικά αν θέλουμε να κρατήσουμε ξεχωριστά τις XML δομές όπως οι πίνακες μιας ΒΔ μπορούμε να χρησιμοποιήσουμε ένα ID για να δείξουμε σε μία αντίστοιχη δομή η οποία έχει ένα IDREF χαρακτηριστικό.

Κάθε τρόπος έχει τα πλεονεκτήματα και τα μειονεκτήματά του. Οι δείκτες είναι πιο ευέλικτοι απ' την περιεκτικότητα, αλλά οι σχέσεις με δείκτες μπορούν να οδηγηθούν μόνο προς μία κατεύθυνση απ' τον επεξεργαστή και τείνουν να είναι πιο αργοί απ' το να χειρίζεσαι σχέσεις γονιός-παιδί.

Το επόμενο πράγμα που πρέπει να αποφασίσουμε είναι το αν θα χρησιμοποιήσουμε containment ή pointing για να αναπαραστήσουμε τις συσχετίσεις μεταξύ των πινάκων. Επιπρόσθετα πρέπει να προσθέσουμε τα απαριθμημένα χαρακτηριστικά που ανταποκρίνονται στους πίνακα – ευρετήρια που χρησιμοποιούμε.

Πρόσθεση απαριθμημένων χαρακτηριστικών σε πίνακες – ευρετήρια.

Αν έχουμε ένα εξωτερικό κλειδί σ' ένα πίνακα το οποίο δείχνει σ' ένα πίνακα ευρετήριο πρέπει να προσθέσουμε ένα απαριθμημένο χαρακτηριστικό στο στοιχείο που αναπαριστά τον πίνακα στον οποίο εμφανίζεται το εξωτερικό κλειδί.

Πριν να το εφαρμόσουμε αυτό στο παράδειγμά μας πρέπει να αναγνωρίσουμε τη φύση των σχέσεων μεταξύ των πινάκων που έχουμε επιλέξει να συμπεριλάβουμε στις δομές XML.

Για κάθε σχέση πρέπει να αναγνωρίσουμε :

1) Αν η σχέση είναι ευρετηρίου ή περιεχομένου.
2) Αν η σχέση είναι περιεχομένου και αν ναι την κατεύθυνση στην οποία θα οδηγηθεί.

Αυτό είναι σημαντικό σε μεγαλύτερες δομές γιατί κάποιες σχέσεις μπορούν να πλοηγηθούν σε περισσότερες από μια κατευθύνσεις. Σαν γενικότερος κανόνας οι σχέσεις πρέπει να κατευθύνονται στην ίδια πορεία στην οποία πιο συχνά τις οδηγεί το πρόγραμμα. Για παράδειγμα στην περίπτωσή μας μάλλον θα κατευθυνθούμε από το Invoice στο LineItem.

Πρέπει να προσδιορίσουμε την κατεύθυνση ανάμεσα στα στοιχεία γιατί έτσι καθορίζεται που θα είναι οι σχέσεις ID- IDREF. Θυμηθείτε ότι αυτά είναι χωρίς κατεύθυνση είναι

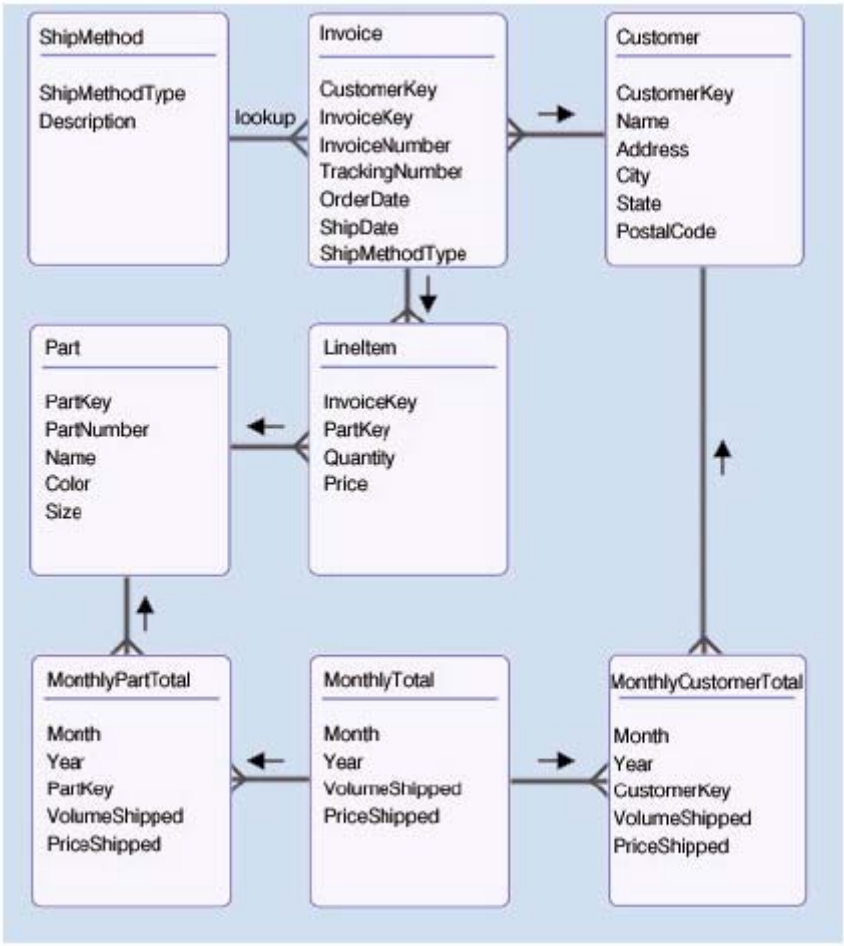
σχετικά εύκολο να διασχίσεις μια δομή από μια τιμή IDREF σε μία ID αλλά όχι και το αντίστροφο.

Αποφασίζοντας πως θα διασχίσουμε κανονικά τις δομές μας βοηθάει να καθορίσουμε τον σχεδιασμό τους.

Αν χρειαστεί να πλοηγηθούμε μεταξύ δύο στοιχείων ανεξαρτήτου κατεύθυνσης ίσως να χρειαστεί να προσθέσουμε ένα IDREF στοιχείο σε κάθε στοιχείο το οποίο θα δείχνει πίσω στο άλλο στοιχείο της σχέσεις. Παρόλα αυτά έτσι θα αυξηθεί ο χρόνος δημιουργίας του εγγράφου και το μέγεθός του.

Από τη στιγμή που θέλουμε το έγγραφό μας να υποστηρίζει τιμολόγια και πληροφορίες μηνιαίων αθροισμάτων συμπεραίνουμε ότι θέλουμε οι σχέσεις μας να μας παρέχουν ένα τρόπο πλοήγησης από τα τιμολόγια στις συσχετιζόμενες πληροφορίες. Για παράδειγμα θέλουμε να μπορούμε να πάμε από ένα τιμολόγιο στις γραμμές αντικειμένων του, στο κομμάτι που σχετίζεται με κάθε γραμμή αντικειμένου ή από ένα τιμολόγιο στον πελάτη ο οποίος το παρήγγειλε. Υπό άλλες συνθήκες μπορεί να διατάξουμε τις σχέσεις μας διαφορετικά για παράδειγμα εάν θέλαμε ένα XML έγγραφο που να επικεντρώνει στους πελάτες.

Αφού προσδιορίσουμε όλες τις συσχετίσεις στην δομή μας μπορούμε να συμπεράνουμε ότι η πλοήγηση μεταξύ των πινάκων θα μοιάζει κάπως έτσι



Έχουμε ένα εξωτερικό κλειδί με όνομα ShipMethodType το οποίο δείχνει σ' ένα πίνακα με όνομα ShipMethod, γι' αυτό το λόγο χρειάζεται να προσθέσουμε μια αριθμημένη τιμή για το ShipMethod του στοιχείου Invoice. Ας υποθέσουμε ότι στη ΒΔ μας ο πίνακας ShipMethod περιέχει τις ακόλουθες τιμές.

ShipMethod	Description
1	US Postal Service
2	Federal Express
3	UPS

Το αριθμημένο χαρακτηριστικό θα πάρει το όνομα του πίνακα ευρετηρίου και θα πρέπει να δηλωθεί ως #REQUIRED αν το εξωτερικό κλειδί δεν επιτρέπει τιμές Null ή #IMPLIED.

Ο καθορισμός των επιτρεπόμενων τιμών για την αρίθμηση είναι μια υποκειμενική διαδικασία και θα εξαρτηθεί από άλλους σχεδιαστικούς περιορισμούς (όπως το μέγεθος).οι τιμές που επιτρέπονται πρέπει τυπικά να είναι από ανθρώπους εκδόσεις τις περιγραφής για κάθε εγγραφή.

Έτσι δημιουργώντας ένα χαρακτηριστικό με επιτρεπόμενες αριθμημένες τιμές για τρεις πιθανές τιμές αναζήτησης κα προσθέτοντας το χαρακτηριστικό στοιχείο Invoice παίρνουμε από αυτό

```
...
<!ELEMENT Invoice EMPTY>
<!ATTLIST Invoice
  InvoiceID ID #REQUIRED
  InvoiceNumber CDATA #REQUIRED
  TrackingNumber CDATA #REQUIRED
  OrderDate CDATA #REQUIRED
  ShipDate CDATA #REQUIRED>
  ShipMethod (USPS | FedEx | UPS) #REQUIRED>
<!ELEMENT Customer EMPTY>
...
```

ΚΑΝΟΝΑΣ 6

Αναπαριστώντας πίνακες ευρετήρια.

Για κάθε εξωτερικό κλειδί το οποίο έχουμε επιλέξει να συμπεριλάβουμε στη δομή XML και σχετίζεται μ’ ένα πίνακα ευρετήριο.

1) Δημιουργούμε ένα χαρακτηριστικό στο στοιχείο που αναπαριστά τον πίνακα στον οποίο βρίσκεται το εξωτερικό κλειδί
2) Δίνουμε στο χαρακτηριστικό το ίδιο όνομα με το πίνακα και κάνουμε #REQUIREDαν το εξωτερικό κλειδί δεν επιτρέπει Null αλλιώς #IMPLIED.
3) Δημιουργία του χαρακτηριστικού της αριθμημένης λίστας. Οι επιτρεπόμενες τιμές πρέπει να είναι σε αναγνώσιμη από ανθρώπους μορφή στη στήλη περιγραφής για όλες τις γραμμές του πίνακα.

6.1.6 Πρόσθεση στοιχείου περιεχομένου στο στοιχείο ρίζα

Όταν δημιουργήσαμε το DTD και προσθέσαμε στοιχεία παιδιά για πίνακες, δε καθορίσαμε τα μοντέλα περιεχομένου των στοιχείων στο DTD.

Ο επόμενος κανόνας είναι να προσθέσουμε το μοντέλο περιεχομένου για το στοιχείο ρίζα στο DTD. Θα πρέπει να προσθέσουμε στοιχεία περιεχομένου που είναι κατάλληλα με τον τύπο πληροφορίας που θέλουμε να επεξεργαστούμε στα έγγραφά μας.

Για το παράδειγμά μας αποφασίσαμε ότι οι βασικές ιδέες που θέλουμε να μεταφέρουμε σχετίζονται με το Invoice και το MonthlyTotal. Όταν προσθέτουμε στοιχεία που αντιπροσωπεύουν αυτό το περιεχόμενο ως επιτρεπτά στοιχεία περιεχομένου για το στοιχείο ρίζα παίρνουμε το ακόλουθο:

```
<!ELEMENT SalesData (Invoice*, MonthlyTotal*)>
<!ATTLIST SalesData
  Status (NewVersion | UpdatedVersion | CourtesyCopy) #REQUIRED>
<!ELEMENT Invoice EMPTY>
```

ΚΑΝΟΝΑΣ 7

Προσθέτοντας στοιχεία περιεχομένου στο στοιχείο ρίζα.

Πρόσθεση ενός στοιχείου-παιδί στο επιτρεπτό περιεχόμενο του στοιχείου ρίζα για κάθε πίνακα που διαμορφώνει τον τύπο πληροφορίας που θέλουμε να αναπαραστήσουμε στο έγγραφό μας.

6.1.7 Επεκτείνοντας τις συσχετίσεις

Πρέπει να επεκτείνουμε τις σχέσεις μεταξύ πινάκων έτσι ώστε να προσθέσουμε στοιχεία περιεχομένου ή ζευγάρια IDREF με τον κατάλληλο τρόπο. Η διαδικασία αυτή είναι παρόμοια της διαδικασίας διάσχισης δένδρου δεδομένων – πλοηγούμαστε σε κάθε σχέση μετά σε σχέση των παιδιών της προηγούμενης και τα λοιπά μέχρι να έχουν επισκεφτεί όλες οι σχέσεις που περιέχονται στο υποσύνολο πινάκων που έχουμε επιλέξει να συμπεριλάβουμε στο XML έγγραφο. Πάλι όταν έχουμε να επιλέξουμε ποια κατεύθυνση θα ακολουθήσει μία σχέση επιλέγουμε την πιο λογική. Π.χ. θα πηγαίναμε μάλλον πιο συχνά από το Invoice στο LineItem απ' ότι αντίθετα. Από την άλλη πλευρά μια σχέση όπως Customer και Invoice μπορεί να χρειάζονται για να πηγαίνει προς τις δύο κατευθύνσεις το ίδιο συχνά καθιστώντας έτσι την σχέση διπλοκατευθυντική. Πρέπει να καθορίσουμε την κατεύθυνση σύμφωνα με την οποία θα οδηγούνται οι σχέσεις έτσι ώστε να αποφασίσουμε ποια δεικτοδότηση ή containment θα πρέπει να χρησιμοποιηθεί για να αναπαραστήσουμε τις σχέσεις.

Σχέση	**Αντίστοιχος τελεστής πολλαπλότητας**
Ένα προς ένα	?
Ένα προς πολλά	*

Για το DTD παράδειγμα προσθέτοντας τις σχέσεις ανασχέτησης παίρνουμε τα' ακόλουθο:

```
<!ELEMENT SalesData (Invoice*, MonthlyTotal*)>
<!ATTLIST SalesData
  Status (NewVersion | UpdatedVersion | CourtesyCopy) #REQUIRED>
<!ELEMENT Invoice (LineItem*)>
<!ATTLIST Invoice
  InvoiceID ID #REQUIRED
  InvoiceNumber CDATA #REQUIRED
  TrackingNumber CDATA #REQUIRED
  OrderDate CDATA #REQUIRED
  ShipDate CDATA #REQUIRED>
  ShipMethod (USPS | FedEx | UPS) #REQUIRED>
...
<!ELEMENT MonthlyTotal (MonthlyCustomerTotal*, MonthlyPartTotal*)>
<!ATTLIST MonthlyTotal
  MonthlyTotalID ID #REQUIRED
  Month CDATA #REQUIRED
  Year CDATA #REQUIRED
  VolumeShipped CDATA #REQUIRED
```

```
  PriceShipped CDATA #REQUIRED>
...
```

ΚΑΝΟΝΑΣ 8

Πρόσθεση σχέσεων μέσω περιεκτικότητας.

Για κάθε σχέση που έχουμε καθορίσει ,(ένα προς ένα και ένα προς πολλά) και καμία άλλη δεν οδηγεί στα παιδιά του επιλεγμένου υποσυνόλου, προσθέτουμε το στοιχείο παιδί ως στοιχείο περιεχομένου του στοιχείου γονιός με την κατάλληλη προτεραιότητα.

Πολλά προς ένα ή πολλαπλές σχέσεις γονιών

Εάν η σχέση είναι πολλά προς ένα ή το παιδί έχει περισσότερους από ένα γονιό τότε πρέπει να χρησιμοποιήσουμε δεικτοδότηση για να περιγράψουμε τη σχέση. Αυτό γίνεται με την προσθήκη IDREF χαρακτηριστικού στο στοιχείο της σχέσης απ' την πλευρά του γονιού. Το IDREF πρέπει να δείχνει στο ID του στοιχείου παιδί.

Εάν η σχέση είναι ένα προς πολλά και το παιδί έχει περισσότερους από ένα γονιούς πρέπει να χρησιμοποιήσουμε το IDREFS.

Σημειώστε ότι αν έχουμε καθορίσει μια σχέση προς τη μία ή την άλλη κατεύθυνση τότε για τους σκοπούς της ανάλυσης μετράει δύο διαφορετικές σχέσεις.

Αυτοί οι κανόνες δίνουν έμφαση στη χρήση περιεκτικότητας έναντι δεικτοδότησης όποτε αυτό είναι εφικτό. Λόγο των έμφυτων μειονεκτημάτων απόδοσης κατά την χρήση του DOM και SAXμε δεικτικές ή περιεκτικότητα είναι συνήθως η προτιμώμενη λύση. Εάν υπάρχει κατάσταση που να χρειάζεται δεικτοδότηση αλλά παρόλα αυτά η παρουσία της στις δομές μας προκαλεί καθυστέρηση ίσως σκεφτούμε να την αλλάξουμε με περιεκτικότητα και να επαναλάβουμε την δεικτοδοτούμενη πληροφορία εκεί που παρουσιαζόταν πριν.

```
<!ELEMENT SalesData (Invoice*, MonthlyTotal*)>
<!ATTLIST SalesData
  Status (NewVersion | UpdatedVersion | CourtesyCopy) #REQUIRED>
<!ELEMENT Invoice (LineItem*)>
<!ATTLIST Invoice
  InvoiceID ID #REQUIRED
  InvoiceNumber CDATA #REQUIRED
  TrackingNumber CDATA #REQUIRED
  OrderDate CDATA #REQUIRED
  ShipDate CDATA #REQUIRED
  ShipMethod (USPS | FedEx | UPS) #REQUIRED
  CustomerIDREF IDREF #REQUIRED>
<!ELEMENT Customer EMPTY>
...
<!ELEMENT MonthlyCustomerTotal EMPTY>
<!ATTLIST MonthlyCustomerTotal
  MonthlyCustomerTotalID ID #REQUIRED
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED
  CustomerIDREF IDREF #REQUIRED>
<!ELEMENT MonthlyPartTotal EMPTY>
<!ATTLIST MonthlyPartTotal
  MonthlyPartTotalID ID #REQUIRED
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED
  PartIDREF IDREF #REQUIRED>
<!ELEMENT LineItem EMPTY>
<!ATTLIST LineItem
  LineItemID ID #REQUIRED
  Quantity CDATA #REQUIRED
  Price CDATA #REQUIRED
  PartIDREF IDREF #REQUIRED>
```

ΚΑΝΟΝΑΣ 9

Πρόσθεση σχέσεων χρησιμοποιώντας IDREF / IDREFS.

Αναγνώριση κάθε σχέσης ως πολλά προς ένα προς την κατεύθυνση που την έχουμε ορίσει ή το παιδί της να είναι παιδί και σε άλλες σχέσεις. Για κάθε μια απ' αυτές τις σχέσεις πρόσθεση ενός IDREF ή IDREFS χαρακτηριστικού στο στοιχείο απ' την πλευρά του γονιού της σχέσης η οποία δείχνει στο ID του στοιχείου απ' την πλευρά του παιδιού της σχέσης.

Πρόσθεση χαμένων στοιχείων στο στοιχείο ρίζα

Μια σημαντική ατέλεια μπορεί να παρατηρηθεί στη τελική δομή που έχουμε φτάσει σ' αυτό το στάδιο όταν κατασκευάζουμε DTD δεν υπάρχει χώρος να προσθέσουμε ένα Customer στοιχείο. Δεν είναι το στοιχείο ρίζας του εγγράφου και δεν εμφανίζεται σε κανένα στοιχείο μοντέλου περιεχομένου των άλλων στοιχείων δομής. Αυτό γιατί είναι μόνο δείκτης δεν περιέχεται. Στοιχεία τα οποία αναφέρονται από το IDREF πρέπει να προστεθούν ως επιτρεπτά στοιχεία περιεχομένου στο στοιχείο ρίζα του DTD. Έπειτα όταν θα δημιουργούμε το έγγραφο τα ορφανά στοιχεία δημιουργούνται μέσα στο στοιχείο ρίζα και μετά δείχνουν όπου πρέπει. Εφαρμόζοντας αυτόν το κανόνα στο παράδειγμά μας βλέπουμε ότι μας λείπουν τα στοιχεία Customer και Part. Προσθέτοντας τα ως επιτρεπτό δομικό περιεχόμενο στο στοιχείο ρίζας παίρνουμε.

```
<!ELEMENT SalesData (Invoice*, Customer*, Part*, MonthlyTotal*)>
<!ATTLIST SalesData
  Status (NewVersion | UpdatedVersion | CourtesyCopy) #REQUIRED>
<!ELEMENT Invoice (LineItem*)>
...
```

ΚΑΝΟΝΑΣ 10

Πρόσθεση στοιχείων που λείπουν.

Για κάθε στοιχείο της δομής μας στο οποίο υπάρχει μόνο δείκτης προσθέτουμε αυτό ως επιτρεπτό στοιχείο περιεχομένου στο στοιχείο ρίζας. Θέτουμε την κατάληξη πληθυκότητας του προστιθέμενου στοιχείου σε *.

6.1.8 Απόρριψη μη αναφερόμενων ID χαρακτηριστικών

Τελικά χρειαζόμαστε να απαλείψουμε τα χαρακτηριστικά ID που δημιουργήσαμε αυτά στο κανόνα πέντε που δεν έχουν IDREF(S) να δείχνουν σ' αυτά. Απ' τη στιγμή που δημιουργήσαμε αυτά τα χαρακτηριστικά κατά την διαδικασία κατασκευής των XML δομών η απόρριψή τους αν δε χρησιμοποιούνται δεν θυσιάζει πληροφορία και γλιτώνει τους δημιουργούς απ' την παραγωγή μοναδικών τιμών για τα χαρακτηριστικά.

ΚΑΝΟΝΑΣ 11

Απομάκρυνση ανεπιθύμητων χαρακτηριστικών ID.

Απομάκρυνση χαρακτηριστικών ID που δεν αναφέρονται από IDREF ή IDREFS χαρακτηριστικά πουθενά μέσα στην XML δομή.

Εφαρμόζοντας αυτό τον κανόνα στο παραδείγματά μας παίρνουμε την τελική δομή. Τα χαρακτηριστικά InvoiceID, LineItemID, MonthlyPartTotalID, MonthlyTotalID και MonthlyCustomerTotalID δεν αναφέρονται από κανένα IDREF ή IDREFS

```
<!ELEMENT SalesData (Invoice*, Customer*, Part*, MonthlyTotal*)>
<!ATTLIST SalesData
  Status (NewVersion | UpdatedVersion | CourtesyCopy) #REQUIRED>
<!ELEMENT Invoice (LineItem*)>
<!ATTLIST Invoice
  InvoiceNumber CDATA #REQUIRED
  TrackingNumber CDATA #REQUIRED
  OrderDate CDATA #REQUIRED
```

```
  ShipDate CDATA #REQUIRED
  ShipMethod (USPS | FedEx | UPS) #REQUIRED
  CustomerIDREF IDREF #REQUIRED>
<!ELEMENT Customer EMPTY>
<!ATTLIST Customer
  CustomerID ID #REQUIRED
  Name CDATA #REQUIRED
  Address CDATA #REQUIRED
  City CDATA #REQUIRED
  State CDATA #REQUIRED
  PostalCode CDATA #REQUIRED>
<!ELEMENT Part EMPTY>
<!ATTLIST Part
  PartID ID #REQUIRED
  PartNumber CDATA #REQUIRED
  Name CDATA #REQUIRED
  Color CDATA #REQUIRED
  Size CDATA #REQUIRED>
<!ELEMENT MonthlyTotal (MonthlyCustomerTotal*, MonthlyPartTotal*)>
<!ATTLIST MonthlyTotal
  Month CDATA #REQUIRED
  Year CDATA #REQUIRED
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED>
<!ELEMENT MonthlyCustomerTotal EMPTY>
<!ATTLIST MonthlyCustomerTotal
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED
  CustomerIDREF IDREF #REQUIRED>
<!ELEMENT MonthlyPartTotal EMPTY>
<!ATTLIST MonthlyPartTotal
  VolumeShipped CDATA #REQUIRED
  PriceShipped CDATA #REQUIRED
  PartIDREF IDREF #REQUIRED>
<!ELEMENT LineItem EMPTY>
<!ATTLIST LineItem
  Quantity CDATA #REQUIRED
  Price CDATA #REQUIRED
 PartIDREF IDREF #REQUIRED>
```

Τελικά, ακολουθεί ένα παράδειγμα ενός XML εγγράφου που θα είναι έγκυρο γι' αυτό το DTD

```
<?xml version="1.0"?>
<!DOCTYPE SalesData SYSTEM "http://myserver/xmldb/ch03_ex01.dtd"
>
<SalesData Status="NewVersion">
   <Invoice InvoiceNumber="1"

TrackingNumber="1"

OrderDate="01012000"

ShipDate="07012000"

ShipMethod="FedEx"

CustomerIDREF="Customer2">
   <LineItem Quantity="2"
```

```
Price="5"

PartIDREF="Part2" />
   </Invoice>
   <Customer CustomerID="Customer2"

Name="BobSmith"

Address="2AnyStreet"

City="Anytown"

State="AS"

PostalCode="ANYCODE" />
   <Part PartID="Part2"
         PartNumber="13"
         Name="Winkle"
         Color="Red"
         Size="10" />
   <MonthlyTotal Month="January"

Year="2000"

VolumeShipped="2"

PriceShipped="10">
   <MonthlyCustomerTotal VolumeShipped="5"

PriceShipped="25"

CustomerIDREF="Customer2" />
   <MonthlyPartTotal VolumeShipped="8"

PriceShipped="40"

PartIDREF="Part2" />
   </MonthlyTotal>     </SalesData>
```

6.2 Η γλώσσα επερωτήσεων XQuery

Μερικές φορές είναι απαραίτητο να εξάγονται υποσύνολα δεδομένων αποθηκευμένα σ' ένα XML έγγραφο. Πληθώρα γλωσσών έχουν δημιουργηθεί για υποβολή ερωτήσεων σε XML έγγραφα. Όπως οι Lorel , Quilt , MnQL , Xduce , XML-QL, XPath, XQL, XQuery και YaTL.

Η XQuery είναι μια λειτουργική γλώσσα στην οποία κάθε ερώτημα είναι μία έκφραση. Οι εκφράσεις XQuery εμπίπτουν σε επτά ευρύς τύπους: εκφράσεις διαδρομής (path expressions), κατασκευαστές στοιχείων (element constructors), εκφράσεις FLWOR (FLWOR expressions), εκφράσεις που περιέχουν χειριστές και συναρτήσεις, υποθετικές εκφράσεις (conditional expressions), ποσοτικές εκφράσεις ή εκφράσεις που δοκιμάζουν ή μετατρέπουν τύπους δεδομένων. Η σύνταξη και η σημασιολογία των διαφορετικών ειδών των εκφράσεων XQuery διαφέρουν σημαντικά.

Η XQuery είναι ένα περίπλοκο σύστημα τύπων βασισμένο στους τύπους δεδομένων των XML schemas και υποστηρίζει τη διαχείριση των κόμβων του εγγράφου σ' αντίθεση με την XPath. Το XML έγγραφο που ακολουθεί (με όνομα books.xml) αποτελεί τη βάση στην οποία θα εφαρμοστούν τα XQUery επερωτήματα είναι το εξής:

```
<bib>
```

```
      <book year="1994">
            <title>TCP/IP Illustrated</title>
            <author><last>Stevens</last><first>W.</first></author>
            <publisher>Addison-Wesley</publisher>
            <price>65.95</price>
      </book>
      <book year="1992">
            <title>Advanced Programming in the UNIX Environment</title>
            <author><last>Stevens</last><first>W.</first></author>
            <publisher>Addison-Wesley</publisher>
            <price>65.95</price>
      </book>
      <book year="2000">
            <title>Data on the Web</title>
            <author><last>Abiteboul</last><first>Serge</first></author>
            <author><last>Buneman</last><first>Peter</first></author>
            <author><last>Suciu</last><first>Dan</first></author>
            <publisher>Morgan Kaufmann Publishers</publisher>
            <price>65.95</price>
      </book>
      <book year="1999">
            <title>The Economics of Technology and Content
            for Digital TV</title>
            <editor>
                  <last>Gerbarg</last>
                  <first>Darcy</first>
                  <affiliation>CITI</affiliation>
            </editor>
            <publisher>Kluwer Academic Publishers</publisher>
            <price>129.95</price>
      </book>
</bib>
```

6.2.1 Εντοπισμός κόμβων με χρήση της Xpath

Η XQuery, όπως και η γλώσσα XSLT, διαχειρίζεται τα XML έγγραφα ως δένδρα που αποτελούνται από κόμβους. Τα είδη των κόμβων που προκύπτουν είναι οι εξής: έγγραφο, στοιχείο, κείμενο, ιδιότητα, χώρος ονοματοδοσίας, οδηγίες επεξεργασίας, και σχόλιο (Περισσότερα για τους κόμβους βλ. κεφ 5). Η σύνταξη των δηλώσεων που χρησιμοποιούνται για τον εντοπισμό των κόμβων (*δηλώσεις διαδρομής*, *path expressions*) προέρχεται από τη σύνταξη της γλώσσας XPath (βλ. κεφ 5). Ένα απλό παράδειγμα χρήσης της XQuery για τον εντοπισμό των κόμβων είναι το εξής:

```
document("books.xml")/bib/book
```

Η παραπάνω δήλωση ανοίγει το έγγραφο `books.xml` χρησιμοποιώντας τη συνάρτηση `document()` (σε ορισμένες εφαρμογές αντί της συνάρτησης `document()` μπορεί να συναντήσετε τη συνάρτηση `doc()`), η οποία επιστρέφει τον κόμβο ρίζα του εγγράφου. Στο επόμενο βήμα δηλώνεται το `/bib` που επιλέγει το στοιχείο `bib` που βρίσκεται στην αρχή του εγγράφου και στη συνέχεια δηλώνεται το `/book` που επιστρέφει όλα τα στοιχεία

`book` που βρίσκονται μέσα στο στοιχείο `bib`. Τα στοιχεία `book` μπορούν να επιστραφούν χρησιμοποιώντας τους χαρακτήρες `//`, με βάση την παρακάτω δήλωση

```
document("books.xml")//book
```

6.2.2 Δημιουργία Κόμβων

Μέσω της γλώσσας XQuery παρέχεται η δυνατότητα δημιουργίας νέων κόμβων. Συγκεκριμένα, για τη δημιουργία των νέων στοιχείων, χρησιμοποιούνται οι αγκύλες οι οποίες περιέχουν δηλώσεις που εκτελούνται για τη δημιουργία ανοικτού περιεχομένου. Για παράδειγμα, το ακόλουθο επερώτημα μπορεί να χρησιμοποιηθεί για την δημιουργία ενός διαδραστικού οδηγού εκμάθησης της γλώσσας XQuery:

```
<example>
      <p> Here is a query. </p>
      <eg> document("books.xml")//book[1]/title </eg>
      <p> Here is the result of the above query.</p>
      <eg>{ document("books.xml")//book[1]/title }</eg>
</example>
```

Το αποτέλεσμα αυτού του επερωτήματος είναι το εξής:

```
<example>
      <p> Here is a query. </p>
      <eg> document("books.xml")//book[1]/title </eg>
      <p> Here is the result of the above query.</p>
      <eg><title>TCP/IP Illustrated</title></eg>
</example>
```

Οι δηλώσεις που περιέχονται στις αγκύλες επιτρέπουν τη δημιουργία νέων XML τιμών αναδομώντας υπάρχοντες XML τιμές. Το ακόλουθο επερώτημα δημιουργεί μια λίστα από τίτλους βιβλίων

```
<titles count="{ count(document ('books.xml')//title) }">
{
document ("books.xml")//title
}
</titles>
```

Το αποτέλεσμα ατού του επερωτήματος είναι το εξής:

```
<titles count = "4">
      <title>TCP/IP Illustrated</title>
      <title>Advanced Programming in the Unix Environment</title>
      <title>Data on the Web</title>
      <title>The Economics of Technology and Content for
      Digital TV</title>
</titles>
```

Θα πρέπει να σημειωθεί ότι η συνάρτηση `document("books.xml")` επιστρέφει τη κόμβο ρίζας του τρέχοντος εγγράφου.

6.2.3 Συνδυασμός και αναδόμηση κόμβων

Δηλώσεις FLWOR

Οι δηλώσεις FLWOR, είναι οι πιο συνηθισμένες και πιο ισχυρές δηλώσεις της γλώσσας XQUERY. Οι δηλώσεις αυτές είναι παρόμοιες με τις δηλώσεις `SELECT – FROM – WHERE` που χρησιμοποιούνται στη γλώσσα SQL για τη διαχείριση των βάσεων δεδομένων.

Αντίθετα με την SQL, όπου οι δηλώσεις καθορίζονται με βάση τους πίνακες, τις εγγραφές και τα χαρακτηριστικά τους, οι FLOWR δηλώσεις δεσμεύουν μεταβλητές με τιμές μέσω των `for` και των `let` συντακτικών μονάδων (clause), και χρησιμοποιούν αυτές τις αντιστοιχίες για τη δημιουργία νέων αποτελεσμάτων. Ένας συνδυασμός τέτοιων δεσμεύσεων ονομάζεται συστοιχία (tuple). Μια απλή FLWOR δήλωση που επιστρέφει τον τίτλο και την τιμή κάθε βιβλίου που εκδόθηκε τη χρονιά 2000 είναι η εξής:

```
for $b in document("books.xml")//book
where $b/@year= "2000"
return $b/title
```

Η δήλωση αυτή δεσμεύει τη μεταβλητή `$b` σε ένα βιβλίο κάθε φορά, δημιουργώντας μια σειρά από συστοιχίες. Κάθε συστοιχία περιέχει μια δέσμευση κατά την οποία η μεταβλητή `$b` αντιστοιχεί σε ένα συγκεκριμένο βιβλίο. Η συντακτική μονάδα where ελέγχει κάθε συστοιχία για το αν ισχύει η ισότητα `$b/@year="2000"`, ενώ η συντακτική μονάδα return υπολογίζεται για κάθε συστοιχία που ικανοποιούν τη συνθήκη που εκφράστηκε στη μονάδα `where`. Στην περίπτωσή μας, μόνο το βιβλίο *Data on the Web* εκδόθηκε τη χρονιά 2000, συνεπώς το αποτέλεσμα του παραπάνς επερωτήματος επιστρέφει το εξής:

```
<title> Data on the Web </title>
```

Τα αρχικά FLWOR προέρχονται από τα πρώτα γράμματα των συντακτικών μονάδων που χρησιμοποιούνται στις εκφράσεις FLWOR και είναι οι εξής:

- `for`: δεσμεύει μια ή περισσότερες μεταβλητές σε δηλώσεις,
- `let`: δεσμεύουν μεταβλητές σε ολόκληρο το αποτέλεσμα μιας δήλωσης, είτε προσθέτοντας αυτές τις δεσμεύσεις στις συστοιχίες που δημιουργούνται από τη μονάδα for, είτε δημιουργώντας μια απλή συστοιχία που να περιέχει αυτές τις δεσμεύσεις σε περίπτωση που δεν έχει οριστεί μια for συντακτική μονάδα.
- `where`: φιλτράρουν συστοιχίες, διατηρώντας μόνο αυτές που ικανοποιούν μια συνθήκη.
- `order by`: ταξινομούν τις συστοιχίες σε μια αλληλουχία συστοιχιών
- `return`: δημιουργούν το αποτέλεσμα μιας FLWOR δήλωσης για μια δωσμένη συστοιχία

Μια FLWOR δήλωση ξεκινάει με μια ή περισσότερες `for` και `let` μονάδες τοποθετημένες σε οποιαδήποτε σειρά, ακολουθούμενες από τις συντακτικές μονάδες `where`, `order by` και `return`. Η χρήση των `where` και `order` είναι προαιρετική ενώ η χρήση της return είναι υποχρεωτική.

Οι συντακτικές μονάδες for και let

Οι FLWOR δηλώσεις καθορίζονται από τις συστοιχίες οι οποίες δημιουργούνται μέσω των μονάδων `for` και `let`, συνεπώς κάθε FLWOR δήλωση πρέπει να αποτελείται από τουλάχιστον μια `for` ή `let` μονάδα. Τα παραδείγματα που ακολουθούν βοηθούν ώστε να γίνει κατανοητός ο τρόπος με τον οποίο δημιουργούνται οι συστοιχίες στις FLWOR δηλώσεις.

Στο επόμενο παράδειγμα ορίζεται ένα επερώτημα που δημιουργεί (μέσω της return) ένα στοιχείο που ονομάζεται `<tuple>` προκειμένου να εμφανίσει τις συστοιχίες που δημιουργούνται από ένα τέτοιο επερώτημα:

```
for $i in (1, 2, 3)
return
      <tuple><i>{ $i }</i></tuple>
```

Στο παράδειγμα αυτό, δεσμεύεται η μεταβλητή `$i` στη δήλωση `(1,2,3)`, που δημιουργεί μια αλληλουχία ακεραίων. Η XQuery παρέχει μια γενική σύνταξη, με αποτέλεσμα οι μονάδες `for` και `let` να δεσμεύουν οποιαδήποτε XQuery δήλωση. Ακολουθεί το

αποτέλεσμα του παραπάνω ερωτήματος, που δείχνει τον τρόπο με τον οποίο η μεταβλητή `$i` δεσμεύεται σε κάθε συστοιχία:

```
<tuple><i>1</i></tuple>
<tuple><i>2</i></tuple>
<tuple><i>3</i></tuple>
```

Η συντακτική μονάδα `let` δεσμεύει μια μεταβλητή σε ολόκληρο το αποτέλεσμα μιας δήλωσης. Αν δεν έχουν οριστεί for μονάδες, τότε δημιουργείται μια απλή συστοιχία, που περιέχει τις δεσμεύσεις μιας μεταβλητής όπως ορίστηκε μέσω της `let` μονάδας. Το ακόλουθο επερώτημα είναι ίδιο με το προηγούμενο, με τη διαφορά ότι χρησιμοποιείται η συντακτική μονάδα `let`:

```
let $i := (1, 2, 3)
return
      <tuple><i>{ $i }</i></tuple
```

Ακολουθεί το αποτέλεσμα του παραπάνω επερωτήματος κατά το οποίο η μεταβλητή `$i` δεσμεύεται σε ολόκληρη την αλληλουχία των ακεραίων:

```
<tuple><i>1 2 3</i></tuple>
```

Στο επερώτημα που ακολουθεί οι δηλώσεις που δεσμεύτηκαν μέσω της `let` μονάδας προστίθενται στις συστοιχίες που δημιουργήθηκαν μέσω της μονάδας `for`.

```
for $i in (1, 2, 3)
let $j := (1, 2, 3)
return
      <tuple><i>{ $i }</i><j>{ $j }</j></tuple>
```

Το αποτέλεσμα του παραπάνω επερωτήματος είναι το εξής:

```
<tuple><i>1</i><j>1 2 3</j></tuple>
<tuple><i>2</i><j>1 2 3</j></tuple>
<tuple><i>3</i><j>1 2 3</j></tuple>
```

Στο επερώτημα του ακολουθεί συνδυάζονται οι μονάδες `for` και `let` με τον ίδιο τρόπο που συνδυάστηκαν στο προηγούμενο επερώτημα:

```
for $b in document("books.xml")//book
let $c := $b/author
return <book>{ $b/title, <count>{ count($c) }</count>}</book>
```

Το ερώτημα αυτό εμφανίζει τον τίτλο του κάθε βιβλίου μαζί με τον αριθμό των συγγραφέων του. Το αποτέλεσμα αυτού του επερωτήματος είναι το εξής:

```
<book>
   <title>TCP/IP Illustrated</title>
   <count>1</count>
</book>
<book>
   <title>Advanced Programming in the UNIX Environment</title>
   <count>1</count>
</book>
<book>
   <title>Data on the Web</title>
   <count>3</count>
</book>
<book>
   <title>The Economics of Technology and Content for
Digital TV</title>
   <count>0</count>
</book>
```

Η συντακτική μονάδα where

Η συντακτική μονάδα `where` αποκλείει τις συστοιχίες αυτές που δεν ικανοποιούν συγκεκριμένες συνθήκες. Η μονάδα `return` εκτελείται μόνο για τις συστοιχίες που επιστρέφονται από τη μονάδα `where`. Το ακόλουθο επερώτημα επιστρέφει τα βιβλία των οποίων η τιμή δε ξεπερνάει τα € 50.00:

```
for $b in document("books.xml")//book
where $b/price < 50.00
return $b/title
```

Το αποτέλεσμα επερωτήματος αυτού είναι το εξής:

```
<title>Data on the Web</title>
```

Η μονάδα `for` μπορεί να περιέχει οποιαδήποτε δήλωση το αποτέλεσμα της οποίας, είναι μια Boolean τιμή. Το ακόλουθο ερώτημα επιστρέφει τον τίτλο των βιβλίων των οποίων οι συγγραφείς είναι παραπάνω από δύο:

```
for $b in document ("books.xml")//book
let $c := $b//author
where count($c) > 2
return $b/title
```

Το αποτέλεσμα της παραπάνω επερώτησης είναι το εξής:

```
<title>Data on the Web</title>
```

Η συντακτική μονάδα order by

Η συντακτική μονάδα `order by`, ταξινομεί τις συστοιχίες πριν την εκτέλεση της μονάδας return προκειμένου να αλλάξει η σειρά των αποτελεσμάτων. Για παράδειγμα, το ακόλουθο επερώτημα εμφανίζει τους τίτλους των βιβλίων σε αλφαβητική σειρά.

```
for $t in document("books.xml")//title
order by $t
return $t
```

Η μονάδα for παράγει μια αλληλουχία από συστοιχίες, όπου κάθε μια περιέχει τον κόμβο `title`. Η μονάδα `order by` ταξινομεί αυτές τις συστοιχίες με βάση την τιμή που περιέχει το στοιχείο `<title>` σε κάθε συστοιχία, ενώ η μονάδα `return` επιστρέφει τα στοιχεία `<title>` με την ίδια σειρά στην οποία βρίσκονται ταξινομημένες οι συστοιχίες. Το αποτέλεσμα αυτού του επερωτήματος είναι το εξής:

```
<title>Advanced Programming in the Unix Environment</title>
<title>Data on the Web</title>
<title>TCP/IP Illustrated</title>
<title>The Economics of Technology and Content for Digital TV</title>
```

Η συντακτική μονάδα return

Όμοια με τις συντακτικές μονάδες for και let, όπου επιτρέπουν τη δέσμευση μεταβλητών σε οποιαδήποτε δήλωση και με τη μονάδα where η οποία μπορεί να περιέχει οποιαδήποτε Boolean δήλωση, η μονάδα return μπορεί να περιέχει οποιαδήποτε δήλωση. Οι πιο συνηθισμένες δηλώσεις είναι αυτές που επιτρέπουν τη δημιουργία στοιχείων. Το παρακάτω επερώτημα χρησιμοποιεί μια τέτοια δήλωση για τη δημιουργία του στοιχείου <quote>, που περιγράφει τον τίτλο και την τιμή των βιβλίων.

```
for $b in document("books.xml")//book
return
      <quote>{ $b/title, $b/price }</quote>
```

Το αποτέλεσμα του παραπάνω επερωτήματος είναι το εξής

```
<quote>
```

```
      <title>TCP/IP Illustrated</title>
      <price>65.95</price>
</quote>
<quote>
      <title>Advanced Programming in the UNIX Environment</title>
      <price>65.95</price>
</quote>
<quote>
      <title>Data on the Web</title>
      <price>39.95</price>
</quote>
<quote>
      <title>The Economics of Technology and Content for Digital
      TV</title>
      <price>129.95</price>
</quote>
```

Δημιουργία ενώσεων

Σε ένα επερώτημα είναι δυνατόν να δεσμευτούν πολλές μεταβλητές, μέσω της συντακτικής μονάδας for, ώστε να συνδυαστεί η πληροφορία που προέρχεται από διαφορετικές δηλώσεις. Η διαδικασία αυτή ακολουθείται προκειμένου να συγκεντρωθεί πληροφορία από διαφορετικές πηγές πληροφορίας. Έστω, για παράδειγμα, ότι υπάρχει ένα αρχείο που ονομάζεται reviews.xml που περιγράφει σχόλια βιβλίων

```
<reviews>
      <entry>
            <title>TCP/IP Illustrated</title>
            <rating>5</rating>
            <remarks>Excellent technical content. Not much plot.</remarks>
      </entry>
</reviews>
```

Μέσω των FLWOR δηλώσεων είναι δυνατόν να δεσμευτεί μια μεταβλητή στα δεδομένα που περιγράφουν τη βιβλιογραφία και άλλη μια στα δεδομένα που περιγράφουν τα σχόλια, συγκρίνοντας έτσι δεδομένα και από τα δύο αρχεία και δημιουργώντας αποτελέσματα που συνδυάζουν την πληροφορία. Για παράδειγμα ένα επερώτημα θα μπορούσε να επιστρέψει τον τίτλο ενός βιβλίου και ότι σχόλια υπάρχουν για το βιβλίο αυτό.

Όταν περισσότερες από μια μεταβλητές δεσμεύονται μέσω μιας μονάδας for, τότε οι συστοιχίες που δημιουργούνται περιέχουν όλους τους δυνατούς συνδυασμούς των αντικειμένων στα οποία δεσμεύτηκαν αυτές οι μεταβλητές. Για παράδειγμα το ακόλουθο επερώτημα δείχνει όλους τους συνδυασμούς μεταξύ του συνόλου των αριθμών 1,2,3 και 4,5,6:

```
for $i in (1, 2, 3),
    $j in (4, 5, 6)
return
    <tuple><i>{ $i }</i><j>{ $j }</j></tuple>
```

Το αποτέλεσμα του παραπάνω επερωτήματος είναι το εξής:

```
<tuple><i>1</i><j>4</j></tuple>
<tuple><i>1</i><j>5</j></tuple>
<tuple><i>1</i><j>6</j></tuple>
<tuple><i>2</i><j>4</j></tuple>
<tuple><i>2</i><j>5</j></tuple>
<tuple><i>2</i><j>6</j></tuple>
```

```
<tuple><i>3</i><j>4</j></tuple>
<tuple><i>3</i><j>5</j></tuple>
<tuple><i>3</i><j>6</j></tuple>
```

Στην περίπτωση που μέσω της μονάδας `where` γίνεται επιλογή των συνδυασμών, η διαδικασία αυτή ονομάζεται *ένωση* (*join*). Το ακόλουθο επερώτημα δημιουργεί μια ένωση για το συνδυασμό δεδομένων που προέρχονται από τη βιβλιογραφία με δεδομένα που προέρχονται από το σύνολο των σχολίων:

```
for $t in doc("books.xml")//title,
    $e in doc("reviews.xml")//entry
where $t = $e/title
return
    <review>{ $t, $e/remarks }</review>
```

Το αποτέλεσμα του παραπάνω επερωτήματος είναι το εξής:

```
<review>
      <title>TCP/IP Illustrated</title>
      <remarks>Excellent technical content. Not much plot.</remarks>
</review>
```

Στο επερώτημα αυτό, μέσω της μονάδας `for` δημιουργούνται συστοιχίες που περιέχουν όλους του δυνατούς συνδυασμούς μεταξύ των δύο πηγών, ενώ μέσω της μονάδας for φιλτράρονται οι συστοιχίες που ο τίτλος ενός σχολίου δεν είναι ίδιος με τον τίτλο ενός βιβλίου. Τέλος, η μονάδα return δημιουργεί το αποτέλεσμα από τους εναπομείναντες κόμβους.

6.3 Ασκήσεις

Άσκηση 1

Με βάση το XML έγγραφο που ακολουθεί (το όνομα του οποίου είναι books.xml και βρίσκεται στη διεύθυνση http://bstore1.example.com/bib.xml):

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price> 65.95</price>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
    </book>
    <book year="2000">
        <title>Data on the Web</title>
        <author><last>Abiteboul</last><first>Serge</first></author>
        <author><last>Buneman</last><first>Peter</first></author>
```

```
        <author><last>Suciu</last><first>Dan</first></author>
        <publisher>Morgan Kaufmann Publishers</publisher>
        <price>39.95</price>
    </book>
    <book year="1999">
        <title>The Economics of Technology and Content for Digital
TV</title>
        <editor>
               <last>Gerbarg</last><first>Darcy</first>
                <affiliation>CITI</affiliation>
        </editor>
        <publisher>Kluwer Academic Publishers</publisher>
        <price>129.95</price>
    </book>
</bib>
```

Ζητείται

α) να σχεδιαστεί το DTD έγγραφο που μοντελοποιεί το παραπάνω έγγραφο

β) Να επιστραφούν τα βιβλία που έχουν εκδοθεί μετά από το 1991 από τον εκδοτικό οίκο Addison – Wesley

(γ) Να επιστραφούν όλα τα ζεύγη <title> - <author>, με κάθε ζεύγος να περιέχεται στο στοιχείο <Result >

(δ) Να επιστραφεί, σε αλφαβητική σειρά, ο τίτλος και η χρονιά όλων των βιβλίων που έχουν εκδοθεί μετά από το 1991 από τον εκδοτικό οίκο Addison – Wesley

Λύση

(α) Από το XML έγγραφο προκύπτει ότι τα στοιχεία <book> και <author> εμφανίζονται παραπάνω από μια φορά. Επίσης μπορούμε να συμπεράνουμε ότι το η χρήση όλων των στοιχείων είναι υποχρεωτική εκτός των στοιχείων <author> και <editor>. Το στοιχείο <author> εμφανίζεται στα δύο από τα τρία στοιχεία <book>, ενώ το στοιχείο <editor> εμφανίζεται μόνο στο τρίτο στοιχείο <book>. Το DTD που μοντελοποιεί το δοσμένο έγγραφο είναι το εξής:

```
<!ELEMENT bib  (book+ )>
<!ELEMENT book  (title,  (author* | editor ?), publisher, price )>
<!ATTLIST book  year CDATA  #REQUIRED >
<!ELEMENT author  (last, first )>
<!ELEMENT editor  (last, first, affiliation )>
<!ELEMENT title  (#PCDATA )>
<!ELEMENT last  (#PCDATA )>
<!ELEMENT first  (#PCDATA )>
<!ELEMENT affiliation  (#PCDATA )>
<!ELEMENT publisher  (#PCDATA )>
<!ELEMENT price  (#PCDATA )>
```

(β) Το επερώτημα που επιστρέφει τα βιβλία που έχουν εκδοθεί μετά από το 1991 από τον εκδοτικό οίκο Addison – Wesley θα έχει την εξής μορφή:

```
<bib>
 {
  for $b in document("http://bstore1.example.com/bib.xml")/bib/book
  where $b/publisher = "Addison-Wesley" and $b/@year > 1991
  return
    <book year="{ $b/@year }">
     { $b/title }
    </book>
 }
</bib>
```

Στο παράδειγμα αυτό χρησιμοποιείται ο Boolean τελεστής and προκειμένου η δομική μονάδα where να μπορεί να φιλτράρει τις συστοιχίες εκείνες που δεν ικανοποιούν τη συνθήκη. Το αποτέλεσμα είναι το εξής:

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
    </book>
</bib>
```

(γ) Το επερώτημα που απαιτείται έχει την εξής μορφή:

```
<results>
  {
    for $b in document("http://bstore1.example.com/bib.xml")/bib/book,
        $t in $b/title,
        $a in $b/author
    return
        <result>
            { $t }
            { $a }
        </result>
  }
</results>
```

Ένας δεύτερος τρόπος με τον οποίο μπορεί να συνταχτεί το ερώτημα αυτό είναι ο εξής:

```
<results>
{
```

```
    for $b in document("http://bstore1.example.com/bib.xml")/bib/book
    return
        <result>
            { $b/title }
            { $b/author  }
        </result>
}
</results>
```

Το αποτέλεσμα και των δύο επερωτημάτων αναμένεται να είναι το εξής:

```
<results>
    <result>
        <title>TCP/IP Illustrated</title>
        <author>
            <last>Stevens</last>
            <first>W.</first>
        </author>
    </result>
    <result>
        <title>Advanced Programming in the Unix environment</title>
        <author>
            <last>Stevens</last>
            <first>W.</first>
        </author>
    </result>
    <result>
        <title>Data on the Web</title>
        <author>
            <last>Abiteboul</last>
            <first>Serge</first>
        </author>
    </result>
    <result>
        <title>Data on the Web</title>
        <author>
            <last>Buneman</last>
            <first>Peter</first>
        </author>
    </result>
    <result>
        <title>Data on the Web</title>
```

```
        <author>
            <last>Suciu</last>
            <first>Dan</first>
        </author>
    </result>
</results>
```

(δ) Προκειμένου να εμφανιστούν τα αποτελέσματα σε αλφαβητική σειρά θα πρέπει να χρησιμοποιηθεί η δομική μονάδα order by. Συνεπώς το επερώτημα θα έχει την εξής μορφή:

```
<bib>
  {
    for $b in document("http://bstore1.example.com/books.xml")//book
    where $b/publisher = "Addison-Wesley" and $b/@year > 1991
    order by $b/title
    return
        <book>
            { $b/@year }
            { $b/title }
        </book>
  }
</bib>
```

Το αναμενόμενο αποτέλεσμα θα έχει την εξής μορφή:

```
<bib>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
    </book>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
    </book>
</bib>
```

Άσκηση 2

Με βάση το XML έγγραφο που ακολουθεί (το όνομα του οποίου είναι review.xml και βρίσκεται στη διεύθυνση http://bstore2.example.com/reviews.xml):

```
<reviews>
    <entry>
        <title>Data on the Web</title>
        <price>34.95</price>
        <review>
               A very good discussion of semi-structured database
               systems and XML.
        </review>
    </entry>
    <entry>
        <title>Advanced Programming in the Unix environment</title>
        <price>65.95</price>
        <review>
               A clear and detailed discussion of UNIX programming.
        </review>
    </entry>
    <entry>
        <title>TCP/IP Illustrated</title>
        <price>65.95</price>
        <review>
               One of the best books on TCP/IP.
        </review>
    </entry>
</reviews>
```

Ζητείται

α) να σχεδιαστεί το DTD έγγραφο που μοντελοποιεί το παραπάνω έγγραφο

β) Για κάθε βιβλίο που βρίσκεται στο έγγραφο bib.xml και reviews.xml να επιστραφεί ο τίτλος του βιβλίου και η τιμή του από κάθε μια πηγή

Λύση

Παρατηρούμε από το έγγραφο αυτό, ότι μόνο το στοιχείο <entry> είναι αυτό που επαναλαμβάνεται μέσα στο στοιχείο ρίζας του εγγράφου <reviews>. Συνεπώς το DTD που μοντελοποιεί το έγγραφο αυτό θα είναι το εξής:

```
<!ELEMENT reviews (entry*)>
<!ELEMENT entry   (title, price, review)>
<!ELEMENT title   (#PCDATA)>
<!ELEMENT price   (#PCDATA)>
```

```
<!ELEMENT review  (#PCDATA)>
```

(β) Το επερώτημα που απαιτείται θα πρέπει να έχει την εξής μορφή:

```
<books-with-prices>
  {
    for $b in document("http://bstore1.example.com/bib.xml")//book,
        $a in document("http://bstore2.example.com/reviews.xml")//entry
    where $b/title = $a/title
    return
        <book-with-prices>
            { $b/title }
            <price-bstore2>{ $a/price/text() }</price-bstore2>
            <price-bstore1>{ $b/price/text() }</price-bstore1>
        </book-with-prices>
  }
</books-with-prices>
```

Στο επερώτημα αυτό το στοιχείο <price-bstore2> αποθηκεύει την τιμή του βιβλίου που προέρχεται από το έγγραφο reviews.xml ενώ το στοιχείο <price-bstore1> αποθηκεύει την τιμή του βιβλίου που προέρχεται από το έγγραφο bib.xml. Το αναμενόμενο αποτέλεσμα θα έχει την εξής μορφή:

```
<books-with-prices>
    <book-with-prices>
        <title>TCP/IP Illustrated</title>
        <price-bstore2>65.95</price-bstore2>
        <price-bstore1>65.95</price-bstore1>
    </book-with-prices>
    <book-with-prices>
        <title>Advanced Programming in the Unix environment</title>
        <price-bstore2>65.95</price-bstore2>
        <price-bstore1>65.95</price-bstore1>
    </book-with-prices>
    <book-with-prices>
        <title>Data on the Web</title>
        <price-bstore2>34.95</price-bstore2>
        <price-bstore1>39.95</price-bstore1>
    </book-with-prices>
</books-with-prices>
```

Βιβλιογραφία

Professional XML Schemas, Jon Duckett,Oliver Griffin, Stephen Mohr, Francis Norton, Ian Stokes-Rees, Kevin Williams, Kurt Cagle, Nikola Ozu, Jeni Tennison, ISBN:1-861005-47-454999

Inside XML, Steven Holzner, ISBN: 0-7357-1020-1

Οδηγός της XML με παραδείγματα, Benolt Marchal, ISBN:960-387-060-9

Namespaces in XML, W3C Recommendation, Διαθέσιμο στην ηλεκτρονική διεύθυνση http://www.w3.org/TR/REC-xml-names/

XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001, Διαθέσιμο στη διεύθυνση http://www.w3.org/TR/xmlschema-0/

XML Schema Part 1: Structures, W3C Recommendation, 2 May 2001, Διαθέσιμο στη διεύθυνση http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Datatypes, W3C Recommendation, 2 May 2001, Διαθέσιμο στη διεύθυνση http://www.w3.org/TR/xmlschema-2/

XSL Transformations (XSLT), W3C Recommendation, 16 November 1999, Διαθέσιμο από την ηλεκτρονική διεύθυνση http://www.w3.org/TR/xslt