ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ

ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

XML Schema και Βάσεις Δεδομένων

ΝΙΚΑΣ ΒΑΣΙΛΕΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΚΕΡΑΜΟΠΟΥΛΟΣ ΕΥΚΛΕΙΔΗΣ

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ: 2008 - 2009

Περίληψη

Η ακαδημαϊκή αυτή εργασία αποτελεί μια πρώτη επαφή με την XML γενικότερα και με τα XML σχήματα ειδικότερα.

Αρχικά, στο πρώτο και δεύτερο κεφάλαιο γίνεται μια εισαγωγή στην XML και στις τεχνολογίες που την απαρτίζουν, όπως: Namespaces, XSL, XSLT, XPath, XQuery και ερχόμαστε για πρώτη φορά σε επαφή με ένα έγγραφο XML.

Μετά τους εισαγωγικούς ορισμούς φθάνουμε αισίως στο δεύτερο και κυρίως μέρος της εργασίας, στο οποίο ασχολούμαστε αποκλειστικά με το XML σχήμα και τα συστατικά του μέρη, όπως απλούς και σύνθετους τύπους, Namespaces, πρότυπα περιεχομένων κ.λ.π

Στο τρίτο μέρος της εργασίας αναφερόμαστε σε διάφορα εργαλεία σύνταξης XML σχημάτων και γενικότερα XML εγγράφων. Παρατίθενται τα White papers (Επίσημα Έγγραφα από τις ιστοσελίδες των εταιρειών) τριών εργαλείων: xmlDraft, oXygen και XMLFox.

Στο τέταρτο και τελευταίο μέρος της εργασίας γίνεται αναφορά στη σχέση των XML σχημάτων και Βάσεων Δεδομένων, πιο συγκεκριμένα για το Σύστημα Διαχείρισης Βάσεων IBM DB2. Δίνονται κάποια παραδείγματα εισαγωγής σχημάτων στη DB2 και πως αυτή χρησιμοποιεί και διαχειρίζεται τα σχήματα αυτά.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Σελ.

1. Εισαγωγή

XML Language

Πριν μιλήσουμε για το XML schema, θα πρέπει να πούμε δύο λόγια για το τι είναι η γλώσσα XML. Κατά τη φάση συλλογής πληροφοριών για την εργασία, διάβασα ένα μέρος μιας δημοσίευσης στο περιοδικό 'Netweek' [1] , το οποίο έλεγε:

«*Αν παρομοιάζαμε την μεταφορά ηλεκτρονικών δεδομένων, με τις διεθνείς μετακινήσεις ενός ταξιδιώτη, είναι βέβαιο ότι τα ταξίδια του τελευταίου στην προ Internet εποχή γίνονταν με ζωήλατες άμαξες ή στην καλύτερη περίπτωση με αργοκίνητα βαπόρια. Αν όμως το διαδίκτυο αποτέλεσε το υπερπολυτελές υπερηχητικό όχημα, η περαιτέρω άνεση και ταχύτητα πήγαιναν σε μεγάλο ποσοστό χαμένες, αφού ο ταξιδιώτης μετέβαινε στις ξένες χώρες, ξεχνώντας ...το λεξικό του.*»

Στις μέρες μας είναι πιο επιτακτική από ποτέ η ανάγκη για την ύπαρξη ενός κοινού κώδικα προγραμματισμού και ανάγνωσης ηλεκτρονικών δεδομένων. Η ύπαρξη, βέβαια, αυτής της κοινώς αποδεκτής και συμβατής γλώσσας δεν είναι κάτι καινούργιο ή μια παροδική μόδα, αφού η απουσία της είχε γίνει αισθητή και κατ' επέκταση επισημανθεί, εδώ και αρκετά χρόνια. Στα χρόνια της προ

διαδικτύου εποχής το κακό συνέχιζε να διαιωνίζεται, χωρίς ιδιαίτερες συνέπειες. Στην καλύτερη περίπτωση, που οι υπολογιστές μιας εταιρείας ήταν συνδεδεμένοι μεταξύ τους σε δίκτυο, οι διάφοροι χρήστες μπορούσαν να ανταλλάξουν μεταξύ τους αρχεία και δεδομένα, χωρίς φυσικά κανείς να παίρνει καν "μυρωδιά" για το τι συμβαίνει στις άλλες εταιρείες, ή ακόμη και έναν όροφο πιο πάνω ή πιο κάτω. Η εκρηκτική ανάπτυξη του Internet, τα τελευταία χρόνια, ήταν η αφορμή που αποκάλυψε την αδυναμία των διάφορων υπολογιστών να επικοινωνήσουν αποτελεσματικά μεταξύ τους.

Από αυτή τη δύσκολη θέση καλείται να μας βγάλει σήμερα η Extensible Mark-up Language (XML), η γλώσσα που θεωρείται από πολλούς ως η πιο σημαντική αλλαγή στο χώρο του προγραμματισμού, μετά την εμφάνιση της Java. Εξελιγμένη αρχικά στα μέσα της δεκαετίας του '80, η XML διαθέτει στις μέρες μας την επίσημη έγκριση του τεχνικού τμήματος του World Wide Web Consortium και απολαμβάνει την χρησιμοποίησή της από ονόματα όπως η Microsoft και η IBM.

Ο όρος *Mark-up* αναφέρεται σε ένα σύνολο από ενδείξεις που αναφέρονται και περιγράφουν τα δεδομένα ενός κειμένου. Πρόκειται για οδηγούς στον τρόπο με τον οποίο πρέπει να ερμηνευτούν τα περιεχόμενα του κειμένου. Μια συλλογή τέτοιων ενδείξεων, που ακολουθούν καθορισμένο συντακτικό και γραμματική, μπορεί να θεωρηθεί γλώσσα. Μία Markup γλώσσα προσδίδει πληροφορία για τα περιεχόμενα ενός κειμένου. Οι Markup Languages είναι μια μέθοδος για

να δημιουργούμε μεταδεδομένα (*metadata*). Τα δεδομένα που κωδικοποιούνται σε XML μετατρέπονται σε έναν standard τύπο εγγράφου τον οποίο μπορεί να διαβάσει και να επεξεργαστεί οποιαδήποτε εφαρμογή που υποστηρίζει XML τεχνολογία.

Απλή τόσο στην χρήση όσο και στην γενικότερη λογική της, η XML στηρίζεται στη δενδρική διάταξη "ετικετών", οι οποίες περιλαμβάνουν ένα σύνολο παρεμφερών πληροφοριών, οι οποίες μπορούν να ερμηνευθούν εύκολα και να ανακτηθούν γρήγορα και συνδυαστικά από οποιονδήποτε υπολογιστή.

Παραθέτοντας ένα πιο παραστατικό παράδειγμα, θα μπορούσαμε να πούμε, ότι αν μέχρι τώρα είχαμε την μεμονωμένη ηχογράφηση ενός τραγουδιού σε μια κασέτα, ένα δίσκο ή ένα CD ανά περίπτωση, χωρίς να μπορούμε να το αναπαράγουμε από την μία συσκευή στην άλλη, αφού προφανώς το CD player δεν θα ήταν δυνατόν να παίξει μια κασέτα. Η XML μας παρέχει τη δυνατότητα να καταγράψουμε το τραγούδι με την αλληλουχία των μουσικών νότων που το απαρτίζουν, ώστε η ανάγνωση και η αναπαραγωγή να είναι εφικτή σε οποιαδήποτε συσκευή. Το όλο ζήτημα έγκειται πλέον στο κατά πόσο ο προγραμματιστής θα αποδώσει πιστά την μελωδία, προκειμένου να καταστήσει το τελικό αποτέλεσμα όσο πιο άρτιο γίνεται. Ενώ λοιπόν μια καλή "εγγραφή" μπορεί να κάνει το "σερφάρισμα" στο δίκτυο μια εύκολη υπόθεση, μια αντιστοίχως κακή δεν αποκλείεται ακόμη και να μπλοκάρει μια μηχανή αναζήτησης.

Για να αποφευχθούν τυχόν λάθη, οι διάφοροι βιομηχανικοί κλάδοι υιοθετούν κάποιες μικρές κωδικές παραλλαγές, γνωστές ως XML schemata, οι οποίες ουσιαστικά είναι η λεπτομερέστερη απόδοση κάποιων εξειδικευμένων όρων, με αποτέλεσμα τη δημιουργία ορισμένων επιμέρους "διαλέκτων".

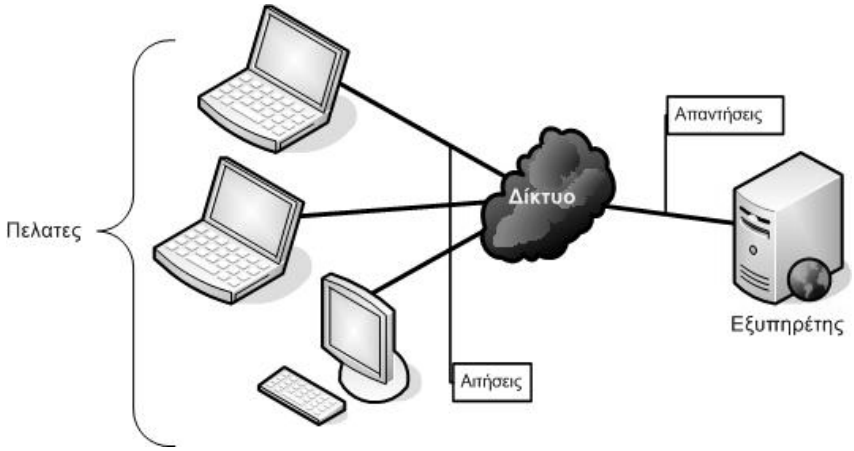
Όπως συνιστά και το ίδιο το όνομά της, η XML σχετίζεται άμεσα με την HTML (Hypertext Mark-up Language), την γλώσσα στην οποία είναι γραμμένες οι ιστοσελίδες του διαδικτύου. Αν και οι βασικές αρχές λειτουργίας προσομοιάζουν σε μεγάλο βαθμό και βασίζονται στην χρήση των ετικετών, που είδαμε πιο πάνω, η HTML περιορίζεται στην περιγραφή μόνο του πως μια ιστοσελίδα θα πρέπει να φαίνεται στην οθόνη του υπολογιστή (χωρισμός παραγράφων, υπογράμμιση και έμφαση λέξεων κ.τ.λ.). Με απλά λόγια, η χρηστική της αξία αφορά περισσότερο στην εμφάνιση και λιγότερο στην ερμηνεία αυτή καθ' αυτή του περιεχομένου. Βασική διαφορά της HTML με την XML είναι ότι ένα XML αρχείο ορίζει αυστηρά που αρχίζει (start tag) και που τελειώνει (end tag) κάθε κομμάτι του κειμένου, δίνοντας έτσι ξεχωριστό νόημα στα περιεχόμενα των tags. Βασικός παράγοντας της δημοτικότητας της XML είναι η απλότητα και η ευκολία στην χρήση της, γεγονός που τη διαφοροποιεί από την πρόγονό της που ήταν η Standard Generalised Mark-up Language (SGML), βασισμένη στην ίδια "φιλοσοφία" δόμησης και χρήσης με την XML, η δυσκολία και η πολυπλοκότητα όμως της

οποίας την κατέστησαν απορριπτέα στην ευρύτερη "κοινωνία" των προγραμματιστών.

Η XML έδωσε στους προγραμματιστές την ελευθερία να δημιουργήσουν τα δικά τους σχήματα για την διανομή και την αποθήκευση πληροφοριών. Οι προγραμματιστές αυτοί, έχουν δημιουργήσει έγγραφα που αντιπροσωπεύουν ένα απίστευτα μεγάλο μέγεθος πληροφοριών και η XML μπορεί να διευκολύνει στην επίλυση πολλών και διαφορετικών προβλημάτων διανομής πληροφορίας. Ένα βασικό μέρος αυτής της διαδικασίας είναι η επίσημη δήλωση και τεκμηρίωση αυτών των σχημάτων, τα οποία αποτελούν τη βάση πάνω στην οποία οι υπεύθυνοι για την ανάπτυξη λογισμικού μπορούν να στηριχτούν.

Όπως απορρέει από τα παραπάνω, η XML είναι μια καλή, ευέλικτη λύση, η οποία υιοθετείται από ολοένα και περισσότερες επιχειρήσεις-οργανισμούς, που επιζητούν μια επιλογή με ουσιαστικά αποτελέσματα και μεσοβραχυπρόθεσμο χαρακτήρα απόδοσης, χωρίς ρίσκα τεχνικής και οικονομικής μορφής. Φαίνεται λοιπόν ότι, επιτέλους, οι δρόμοι της τεχνολογικής ανάπτυξης και της επιχειρηματικής λογικής συγκλίνουν και τελικά ενώνονται, καθώς προγραμματιστές και μάνατζερ έχουν πλέον τη δυνατότητα να μιλούν την ίδια γλώσσα.

1.1 Εισαγωγή στα XML έγγραφα



Ο τρόπος λειτουργίας του διαδικτύου περιλαμβάνει τη διαρκή ανταλλαγή δεδομένων μεταξύ πελατών (*clients*) και εξυπηρετητών (*servers*). Μέχρι πρότινος, κοινή γλώσσα επικοινωνίας για το σκοπό αυτό ήταν η HTML την οποία μπορούσε να ερμηνεύσει ο φυλλομετρητής (*browser*) του πελάτη. Η HTML αποτελεί ένα πεπερασμένο σύνολο ετικετών (*tags*) οι οποίες αφορούν κυρίως τη μορφοποίηση των δεδομένων και ερμηνεύονται ενιαία από όλους τους φυλλομετρητές.

Ωστόσο, ένας μεγάλος όγκος των δεδομένων που διακινούνται στο διαδίκτυο δεν παρουσιάζουν σταθερή δομή και η HTML, λόγω του περιορισμένου αριθμού των ετικετών της και του προσανατολισμού της στην περιγραφή της μορφής των εγγράφων, δεν ήταν δυνατό να χειριστεί ευέλικτα ανομοιογενείς μεταξύ τους συλλογές δεδομένων οι οποίες χαρακτηρίζονταν από μια χαλαρή δομή, με μια ενιαία μορφοποίηση. Τα δεδομένα τέτοιου τύπου ονομάζονται «ημιδομημένα».

1.2 XML (eXtesible Mark-up Language)

Η γλώσσα XML αναπτύχθηκε από τον οργανισμό W3C το 1998 και πολύ σύντομα καθιερώθηκε ως στάνταρτ για την ανταλλαγή δεδομένων στη βιομηχανία λογισμικού. Ο λόγος ήταν η ανάγκη αναπαράστασης ημιδομημένων δεδομένων.

Η γλώσσα XML (*eXtensible Mark-up Language*) ήρθε για να καλύψει τα κενά και τις αδυναμίες της HTML, χωρίς, ωστόσο, να την ακυρώνει. Ο περιορισμένος αριθμός ετικετών που διέθετε η HTML την έκαναν απλούστερη στη χρήση αλλά προβληματική στο να αντιμετωπίζει καινούρια επικοινωνιακά δεδομένα. Με την έλευση της XML τα προβλήματα αυτά λύθηκαν αφού ο σχεδιαστής έχει την δυνατότητα να φτιάχνει τις δικές του ετικέτες (*tags*) και να προσδίδει λειτουργικότητα στα έγγραφα που δεν ήταν δυνατή με την HTML.

Πέρα από την επεκτασιμότητα που μόλις αναφέρθηκε, άλλα πλεονεκτήματα της είναι ο διαχωρισμός περιεχομένου και της παρουσίασης των δεδομένων, η δυνατότητα χειρισμού δεδομένων από πολλές διαφορετικές πηγές και η ικανότητα ανάκτησης και επεξεργασίας των δεδομένων χωρίς την ανάγκη προϋπάρχουσας περιγραφής για αυτά.

Η XML αποτελεί μια ενιαία μέθοδο και ένα διεθνές, ανεξάρτητο κατασκευαστή, πρότυπο για την περιγραφή και ανταλλαγή δομημένης ή αδόμητης πληροφορίας, επιτρέποντας στους διακομιστές να αλληλεπιδρούν ευκολότερα. Το κύριο γνώρισμά της είναι να περιγράφει τη δομή και τη σημασιολογία και όχι το περιεχόμενο και τη μορφοποίησή του. Έτσι το περιεχόμενο διαχωρίζεται από τον τρόπο παρουσίασής του επιτρέποντας πολλαπλές οπτικές των ίδιων δεδομένων.

1.3 Η δομή ενός XML εγγράφου

Ένα υπόδειγμα xml αρχείου comp.xml.

```
<? xml version”1.0” encoding =”utf-8”?>
<company id=”1234”>
    <name>Strg</name>
    <type>Home</type>
    <address>
        <street>street</street>
        <city>city</city>
        <country>country</country>
    </address>
</company>
```

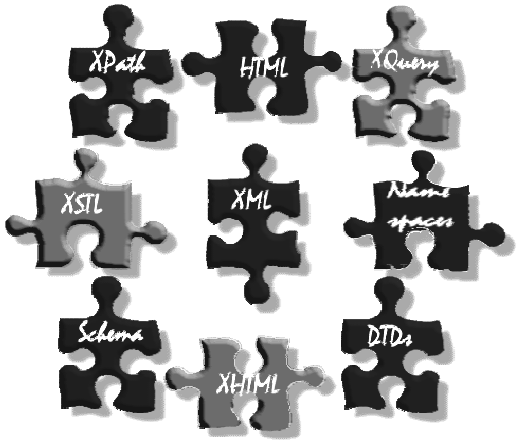
Όπως φαίνεται από το υπόδειγμα του εγγράφου που παρουσιάζεται παραπάνω, η πρώτη γραμμή ενός εγγράφου XML που είναι συνήθως η δήλωση XML περιλαμβάνει την έκδοση της XML που χρησιμοποιείται, την κωδικοποίηση και το αν θα χρησιμοποιηθούν

εξωτερικές περιγραφές του περιεχομένου. Το δεύτερο στοιχείο που παρατηρείται είναι η ύπαρξη των ετικετών ή στοιχείων που περιγράφουν το περιεχόμενο, το πρώτο στοιχείο θεωρείται η ρίζα και σε αυτό εμφωλεύονται όλα τα υπόλοιπα. Πρέπει να προσεχθεί, επίσης, το γεγονός ότι τα ονόματα των ετικετών στην XML είναι διαφορετικά αν είναι κεφαλαία από το αν είναι μικρά. Παραδείγματος χάριν δεν είναι το ίδιο η ετικέτα <*NAME*> με την ετικέτα <*name*>. Άλλο χαρακτηριστικό της XML είναι οι ιδιότητες (*attributes*) οι οποίες είναι ζευγάρια ονόματος-τιμής που περιέχουν περιγραφικές πληροφορίες για ένα στοιχείο, όπως στο στοιχείο ***<company id="1234">***.

Συνοψίζοντας, μπορούμε να πούμε ότι ένα XML έγγραφο θα θεωρείται καλώς μορφοποιημένο και έγκυρο εάν υπακούει στους παρακάτω κανόνες:

- Μοναδικό ριζικό στοιχείο
- Συμφωνία Πεζών – Κεφαλαίων
- Συμφωνία ετικετών Ανοίγματος – Κλεισίματος
- Σωστά εμφωλευμένες ετικέτες
- Οι τιμές των ιδιοτήτων να εσωκλείονται σε εισαγωγικά
- Όχι επαναλαμβανόμενες ιδιότητες στο ίδιο στοιχείο

2. Τεχνολογίες γύρω από την XML



Με την ανάπτυξη της XML προέκυψε η ανάγκη για μια σειρά από άλλες σχετικές τεχνολογίες που τη συμπληρώνουν και επεκτείνουν τη λειτουργικότητά της. Αυτές αφορούν κυρίως ειδικές γλώσσες υποβολής ερωτημάτων σε XML έγγραφα με σκοπό την εξαγωγή από αυτά συγκεκριμένων τμημάτων τους. Τέτοια παραδείγματα είναι οι γλώσσες *XPath* και *XQuery*. Άλλα πρότυπα περιλαμβάνουν το δομικό ορισμό (κανόνες σύνταξης) για συγκεκριμένους τύπους XML εγγράφων, όπως τα *DTDs* και το *XMLSchema*. Επίσης, γλώσσες που παράγονται από XML με κατάλληλους μετασχηματισμούς (*HTML* και *XHTML*) και τα εργαλεία για την επίτευξη των μετασχηματισμών αυτών (*XSL* και *XSTL*) και τέλος επεκτάσεις στη δομή των ίδιων των XML εγγράφων για εμπλουτισμό των δυνατοτήτων τους (*namespaces*). Στη συνέχεια θα εξεταστούν πιο αναλυτικότερα οι παραπάνω τεχνολογίες.

2.1 Namespaces

Σε αρκετές περιπτώσεις χρειάζεται να αναφερθούμε με το ίδιο όνομα σε ένα στοιχείο ή ιδιότητα ενός XML εγγράφου το οποίο εμφανίζεται με διαφορετικό εννοιολογικό περιεχόμενο σε κάθε περίπτωση. Η χρήση των *namespaces* εγγυάται τη μοναδική ονομασία στοιχείων και XML ιδιοτήτων τα οποία αποτελούν συλλογές που σχετίζονται με ένα συγκεκριμένο URI.

Αυτό που πρέπει να επισημανθεί εδώ είναι ότι αν και το *namespace* δείχνει σε μια δικτυακή τοποθεσία δεν είναι απαραίτητο να στοχεύει σε υπαρκτή ιστοσελίδα, αρκεί αυτό να είναι μοναδικό.

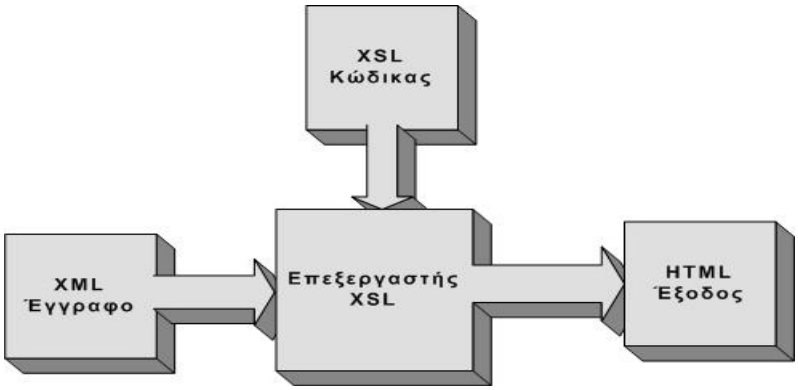
<channel xmlns:xsi=http://www.w3.org/2001/XMLSchema-nstance>

τοπικό όνομα στοιχείου — **πρόθεμα** — **Namespace URI**

2.2 Μετασχηματισμοί XSL και XSLT

Ανάμεσα στα σημαντικά θετικά στοιχεία που αναφέρθηκαν σχετικά με το πρότυπο XML ήταν και η ανεξαρτησία του από τον τρόπο προβολής του περιεχομένου. Παρόλα αυτά όμως εξακολουθεί να υπάρχει η ανάγκη μετατροπής των συχνά ακατάληπτων XML εγγράφων σε πιο οικίες και ευπαρουσίαστες μορφές, παρόμοιες με αυτές που

συναντώνται στα HTML έγγραφα. Το θέμα αυτό έρχεται να αντιμετωπίσει η τεχνολογία *XSL* (*eXtensible Style Language*).



Μηχανισμός λειτουργίας μετατροπέα XSL.

2.2.1 Τι είναι η XSL;

Η XSL είναι ένα στάνταρτ που έχει συσταθεί από το World Wide Web Consortium (W3C). Τα πρώτα δύο τμήματα της γλώσσας αποτέλεσαν μια σύσταση (Recommendation) του W3C τον Νοέμβριο του 1999. Η πλήρης σύσταση της XSL που περιελάμβανε τη μορφοποίηση (formatting) της XSL έγινε υποψήφια για σύσταση (Candidate Recommendation) στο W3C τον Νοέμβριο του 2000.

Επειδή η XML δεν χρησιμοποιεί προκαθορισμένα tags, όπως η HTML, αλλά μπορούμε να χρησιμοποιήσουμε εμείς ότι tags θέλουμε, οι έννοιες αυτών των tags δεν είναι κατανοητές: για παράδειγμα, το <table> μπορεί να σημαίνει έναν πίνακα της HTML ή ένα τραπέζι.

Εξαιτίας της φύσης της XML, ο φυλλομετρητής (browser) δεν είναι σε θέση να γνωρίζει πώς πρέπει να εμφανίσει ένα XML έγγραφο.

Για να μπορέσουμε να εμφανίσουμε τα XML έγγραφα, είναι απαραίτητο να διαθέτουμε έναν μηχανισμό που να περιγράφει το πώς πρέπει να εμφανισθούν τα έγγραφα. Ένας απ' αυτούς τους μηχανισμούς είναι τα CSS, αλλά η XSL (eXtensible Stylesheet Language) είναι η προτιμώμενη γλώσσα φύλλων στυλ της XML. Η XSL είναι κάτι περισσότερο από ένα φύλλο στυλ (style sheet), είναι μια γλώσσα που μπορεί να μετασχηματίσει την XML σε HTML, μια γλώσσα που μπορεί να φιλτράρει και να ταξινομήσει τα δεδομένα της XML, μια γλώσσα που μπορεί να έχει πρόσβαση σε τμήματα ενός XML εγγράφου, μια γλώσσα που μπορεί να μορφοποιήσει τα δεδομένα της XML βασισμένη στις τιμές των δεδομένων, όπως την εμφάνιση των αρνητικών αριθμών με κόκκινο και μια γλώσσα που μπορεί να εξάγει τα XML δεδομένα σε διαφορετικές συσκευές, όπως οθόνη, χαρτί ή φωνή.

Πως δουλεύει;

Η διαδικασία styling απαιτεί: α) ένα βασικό έγγραφο XML, το οποίο θα περιλαμβάνει τις πληροφορίες που το φύλλο στυλ θα προβάλει, και β) το ίδιο το φύλλο στυλ το οποίο περιγράφει την προβολή του συγκεκριμένου αρχείου XML.

Παρακάτω ακολουθεί ένα παράδειγμα:

Το αρχείο XML:

```
<scene>
    <FX>General Road Building noises. </FX>
      <speech speaker=”Prosser”>
        Come off it Mr Dent , you can’t win you know. There’s no point in lying down in the path of progress.
      </speech>
      <speech speaker=”Arthur”>
        I’ve gone off the idea of progress. It’s overrated
      </speech>
</scene>
```

Αυτό το XML αρχείο δεν περιέχει πληροφορίες παρουσίασης, οι οποίες περιλαμβάνονται στο φύλλο στυλ. Διαχωρίζοντας το περιεχόμενο του βασικού εγγράφου XML και τις πληροφορίες του φύλλου στυλ σε δύο διαφορετικά αρχεία, επιτρέπει στους χρήστες να δουν το έγγραφο σύμφωνα με τις προτιμήσεις και τις δυνατότητές τους με την τροποποίηση του φύλλου στυλ.

Το φύλλο στυλ:

```
...
<xsl:template match=”FX”>
  <fo:block font- weight="bold">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
<xsl:template match="speech[@speaker='Arthur']">
  <fo:block background-color="blue">
    <xsl:value-of select="@speaker"/>:
```

```
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>
  …
```

Το πρώτο κομμάτι κώδικα λέει ότι ένα στοιχείο FX θα μετασχηματιστεί σε έναν μπλοκ με ένα έντονο φόντο.

Το δεύτερο κομμάτι κώδικα ισχύει για όλα τα στοιχεία *speech* που έχουν στην ιδιότητα *speaker* την τιμή *Arthur* και τα σχηματοποιεί σε ένα μπλε μπλοκ μέσα στο οποίο η τιμή της ιδιότητας *speaker* προστίθεται πριν από το κείμενο.

Η απόδοση:

General Road Building noises.

Prosser: Come off it Mr Dent, you can't win you know. There's no point in lying down in the path of progress.

Arthur: I've gone off the idea of progress. It's overrated

Η XSL ουσιαστικά αποτελείται από τρεις γλώσσες, η σπουδαιότερη από τις οποίες είναι η XSLT :

Η ***XSLT*** είναι μια γλώσσα για τον μετασχηματισμό των XML εγγράφων σ' άλλα είδη εγγράφων ή σ' άλλα XML έγγραφα.

Η ***XPath*** είναι μια γλώσσα για να έχουμε πρόσβαση στα τμήματα ενός XML εγγράφου και σχεδιάστηκε για να χρησιμοποιηθεί από την XSLT.

Η ***μορφοποίηση*** (Formatting) είναι η διαδικασία της μετατροπής του αποτελέσματος ενός μετασχηματισμού της XSL σε μια κατάλληλη μορφή εξόδου για έναν αναγνώστη ή ακροατή.

Η XSLT και η XPath παρουσιάστηκαν σαν δύο ξεχωριστές συστάσεις (Recommendations) του W3C στις 16 Νοεμβρίου 1999. Δεν υπάρχει κάποιο ξεχωριστό έγγραφο του W3C για τα XSL Formatting Objects, αλλά υπάρχει μια περιγραφή μέσα στη σύσταση XSL 1.0.

2.2.2 Η XSLT (XSL Transformations)

Η *XSLT* αποτελεί το σημαντικότερο κομμάτι της XSL στάνταρτ. Είναι το μέρος εκείνο της XSL που χρησιμοποιείται για να μετασχηματίσει ένα XML έγγραφο σ' ένα άλλο XML έγγραφο ή σ' έναν άλλον τύπο εγγράφου. Η XSLT μπορεί να χρησιμοποιηθεί για να μετασχηματίσουμε ένα XML έγγραφο σε μια μορφή (format) που να αναγνωρίζεται από έναν φυλλομετρητή (browser). Μια τέτοια μορφή, ως γνωστόν, είναι η HTML. Κανονικά η XSLT το κάνει αυτό μετασχηματίζοντας το κάθε στοιχείο της XML σ' ένα στοιχείο της HTML.

Η XSLT μπορεί επίσης να προσθέσει τελείως καινούργια στοιχεία στο αρχείο εξόδου ή να αφαιρέσει στοιχεία. Μπορεί να αναδιατάξει και

να ταξινομήσει τα στοιχεία, να κάνει δοκιμές και να πάρει αποφάσεις για το ποια στοιχεία να εμφανίσει και πολλά άλλα.

Κατά τη διαδικασία του μετασχηματισμού, η XSLT χρησιμοποιεί την XPath για να ορίσει τμήματα του πηγαίου εγγράφου που ταιριάζουν μ' ένα ή περισσότερα προκαθορισμένα πρότυπα (templates). Όταν βρεθεί ένα ταίριασμα, η XSLT θα μετασχηματίσει το τμήμα του πηγαίου εγγράφου που ταιριάζει στο προκύπτον έγγραφο. Τα τμήματα του πηγαίου εγγράφου που δεν ταιριάζουν μ' ένα πρότυπο θα πάνε αμετάβλητα στο αποτέλεσμα. Αυτό ισχύει σαν ένας γενικός κανόνας.

Λίγοι φυλλομετρητές υποστηρίζουν την XSL προς το παρόν, ένας εκ των οποίων είναι η έκδοση του Internet Explorer 5.0 και μετά, καθώς επίσης ο Mozilla/Firefox. Για να μπορέσουμε να επεξεργαστούμε ένα έγγραφο της XML χρησιμοποιώντας την XSL, χρειαζόμαστε έναν XML parser με μια XSL Engine. Ο Internet Explorer 5.0 περιέχει έναν XML parser μαζί με μια XSL engine.

Παράδειγμα:

Αρχικά ξεκινάμε με το XML έγγραφο που θέλουμε να μετασχηματίσουμε σε HTML :

```
<?xml version="1.0"?>
    <CATALOG>
        <CD>
            <TITLE> Secret Combination </TITLE>
            <ARTIST> Kalomoira </ARTIST>
            <COUNTRY> Greece </COUNTRY>
            <COMPANY> EMI </COMPANY>
            <PRICE> 20.90€ </PRICE>
            <YEAR> 2008 </YEAR>
        </CD>
        ……..
```

Μετά δημιουργούμε ένα έγγραφο XSL Style Sheet μ' ένα πρότυπο μετασχηματισμού (transformation template) :

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
 <html>
 <body>
  <table border="2" bgcolor="yellow">
   <tr>
    <th> Title </th>
    <th> Artist </th>
   </tr>
   <xsl:for-each select="CATALOG/CD">
```

```
<tr>
  <td><xsl:value-of select="TITLE"/></td>
  <td><xsl:value-of select="ARTIST"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Μετά προσθέτουμε μια αναφορά για το XSL Style Sheet έγγραφο μέσα στο XML έγγραφο :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="cd_catalog.xsl"?>
<CATALOG>
<CD>
    <TITLE> Secret Combination </TITLE>
    <ARTIST> Kalomoira </ARTIST>
    <COUNTRY> Greece </COUNTRY>
    <COMPANY> EMI </COMPANY>
    <PRICE> 20.90€ </PRICE>
    <YEAR> 2008 </YEAR>
</CD>
……
```

Το επόμενο XSL Style Sheet περιέχει ένα πρότυπο για να δημιουργήσει το XML CD Catalog:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
 <html>
 <body>
  <table border="1">
    <tr>
      <th> Title </th>
      <th> Artist </th>
    </tr>
    <tr>
      <td> . </td>
      <td> . </td>
    </tr>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Εφόσον το style sheet αποτελεί το ίδιο ένα XML έγγραφο, ξεκινάει με μια xml δήλωση (declaration): *<?xml version='1.0'?>*. Το tag

xsl:stylesheet στην δεύτερη γραμμή ορίζει την αρχή του stylesheet. Το tag *xsl:template* στην τρίτη γραμμή ορίζει την αρχή ενός προτύπου. Το χαρακτηριστικό *match="/"* του template συσχετίζει (ταιριάζει) το πρότυπο με τη ρίζα (root, /) του πηγαίου XML εγγράφου. Το υπόλοιπο του εγγράφου περιέχει το ίδιο το πρότυπο, εκτός από τις δύο τελευταίες γραμμές που ορίζουν το τέλος του προτύπου και το τέλος του style sheet.

Το αποτέλεσμα του μετασχηματισμού θα μοιάζει ως εξής :

Title Artist

. .

Με τη χρήση του *XSL*, των *μετασχηματιστών XSL* (*XSLT*), που περιέχουν το συντακτικό της *XSL* και αναλαμβάνουν την ταύτιση των προτύπων και τη μορφοποίηση της εξόδου, και τη βοήθεια της γλώσσας *XPath* που θα εξεταστεί παρακάτω, είναι δυνατή η παρουσίαση ενός XML εγγράφου σε πρακτικά οποιαδήποτε μορφή (html, PDF, κ.ο.κ.). Αυτό που πραγματικά κάνει το *XSL* είναι να μετασχηματίζει το XML σε μια δενδρική δομή, έπειτα μετασχηματίζει αυτή τη δομή σε μια άλλη και τελικά εξάγει τη νέα δομή σε μορφή XML.

Παρακάτω δίνεται ένα παράδειγμα εγγράφου XML και ενός αρχείου *XSL* για τη μετατροπή του σε HTML.

Το υπό μετατροπή xml αρχείο Hello.xml

```
<?xml version=”1.0” encoding=”utf-8”>
<?xml-stylesheet type=text/xsl href=”hello.xsl”?>
<greet>Hello World</greet>
```

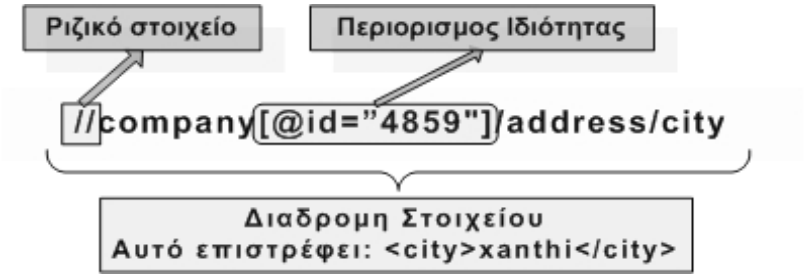
Το αρχείο hello.xsl που μορφοποιεί το παραπάνω xml αρχείο

```
<?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>Greedings</title></head>
        <body>
        <font color=”red” face=”arial”>
                <xsl:value-of select=”greet”/>
        </font>
        </body>
        </html>
</template>
</stylesheet>
```

2.3 Οι γλώσσες υποβολής ερωτημάτων XPath και XQuery

Ένα από τα βασικά στοιχεία στις δομές *XSTL* είναι και η δυνατότητα εκτέλεσης πολύπλοκων διαδρομών μέσα στα έγγραφα XML με τη χρήση ειδικής σημασιολογίας που αποτελεί τη γλώσσα *XPath*. Με τη γλώσσα αυτή εκμεταλλευόμαστε τη δενδρική δομή με την οποία μπορούν να αναπαρασταθούν τα XML δεδομένα για να

προσπελάσουμε στοιχεία, ιδιότητες και τις τιμές τους. Κάθε βήμα για τον υπολογισμό της έκφρασης μονοπατιού αναπαριστά κίνηση μέσα σε ένα έγγραφο σε μια καθορισμένη κατεύθυνση. Η γλώσσα ορίζει επίσης και ένα σύνολο συναρτήσεων που επεκτείνουν ακόμη περισσότερο τις δυνατότητές της.



Ερώτημα XPath

Παρακάτω παρατίθεται ένας συνοπτικός πίνακας με διάφορα παραδείγματα εκφράσεων μονοπατιού (*path expressions*) και τις επεξηγήσεις τους.

/**bookstore**/*	Όλοι οι απόγονοι του στοιχείου bookstore
/**bookstore**/**book[1]**	Το πρώτο στοιχείο book που είναι απόγονος του στοιχείου bookstore
//**title[@lang]**	Όλα τα στοιχεία title που έχουν μια ιδιότητα με το όνομα lang
//**title[@*]**	Όλα τα στοιχεία title που έχουν οποιαδήποτε ιδιότητα
//**book**/**title** | //**book**/**price**	Όλα τα title και price στοιχεία από όλα τα στοιχεία book

Τα βασικά μειονεκτήματα της *XPath* είναι οι καινούριοι συντακτικοί κανόνες που εισάγονται σε συνδυασμό με ένα σύνολο περίεργων συμβολισμών (*/,[],(),*,:,::,.,..*). Τις εκφράσεις μονοπατιού της *XPath* χρησιμοποιεί μια επέκταση της γλώσσας XML, η *XQuery*.

Η *XQuery* αποτελεί μια "*native*" γλώσσα επερωτήσεων για XML, όπου ο όρος "*native*" ορίζεται σε αντιδιαστολή με τον όρο "*σχεσιακός*". Η γλώσσα *XQuery* υποστηρίζεται από ΒΔ όπως η *Oracle* από την έκδοση *10g R2*, επιτρέπει τη δημιουργία XML όψεων από σχεσιακά δεδομένα και τη βελτιστοποίηση επερωτήσεων (*query optimization*) ενώ παρέχει την δυνατότητα μίξης στο ίδιο ερώτημα *XQuery* και *SQL*. Τα αποτελέσματά της αποτελούν ετερογενείς ακολουθίες μιας ταξινομημένης λίστας αντικειμένων. Κάθε βήμα, επίσης, παράγει ένα σύνολο από αντικείμενα που χρησιμεύουν ως σημείο εκκίνησης για το επόμενο βήμα. Ένα παράδειγμα *XQuery* φαίνεται παρακάτω.

<u>Δείγμα κώδικα XQuery:</u>

```
<descriptive-catalog>
 {
      for $i in document("catalog.xml")//item,
          $p in document("parts.xml")//part[partno=$i/partno],
          $s in
          document("suppliers.xml")//supplier[suppno=$isuppno]
```

```
        order by $p/description, $s/suppname
        where $i/price<1000
        return
         <item>
           {
              $p/description,
              $s/suppname,
              $i/price
           }
         </item>
 }
</descriptive-catalog>
```

3. "Σκελετοί" XML εγγράφων.

3.1 DTDs και XML Schemas

Η XML παρέχει τη δυνατότητα να δημιουργηθούν πρακτικά άπειρες γλώσσες περιγραφής δεδομένων όπως και έχει ήδη συμβεί γνωστές τέτοιες γλώσσες είναι οι *MML* για μουσικά κομμάτια, *CML* για χημικά, *MathML* για μαθηματικές εξισώσεις και *RSS* για μετάδοση ειδήσεων μέσω ιστοσελίδων. Για να ερμηνευτεί όμως κατάλληλα η κάθε μια θα πρέπει να οριστεί ένα σύνολο συντακτικών κανόνων που θα πρέπει να ακολουθούν όλα τα έγγραφα της ίδιας γλώσσας για να θεωρούνται έγκυρα και να μπορούν να ερμηνευτούν σωστά. Τον ρόλο αυτό παίζουν οι δομές *DTDs* και τα *XML Schemas*.

Το *DTD* (*Document Type Definition*), ορίζει τα στοιχεία που επιτρέπεται να υπάρχουν σε ένα XML έγγραφο, τη σειρά εμφάνισής τους, το εύρος τιμών τους και την αρχική τιμή καθώς και διάφορα αναγνωριστικά μέσω συμβόλων όπως "," (*ακολουθία*), "|" (*ένα από*), "?" (*προαιρετικό*), "*" (*μηδέν ή περισσότερα*), "+" (*ένα ή περισσότερα*). Παρακάτω φαίνεται ένα παράδειγμα κώδικα ενός *DTD*.

Comp.dtd

```
<!DOCTYPE company[
        <!ELEMENT company (name, type, address*)>
        <!ELEMENT company id ID #REQUIRED>
        <!ELEMENT  name (#PCDATA)>
        <!ELEMENT type (#PCDATA)>
        <!ELEMENT address ( street, city, country)>
        <!ELEMENT street (#PCDATA)>
        <!ELEMENT city (#PCDATA)>
        <!ELEMENT country (#PCDATA)>
]>
```

Για να καταστεί δυνατό ένα XML αρχείο να αναγνωρίσει αν και ποιανού *DTD* εγγράφου τους κανόνες θα πρέπει να ακολουθήσει, πρέπει να ενσωματωθεί σε αυτό είτε ολόκληρο το *DTD* έγγραφο, είτε απλά μια αναφορά σε αυτό η οποία τοποθετείται αμέσως κάτω από το ριζικό στοιχείο του εγγράφου. Η αναφορά αυτή είναι:

<!DOCTYPE company SYSTEM "Comp.dtd">

Όπως γίνεται εύκολα κατανοητό από τη μορφή του *DTD* αρχείου, αυτό αποτελεί μια καινούρια μορφή σύνταξης που δεν έχει σχέση με την XML. Άλλα μειονεκτήματά του είναι η έλλειψη υποστήριξης για τα *namespaces*, καθώς και το περιορισμένο του λεξιλόγιο.

Οι αδυναμίες που παρουσίαζε το *DTD* λύθηκαν με την έλευση του *XML Schema*. Ένα *XML Schema* αποτελεί, και από πλευράς οργάνωσης και ως προς τους τύπος των δεδομένων του, τον ορισμό ενός συγκεκριμένου XML εγγράφου. Το *XML Schema* είναι από μόνο του ένα XML έγγραφο, οπότε μπορεί να το χειριστεί κανείς με τα ίδια εργαλεία που χειρίζεται και το ίδιο το XML έγγραφο το οποίο περιγράφει το *XML Schema*.

Τα βασικά χαρακτηριστικά του είναι εμφωλευμένες δηλώσεις στοιχείων και δομικών αντικειμένων με το πρόθεμα "*xsd:*" που αποτελεί το όνομα του *namespace*: ***"http://www.w3.org/2001/XMLSchema"***. Είναι δυνατός ο ορισμός ομάδων (*groups*), τόσο στοιχείων, όσο και ιδιοτήτων που αποτελούν συλλογές αντικειμένων στις οποίες μπορούν να γίνουν αναφορές από διάφορα σημεία του εγγράφου. Επίσης πέρα από τους ήδη ενσωματωμένους τύπους δεδομένων στους οποίους συμπεριλαμβάνονται οι *Boolean*, *string*, *decimal*, *float*, *date* και άλλοι, μπορεί ο χρήστης να ορίζει τους δικούς του τύπους δεδομένων είτε *simpleType*, είτε *complexType*. Μπορούν να οριστούν επίσης σύνθετες δομές όπως η *sequence* για την περιγραφή σύνθετων στοιχείων, η *choice* που ορίζει μια επιλογή μεταξύ δυνατών στοιχείων και η *all* που ορίζει ένα αταξινόμητο σετ στοιχείων, καθώς και περιορισμοί, όπως αυτοί του κλειδιού και της μοναδικότητας. Το παραγόμενο, από το υπόδειγμα XML αρχείο, έγγραφο φαίνεται παρακάτω.

Comp.xsd

```
<?xml version =”1.0” encoding=”utf-8”?>
<xsd: schema xmls: xsd = http://www.w3.org/2001/XMLSchema>
<xsd: element name = “company”>
  <xsd: annotation>
      <xsd: documentation>Root Element</xsd: documentation>
  </xsd: annotation>
  <xsd: complexType>
      <xsd: sequence>
          <xsd: element name = “name” type =”xsd: string”/>
          <xsd: element name = “type” type =”xsd: string”/>
          <xsd: element name = “address”>
              <xsd: complexType>
                  <xsd: sequence>
                      <xsd: element name = “street” type =”xsd:
string”/>
                      <xsd: element name = “city” type =”xsd: string”/>
                      <xsd: element name = “country” type =”xsd:
string”/>
                  </xsd: sequence>
              </xsd: complexType>
          </xsd: element>
      </xsd: sequence>
      <xsd: attribute name =”id” type =”xsd: integer” use
=”required”/>
    </xsd: complexType>
</xsd: element>
</xsd: schema>
```

Και σε αυτή την περίπτωση απαιτείται η προσθήκη μιας αναφοράς στο XML αρχείο που θα δηλώνει τη θέση του *XML Schema* στο οποίο υπακούει. Η αναφορά αυτή, η οποία αναγράφεται παρακάτω, τοποθετείται ως *namespace* ιδιότητα στο ριζικό στοιχείο του XML εγγράφου.

xmlns: xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi: schemaLocation =
"http://mycompany.com/namespace/comp.xsd"

3.2 Τι είναι και τι κάνει ένα σχήμα για την XML;

Η XML Schema είναι μια γλώσσα διαμόρφωσης περιορισμών, που εκφράζεται ως κανόνες ή ως πρότυπο δομής, οι οποίοι κανόνες ισχύουν για μια κατηγορία εγγράφων XML. Από πολλές απόψεις, τα σχήματα χρησιμεύουν σαν εργαλείο σχεδίου που θεσπίζουν ένα πλαίσιο πάνω στο οποίο μπορούν να στηριχτούν εφαρμογές. Δεδομένου ότι η διαμόρφωση είναι ένα απαραίτητο κομμάτι για τους σχεδιαστές λογισμικού, η τυποποίηση των περιορισμών και των δομών XML εγγράφων μπορεί να οδηγήσει σε διαφορετικές εφαρμογές. Αν και εφευρίσκονται καθημερινά νέες εφαρμογές για σχήματα, οι περισσότερες από αυτές μπορούν να ταξινομηθούν ως εφαρμογές για επικύρωση, τεκμηρίωση, υποστήριξη ερωτημάτων, «σύνδεση» δεδομένων, σύνταξη με καθοδήγηση.

- Επικύρωση

Η επικύρωση είναι η πιο κοινή χρήση των σχημάτων για την XML. Υπάρχουν πολλοί λόγοι για να επικυρωθεί ένα έγγραφο XML, όταν λαμβάνουμε ένα, όταν παραγάγουμε ένα, για να εξετάσει την έξοδο μιας εφαρμογής, κ.λπ.... Σε όλες αυτές τις περιπτώσεις, ένα σχήμα βοηθά στην ολοκλήρωση ενός ουσιαστικού μέρος της εργασίας. Τα διαφορετικά είδη σχημάτων ερμηνεύουν διαφορετικά είδη επικύρωσης, και μερικοί ιδιαίτερα σύνθετοι κανόνες μπορούν να εκφραστούν καλύτερα στο διαδικαστικό κώδικα παρά σε ένα περιγραφικό σχήμα, αλλά η επικύρωση είναι γενικά ο αρχικός σκοπός ενός σχήματος.

Η επικύρωση μπορεί να θεωρηθεί σαν ένα "firewall" ενάντια στην ποικιλομορφία της XML. Το χρειαζόμαστε αυτό κυρίως σε δύο καταστάσεις: για να χρησιμεύσει ως πραγματικό firewall όταν λαμβάνουμε έγγραφα από τον εξωτερικό κόσμο (όπως είναι συνήθως η περίπτωση με τις υπηρεσίες Ιστού και άλλες επικοινωνίες XML), και για να παρέχει σημεία ελέγχου όταν σχεδιάζουμε τις διαδικασίες ως αγωγός των μετασχηματισμών. Με την επικύρωση των εγγράφων με τα σχήματα, μπορούμε να εξασφαλίσουμε ότι το περιεχόμενο των εγγράφων προσαρμόζεται στο αναμενόμενο σύνολο κανόνων μας, που απλοποιούν τον κώδικα που απαιτείται για να επεξεργαστεί τους κανόνες αυτούς.

Η επικύρωση των εγγράφων μπορεί ουσιαστικά να μειώσει τον κίνδυνο εγγράφων XML που παραλαμβάνονται από πηγές πέρα από τον έλεγχό μας. Είναι ένα χρήσιμο πρόσθετο στρώμα δοκιμών που εγκαθιστάτε μεταξύ της διεπαφής (interface) επικοινωνιών και του εσωτερικού κώδικά μας.

Η επικύρωση μπορεί να πραγματοποιηθεί σε διάφορα επίπεδα. Η δομική επικύρωση επιβεβαιώνει ότι οι δομές στοιχείων και ιδιοτήτων XML καλύπτουν τις συγκεκριμένες απαιτήσεις, αλλά δεν διευκρινίζει πολύ το περιεχόμενο κειμένου εκείνων των δομών. Η επικύρωση στοιχείων εξετάζει το περιεχόμενο των δομών εκείνων (στοιχείων και ιδιοτήτων), που εξασφαλίζουν την προσαρμογή στους κανόνες, οι οποίοι κανόνες συσχετίζονται με τις πληροφορίες που παρουσιάζονται. Άλλα είδη επικύρωσης, αποκαλούμενα συχνά επιχειρησιακοί κανόνες, μπορούν να ελέγξουν τις σχέσεις μεταξύ των πληροφοριών και ενός πιο υψηλού επιπέδου λογικότητας ελέγχου, αλλά αυτό είναι συνήθως η περιοχή του διαδικαστικού κώδικα και όχι επικύρωση βασισμένη σε σχήμα.

- Τεκμηρίωση

Τα σχήματα χρησιμοποιούνται συχνά και για λεξιλόγια σε ένα έγγραφο XML, ακόμα και όταν η επικύρωση δεν είναι μια απαίτηση. Τα σχήματα παρέχουν μια επίσημη περιγραφή του λεξιλογίου με ακρίβεια και περιεκτικότητα που είναι δύσκολο να επιτευχθούν στην πεζογραφία.

Είναι πολύ ασυνήθιστο να δημοσιευθεί η προδιαγραφή ενός νέου λεξιλογίου XML χωρίς την προσθήκη κάποιας μορφής σχήματος XML.

Η αναγνωσιμότητα των σχημάτων από τις 'μηχανές' τους δίνει διάφορα πλεονεκτήματα όπως συμβαίνει με την τεκμηρίωση. Η κατανοήσιμη από τον άνθρωπο τεκμηρίωση μπορεί να παραχθεί από την επίσημη περιγραφή του σχήματος. Τα σχήματα IDEs, παραδείγματος χάριν, παρέχουν γραφικές απεικονίσεις που βοηθούν στην κατανόηση της δομή των εγγράφων. Το W3C XML σχήμα έχει εισαγάγει τις πρόσθετες διευκολύνσεις για τον σχολιασμό των σχημάτων με δομημένες και μη πληροφορίες, που καθιστούν ευκολότερη την άμεση χρήση των σχημάτων ως πλαίσια τεκμηρίωσης.

- Υποστήριξη ερωτημάτων

Οι πρώτες εκδόσεις XPath και XSLT καθορίστηκαν για να λειτουργήσουν χωρίς οποιαδήποτε άμεση κατανόηση της δομής των εγγράφων που χειρίζονται. Αυτό έχει θέσει όρια στην απόδοση και την λειτουργία τους. Η δεύτερη έκδοση XPath και XSLT και η πρώτη έκδοση XQuery, μια νέα προδιαγραφή που καθορίζει μια γλώσσα διατύπωσης ερωτημάτων XML, θα βασιστούν στη διαθεσιμότητα ενός W3C XML σχήματος για εκείνα τα χαρακτηριστικά γνωρίσματα.

- «Σύνδεση» δεδομένων

Η σύνδεση στοιχείων αν και δεν είναι ιδιαίτερα δύσκολο να γραφτούν οι εφαρμογές που επεξεργάζονται τα έγγραφα XML χρησιμοποιώντας το SAX, DOM, και παρόμοια APIs, αυτό είναι ένας

χαμηλού επιπέδου στόχος, επαναλαμβανόμενος και επιρρεπής σε λάθη. Το κόστος για αυτά τα προγράμματα αυξάνεται ραγδαία καθώς ο αριθμός στοιχείων και ιδιοτήτων σε ένα λεξιλόγιο αυξάνεται. Η ιδέα της αυτοματοποίησης αυτών μέσω "της σύνδεσης" των διαθέσιμων πληροφοριών στα έγγραφα XML κατευθείαν στις δομές των εφαρμογών (γενικά ως αντικείμενα ή πίνακες RDBMS) είναι πιθανώς τόσο παλαιά όσο η σήμανση.

Πολλές διαφορετικές γλώσσες, είτε συγκεκριμένες είτε γλώσσες γενικής χρήσης σχημάτων XML, καθορίζουν αυτές τις συνδέσεις. Το W3C XML σχήμα κατευθύνεται προς αυτήν την περιοχή. Πολλά στοιχείο-συνδετικά εργαλεία άρχισαν να υποστηρίζουν το W3C XML σχήμα, ακόμη και από τις πρώτες εκδόσεις του πολύ προτού να οριστικοποιηθεί η προδιαγραφή της σύνδεσης δεδομένων.

- Σύνταξη με καθοδήγηση

Οι συντάκτες XML (και οι συντάκτες SGML πριν από αυτούς) έχουν χρησιμοποιήσει από καιρό τα σχήματα για να παρουσιάσουν στους χρήστες τις κατάλληλες επιλογές κατά τη διάρκεια της δημιουργίας και της έκδοσης εγγράφων. Ενώ τα DTDs παρείχαν δομικές πληροφορίες, οι πρόσφατες γλώσσες XML σχημάτων προσθέτουν περιπλοκότερες δομικές πληροφορίες και πληροφορίες datatype.

Ο οργανισμός W3C δημιουργεί ένα τυποποιημένο API που μπορεί να χρησιμοποιηθεί από καθοδηγούμενες εφαρμογές σύνταξης

για να ρωτηθεί ένας επεξεργαστής σχημάτων ποια ενέργεια μπορεί να εκτελεστεί σε μια ορισμένη θέση στο έγγραφο, για παράδειγμα: "Μπορώ να παρεμβάλω αυτό το νέο στοιχείο εδώ;", "Μπορώ να ενημερώσω αυτόν τον κόμβο κειμένων με αυτήν την τιμή;", κ.λπ. Όταν αυτό το API οριστικοποιηθεί και διαδοθεί στο ευρύτερο κοινό, θα πρέπει να επιτρέψει στους χρήστες να συνδέσουν τον επεξεργαστή σχημάτων της επιλογής τους με οποιαδήποτε εφαρμογή σύνταξης σχήματων.

3.3 Το XML σχήμα του οργανισμού W3C

Η έκδοση XML 1,0 περιλαμβάνει ένα σύνολο εργαλείων για τις δομές εγγράφων XML, τα οποία καλούνται *Document Type Definitions (DTD)*. Τα DTDs παρέχουν ένα σύνολο εργαλείων για τον καθορισμό στοιχείων και ιδιοτήτων που επιτρέπονται σε ένα έγγραφο, καθώς επίσης και τους μηχανισμούς για τις προκαθορισμένες τιμές για τις ιδιότητες αυτές, καθορίζοντας το περιεχόμενο αυτό ως επαναχρησιμοποιήσιμο και μερικά είδη πληροφοριών μεταδεδομένων. Αν και τα DTDs υποστηρίζονται και χρησιμοποιούνται ευρέως, πολλοί υπεύθυνοι για την ανάπτυξη της XML ξεπέρασαν γρήγορα τις ικανότητες που παρέχουν τα DTDs.

Μια εναλλακτική πρόταση σχημάτων, XML-Data, υποβλήθηκε στον οργανισμό W3C προτού καν η XML 1,0 οριστικοποιηθεί ως σύσταση.

Η κοινοπραξία World Wide Web (W3C), θεσμικός φύλακας της προδιαγραφής XML, επιδίωξε να χτίσει μια νέα γλώσσα για την περιγραφή των εγγράφων XML. Υπήρχε η ανάγκη να παρέχεται περισσότερη ακρίβεια στην περιγραφή των δομών εγγράφων και του περιεχομένου τους, για την υποστήριξη των XML namespaces, και να χρησιμοποιήσει ένα λεξιλόγιο XML για να περιγράψει την ίδια τη γλώσσα. Η ομάδα εργασίας των XML σχημάτων του οργανισμού W3C πέρασε δύο έτη αναπτύσσοντας δύο κανονιστικές συστάσεις, XML σχήματα μέρος 1ο : Δομές, και XML σχήματα μέρος 2ο: Τύποι δεδομένων, μαζί με μια μη-κανονιστική σύσταση, μέρος 0: Εισαγωγή. Το XML σχήμα του οργανισμού W3C έχει σχεδιαστεί να υποστηρίξει όλες τις παραπάνω εφαρμογές.

3.4 Το πρώτο μας σχήμα

Αρχίζοντας από ένα απλό παράδειγμα, με περιορισμένο αριθμό στοιχείων και ιδιοτήτων και κανενός είδους περιορισμού namespaces[1], θα δούμε πώς ένα σχήμα μπορεί να προέλθει απλά από τη δομή εγγράφων, χρησιμοποιώντας έναν κατάλογο στοιχείων σε ένα έγγραφο δεδομένων καθώς συντάσσουμε ένα DTD για το παρόν έγγραφο.

Το έγγραφο δείγμα παρακάτω, είναι ένα απλό αρχείο βιβλιοθηκών που περιγράφει ένα βιβλίο, τον συγγραφέα και τους χαρακτήρες του:

[1] Για τις ονομασίες (namespaces) XML αρχείων θα μιλήσουμε στη συνέχεια.

<?xml version=”1.0”> was used to <library>

<book id=”0-552-14951-9” available=”true”>

<isbn> 0-552-14951-9</isbn>

<title lang=”en”> The Da Vinci Code</title>

<author id=”DB”>

<name> Dan Brown</name>

<born> 1965-11-26 </born>

<dead> undead </dead>

</author>

<character id=”RL”>

<name> Robert Langdon</name>

<born> 1966-08-22 </born>

<qualification> Intelligent and determined</qualification>

</character>

<character id=”SN”>

<name> Sophie Neveu</name>

<born> 1971-10-04 </born>

</qualification> Elegant and sensitive </qualification>

</character>

<character id=”LT”>

<name> Leigh Teabing </name>

<born> 1947-05-30 </born>

```
        </qualification> Sneaky and greedy</qualification>
    </character>
    <character id="JS">
        <name> Jacques Sauniere </name>
        <born> 1932-03-03 </born>
        </qualification> Wise and caring</qualification>
    </character>
    </book>
    </library>
```

Σ' αυτό το σημείο θα ήταν καλό να αναφερθεί ότι υπάρχουν πολλές διαφορετικές μορφές για τη σύνταξη ενός σχήματος, και υπάρχουν ακόμη περισσότερες προσεγγίσεις άντλησης ενός σχήματος από ένα έγγραφο δείγμα. Για το πρώτο σχήμα μας, θα αρχίσουμε με τη δημιουργία ενός ταξινομημένου καταλόγου των στοιχείων και των ιδιοτήτων που βρίσκονται στο σχήμα.

Τα στοιχεία που υπάρχουν στο έγγραφο μας είναι: Συγγραφέας (author), Βιβλίο (book), Ημερομηνία Γέννησης (born), Χαρακτήρας (character), Νεκρός (dead), ISBN, Βιβλιοθήκη (library), Όνομα (name), Προσόντα (qualification), και Τίτλος (title), και οι ιδιότητες είναι Διαθεσιμότητα, Ταυτότητα (id) και Γλώσσα (lang). Ξεκινώντας θα πρέπει να καθορίσουμε κάθε στοιχείο κάτω από το στοιχείο του εγγράφου του σχήματός, το οποίο ανήκει στο W3C XML σχήμα

namespace (http://www.w3.org/2001/XMLSchema) και προτάσσεται συνήθως ως "xs."

Προτού να αρχίσουμε, πρέπει να ταξινομήσουμε τα στοιχεία και να δώσουμε μερικούς βασικούς ορισμούς για την κατανόηση πώς το W3C XML σχήμα κάνει αυτήν την ταξινόμηση.

Το πρότυπο περιεχομένου χαρακτηρίζει τους τύπους στοιχείων παιδιών και κόμβων κειμένου που μπορούν να περιληφθούν σε ένα στοιχείο (χωρίς να δώσει οποιαδήποτε περαιτέρω αναφορά στις ιδιότητες).

Το πρότυπο περιεχομένου θεωρείται ότι είναι α)"κενό" όταν δεν αναμένονται το στοιχείο παιδί, ούτε κόμβος κειμένου, "απλό" όταν γίνονται αποδεκτοί μόνο οι κόμβοι κειμένων, "σύνθετο" όταν αναμένονται μόνο τα υποστοιχεία, και "μικτό" όταν οι κόμβοι και τα υποστοιχεία κειμένων μπορούν να είναι παρόντα. Σημειώστε ότι για να καθορίσουμε το πρότυπο περιεχομένου, δίνουμε προσοχή μόνο στους κόμβους στοιχείων και κειμένων και αγνοούμε οποιαδήποτε ιδιότητα, σχόλιο, ή οδηγία επεξεργασίας που θα μπορούσαν να περιληφθούν. Παραδείγματος χάριν, ένα στοιχείο με μερικές ιδιότητες, ένα σχόλιο, και μερικές οδηγίες επεξεργασίας θα είχε ένα "κενό" πρότυπο περιεχομένου εάν δεν έχει κανένα παιδί (child) κειμένων ή στοιχείων.

Τα στοιχεία όπως το Όνομα-name, Ημερομηνία Γέννησης -born, και ο Τίτλος-title έχουν τα απλά πρότυπα περιεχομένου:

…./…

<title lang=”en”> The Da Vinci Code </title>

…/…

<name> Dan Brown </name>

<born> 1965-11-26 </born>

…/…

Στοιχεία όπως library-βιβλιοθήκη ή character-χαρακτήρας έχουν σύνθετο πρότυπο περιεχομένου:

<library>

<book id=”0-552-14951-9” available=”true”>

…/…

</book>

</library>

<character id=”SN”>

<name> Sophie Neveu</name>

<born> 1971-10-04 </born>

</qualification> Elegant and sensitive </qualification>

</character>

Το W3C το XML σχήμα θεωρεί τα στοιχεία που έχουν ένα απλό πρότυπο περιεχομένου και καμία ιδιότητα ως "απλούς τύπους," ενώ όλα τα άλλα στοιχεία (όπως το απλό περιεχόμενο με τις ιδιότητες και άλλα πρότυπα περιεχομένου) τα δέχεται ως "σύνθετους τύπους".

Με άλλα λόγια, όταν ένα στοιχείο μπορεί να έχει μόνο τους κόμβους κειμένων και δεν δέχεται οποιαδήποτε στοιχεία ή ιδιότητες παιδιών, θεωρείται απλός τύπος σε όλες τις άλλες περιπτώσεις, είναι ένας σύνθετος τύπος. Οι ιδιότητες έχουν πάντα απλό τύπο δεδομένων όταν δεν έχουν κανένα κόμβο παιδί και περιέχουν μόνο μια αξία κειμένου.

Στο παράδειγμά μας, τα στοιχεία όπως ο Συγγραφέας ή ο Τίτλος έχουν έναν σύνθετο τύπο:

<author id="DB">

<name> Dan Brown</name>

<born> 1965-11-26 </born>

<dead> undead </dead>

</author>

..../...

<title lang="en"> The Da Vinci Code</title>

Ενώ τα στοιχεία born και qualification, όπως φυσικά και όλες οι ιδιότητες, είναι απλού τύπου:

<born> 1965-11-26 </born>

.../...

</qualification> Elegant and sensitive </qualification>

.../...

<book id=”0-552-14951-9” available=”true”>

Τώρα που έχουμε τα κριτήρια για να ταξινομήσουμε τα συστατικά μας, μπορούμε να καθορίσουμε κάθε ένα από αυτά. Μπορούμε να αρχίσουμε με τον απλούστερο τύπο στοιχείου, όπως το στοιχείο του ονόματος- name που μπορεί να βρεθεί στον Συγγραφέα ή το Χαρακτήρα:

<name> Dan Brown</name>

Για να καθορίσουμε ένα τέτοιο στοιχείο, χρησιμοποιούμε έναν καθορισμό **xs:element(global definition)**, που περιλαμβάνεται άμεσα κάτω από το στοιχείο του εγγράφου xs:schema:

<xs:schema xmlns:xs=”http://www.w3.org/2001/XMLSchema”>
<xs:element name=”name” type=”xs:string”/>
…/…
</xs:schema>

Η τιμή που χρησιμοποιείται για να παραπέμψει τον τύπο δεδομένων (xs:string) δέχεται ως πρόθεμα το xs, το οποίο πρόθεμα συνδέεται με το W3C XML σχήμα. Αυτό σημαίνει ότι το *xs:string* είναι ένας προκαθορισμένος τύπος δεδομένων του W3C XML σχήμα. Το ίδιο

πράγμα μπορεί να γίνει για όλους τους άλλους απλούς τύπους καθώς επίσης και για τις ιδιότητες:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="name" type="xs:string"/>
        <xs:element name="qualification" type="xs:string"/>
        <xs:element name="born" type="xs:date"/>
        <xs:element name="dead" type="xs:date"/>
        <xs:element name="isbn" type="xs:string"/>
        <xs:element name="id" type="xs:ID"/>
        <xs:element name="available" type="xs:boolean"/>
        <xs:element name="lang" type="xs:language"/>
</xs:schema>
```

Το παραπάνω στυλ σύνταξης σχημάτων είναι γνωστό στους χρήστες σύνταξης DTDs. Πρέπει να επισημάνουμε ότι αυτός ο τρόπος είναι πιο «επίπεδος» από τη σύνταξη ενός DTD, δεδομένου ότι η δήλωση των ιδιοτήτων γίνεται έξω από τη δήλωση των στοιχείων. Αυτό οδηγεί σε ένα σχήμα, στο οποίο τα στοιχεία και οι ιδιότητες δέχονται ίση μεταχείριση. Ωστόσο, όταν ένα σχήμα περιγράφει ένα γλωσσάρι XML, που χρησιμοποιεί ένα namespace, αυτό το απλό «επίπεδο» στυλ είναι αδύνατο να χρησιμοποιήσει το μεγαλύτερο μέρος του χρόνου.

Η αφομοίωση των απλών τύπων στοιχείων και ιδιοτήτων είναι μια απλοποίηση σε σύγκριση με τα εργαλεία XPath, DOM, και στοιχείων Infoset. Αυτοί θεωρούν ένα απλό τύπο στοιχείου ως ένα ιδιαίτερο στοιχείο, που έχει ένα ενιαίο στοιχείο παιδί-child του τύπου "character," και μια ιδιότητα ως ένα στοιχείο που έχει μια ομαλοποιημένη τιμή. Το όφελος αυτής της απλοποίησης είναι ότι εμείς μπορούμε να χρησιμοποιήσουμε τους απλούς τύπους δεδομένων, για να καθορίσουμε τους απλούς τύπους στοιχείων και τις ιδιότητες και να συντάξουμε με έναν πιο συνεπή τρόπο:

<xs:element name=”isbn” type=”xs:string”/>

ή

<xs:attribute name=”isbn” type=”xs: string”/>

Η διάταξη των ορισμών σε ένα σχήμα δεν είναι κάτι το σημαντικό. Τώρα μπορούμε να κάνουμε το επόμενο βήμα από άποψη πολυπλοκότητας τύπων και να καθορίσουμε το στοιχείο τίτλου που εμφανίζεται στο έγγραφο δείγμα ως εξής:

<title lang=”en”>

The Da Vinci Code

</title>

Δεδομένου ότι αυτό το στοιχείο έχει μια ιδιότητα, έχει έναν σύνθετο τύπο. Και εφόσον έχει μόνο έναν κόμβο κειμένων, θεωρείται να έχει ένα απλό περιεχόμενο.

Επομένως, θα συντάξουμε τον καθορισμό του ως εξής:

```
<xs:element name=”title”>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base=”xs:string”>
        <xs:attribute ref=”lang”/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Η σύνταξη XML το καθιστά φλύαρο, αλλά μπορεί σχεδόν να διαβαστεί σε τόσο σαφή αγγλικά όπως "το στοιχείο που ονομάζεται «title» έχει έναν σύνθετο τύπο, ο οποίος είναι ένα απλό περιεχόμενο που εξασφαλίζεται με την επέκταση του προκαθορισμένου τύπου xs:string από την προσθήκη των ιδιοτήτων που καθορίζονται σε αυτό το σχήμα και αναφέρονται στο όνομα lang."

Τα υπόλοιπα στοιχεία (Βιβλιοθήκη, Βιβλίο, Συγγραφέας, και Χαρακτήρας) είναι όλοι σύνθετοι τύποι με σύνθετο περιεχόμενο.

Καθορίζονται με τον καθορισμό της ακολουθίας στοιχείων και ιδιοτήτων που θα τους συνθέσει.

Το στοιχείο Βιβλιοθήκη, ο απλούστερος από αυτούς, ορίζεται ως εξής:

```
<xs:element name=”library”>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=”book” maxOccurs=”unbounded”/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Αυτός ο καθορισμός δεδομένου μπορεί να διαβαστεί ως "το στοιχείο που ονομάζεται Βιβλιοθήκη είναι ένας σύνθετος τύπος που αποτελείται από μια ακολουθία περιπτώσεων (*τύπου μία προς πολλά, maxOccurs*) των στοιχείων που ορίζονται και έχοντας ως αναφορά το όνομα book."

Το στοιχείο Συγγραφέας (author), που έχει μια ιδιότητα και για τον οποίο μπορούμε να εξετάσουμε την ημερομηνία θανάτου προαιρετικά, θα μπορούσε να είναι ως εξής:

```
<xs:element name=”author”>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=”name”/>
      <xs:element ref=”born”/>
      <xs:element ref=”dead” minOccurs=”0”/>
    </xs:sequence>
    </xs:attribute ref=”id”/>
  </xs:complexType>
</xs:element>
```

Αυτό σημαίνει ότι το στοιχείο Συγγραφέας είναι ένας σύνθετος τύπος που αποτελείται από μια ακολουθία τριών στοιχείων Όνομα, Ημερομηνία Γεννήσεως, Ημερομηνία θανάτου και Ταυτότητα (id). Το στοιχείο Ημερομηνία θανάτου είναι προαιρετικό και μπορεί να εμφανιστεί μηδέν φορές.

Οι ιδιότητες minOccurs και maxOccurs, που έχουμε δει σε μερικά προηγούμενα στοιχεία, μας επιτρέπουν να καθορίσουμε τον ελάχιστο και μέγιστο αριθμό περιπτώσεων. Η προκαθορισμένη τιμή τους είναι 1 (ένα), που σημαίνει ότι όταν λείπουν και τα δύο, το στοιχείο πρέπει να εμφανιστεί ακριβώς μία φορά στην ακολουθία. Η ειδική τιμή "unbounded" μπορεί να χρησιμοποιηθεί για την ιδιότητα maxOccurs όταν ο μέγιστος αριθμός περιπτώσεων είναι απεριόριστος.

Οι ιδιότητες πρέπει να καθοριστούν μετά από την ακολουθία. Τα υπόλοιπα στοιχεία Βιβλίο και Χαρακτήρας μπορούν να καθοριστούν με τον ίδιο τρόπο, ο οποίος μας οδηγεί στο ακόλουθο πλήρες σχήμα:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="name" type="xs:string"/>
    <xs:element name="qualification" type="xs:string"/>
    <xs:element name="born" type="xs:date"/>
    <xs:element name="dead" type="xs:date"/>
    <xs:element name="isbn" type="xs:string"/>
    <xs:element name="id" type="xs:ID"/>
    <xs:element name="available" type="xs:boolean"/>
    <xs:element name="lang" type="xs:language"/>
    <xs:element name="title">
    <xs:complexType>
    <xs:simpleContent>
    <xs:extension base="xs:string">
    <xs:attribute ref="lang">
    </xs:extension>
    </xs:simpleContent>
    </xs:complexType>
    </xs:element>
    <xs:element name="library">
```

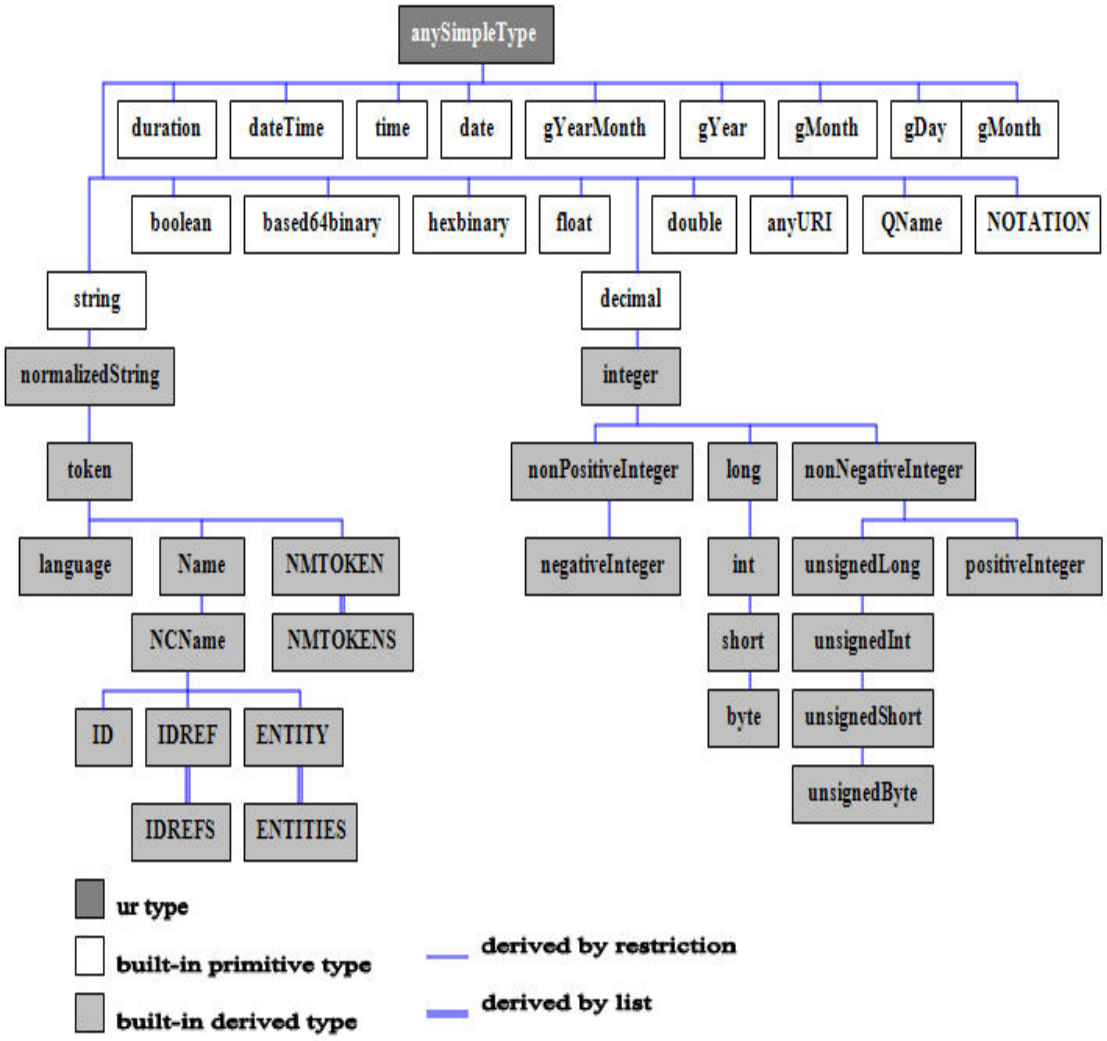
```
<xs:complexType>
<xs:sequence>
<xs:element ref=”book” maxOccurs=”unbounded”/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name=”author”>
<xs:complexType>
<xs:sequence>
<xs:element ref=”name”/>
<xs:element ref=”born”/>
<xs:element ref=”dead” minOccurs=”0”/>
</xs:sequence>
<xs:attribute ref=”id”/>
</xs:complexType>
</xs:element>
<xs:element name=”book”>
<xs:complexType>
<xs:sequence>
<xs:element ref=”isbn”/>
<xs:element ref=”title”/>
<xs:element ref=”author” minOccurs=”0”
                    maxOccurs=”unbounded “/>
```

```
<xs:element ref=”character” minOccurs=”0”
                                maxOccurs=”unbounded “/>
</xs:sequence>
<xs:attribute ref=”id”/>
<xs:attribute ref=”available”/>
</xs:complexType>
</xs:element>
<xs:element name=”character”>
<xs:complexType>
<xs:sequence>
<xs:element ref=”name”/>
<xs:element ref=”born”/>
<xs:element ref=”qualification”/>
</xs:sequence>
<xs:attribute ref=”id”/>
</xs:complexType>
</xs:element>
</xs:schema>
```

4. Τα διάφορα είδη Τύπων Δεδομένων

4.1 Χρησιμοποιώντας προκαθορισμένους τύπους δεδομένων

Το W3C XML σχήμα παρέχει ένα εκτενές σύνολο προκαθορισμένων τύπων. Παράγει πολλούς από αυτούς από ένα μικρότερο σύνολο "αρχικών" τύπων που έχουν μια συγκεκριμένη έννοια και σημασιολογία και δεν μπορούν να προέλθουν από άλλους τύπους. Το παρακάτω σχήμα παρέχει έναν χάρτη προκαθορισμένων τύπων και τις σχέσεις μεταξύ τους.



4.2 Λεξιλογικά διαστήματα και διαστήματα τιμών

Το XML σχήμα λεξικολογικών διαστημάτων και διαστημάτων τιμών εισήγαγε μια αποσύζευξη μεταξύ των δεδομένων, καθώς μπορούν να διαβαστούν από τα έγγραφα.

Προτού μπορέσουμε να οριστικοποιήσουμε τον καθορισμό αυτών των δύο διαστημάτων, πρέπει να εξετάσουμε το πρότυπο επεξεργασίας και τους μετασχηματισμούς που δέχονται από μια τιμή που γράφεται σε ένα έγγραφο XML προτού επικυρωθεί. Το περιεχόμενο των στοιχείων και των ιδιοτήτων εξελίσσεται μέσω των ακόλουθων βημάτων κατά τη διάρκεια της επεξεργασίας:

- Διάστημα αποθήκευσης σειριακών bytes (Serialization space). Η ακολουθία από bytes που αποθηκεύεται πραγματικά σε ένα έγγραφο μπορεί να τη δούμε αποθηκευμένη σε ένα πρώτο διάστημα, το οποίο μπορούμε να το πούμε "serialization space"
- Διάστημα ανάλυσης (Parsed space). Η σύσταση XML 1.0 καθιστά σαφές ότι το serialization space δεν είναι άμεσα σημαντικό στις εφαρμογές και ένας πρώτος μετασχηματισμός εκτελείται στις τιμές από τους αναλυτές XML προτού να φθάσει η τιμή σε μια εφαρμογή. Οι χαρακτήρες μετατρέπονται σε Unicode και το τέλος των γραμμών και τα κενά είναι ομαλοποιημένα. Το αποτέλεσμα

αυτού του μετασχηματισμού είναι αυτό που φθάνει στις εφαρμογές και καλείται διάστημα ανάλυσης.

- Λεξικολογικό διάστημα (lexical space). Πριν κάνουμε οποιαδήποτε επικύρωση, το W3C XML σχήμα εκτελεί έναν δεύτερο κύκλο επεξεργασίας κενών διαστημάτων στην τιμή που αναφέρεται από τον αναλυτή και μπορεί είτε να τα αγνοήσει, είτε να τα ομαλοποιήσει, είτε να καταργήσει τα κενά. Η τιμή αυτή μετά την επεξεργασία των κενών διαστημάτων ανήκει στο λεξικολογικό διάστημα.
- Διάστημα τιμής (Value space). Το W3C XML σχήμα θεωρεί ένα στοιχείο από το λεξικολογικό διάστημα ως μια αντιπροσώπευση μιας αφηρημένης τιμής της οποίας η σημασία καθορίζεται από τον τύπο δεδομένων της και μπορεί να είναι ένα κομμάτι κειμένου, ένας αριθμός, μια ημερομηνία. Το σύνολο αφηρημένων τιμών ορίζεται ως το διάστημα τιμής.

4.3 Επεξεργασία των Whitespaces

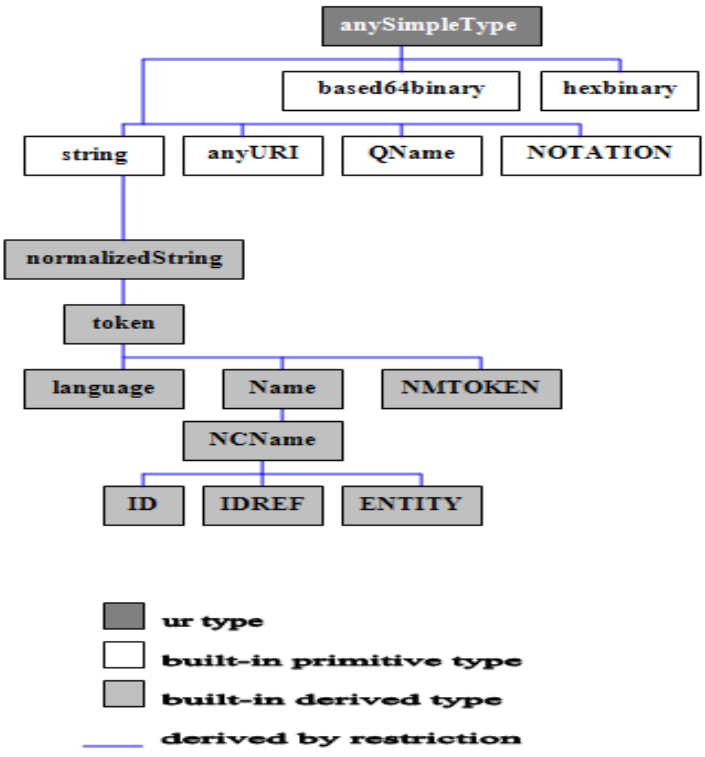
Ο χειρισμός των ειδικών χαρακτήρων tab, linefeeds, και spaces, τα οποία χρησιμοποιούνται συχνά σε καλοτυπωμένα XML έγγραφα, ήταν πάντα πολύ αμφιλεγόμενος. Το W3C XML σχήμα έχει θέσει γενικό αλγόριθμο, ο οποίος εφαρμόζεται σε δύο στάδια. Ο αλγόριθμος αυτός

εφαρμόζεται σχεδόν σε όλους τους προκαθορισμένους τύπους εκτός από δύο: xs:string και xs:normalizedString. Τα δύο στάδια είναι:

- Η αντικατάσταση του Whitespace
- Η κατάργηση του Whitespace

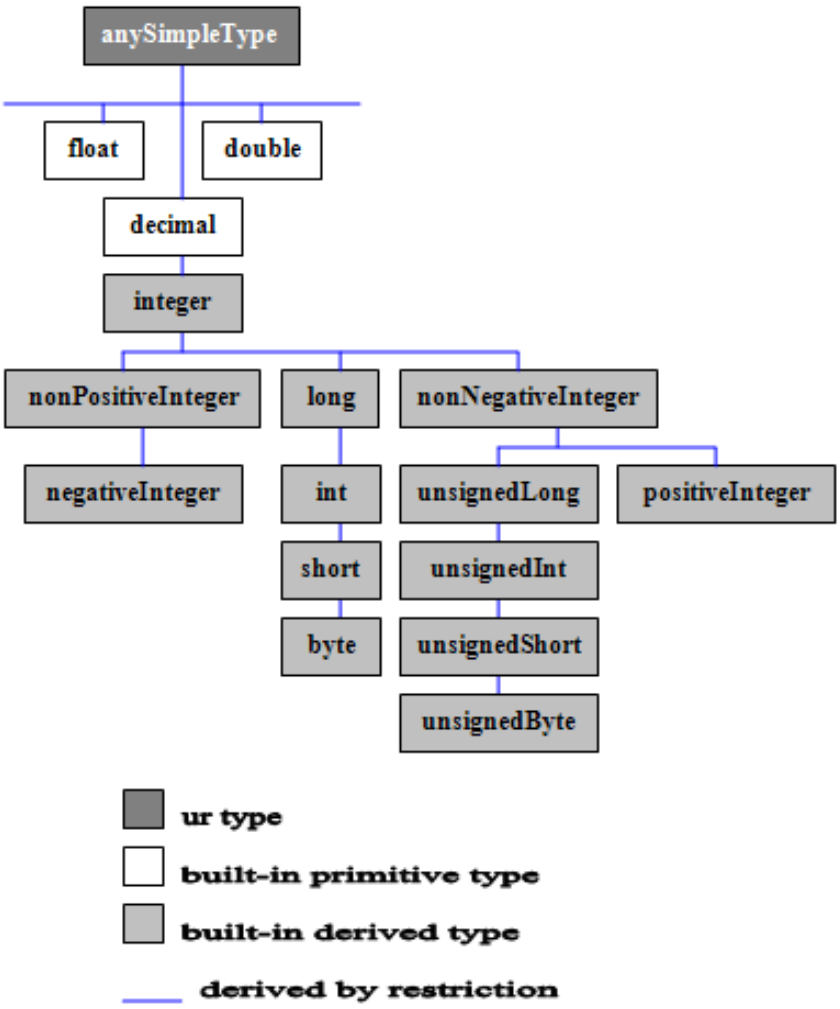
4.4 String τύπος δεδομένων

Ο αρχικός-πρωτόγονος τύπος xs:string καθώς επίσης και άλλοι τύποι με παρόμοια συμπεριφορά xs:hexBinary, xs:base64Binary, xs:anyURI, xs:QName, και xs:NOTATION, δεν αναμένονται να πάρουν οποιαδήποτε ποσοτικά προσδιορίσιμη τιμή και το διάστημα τιμής τους είναι ίδιο με το λεξικολογικό διάστημά τους εκτός από όταν ρητά περιγράφεται ειδάλλως. Οι υποκατηγορίες του τύπου string όπως φαίνεται και στο παρακάτω σχήμα είναι:



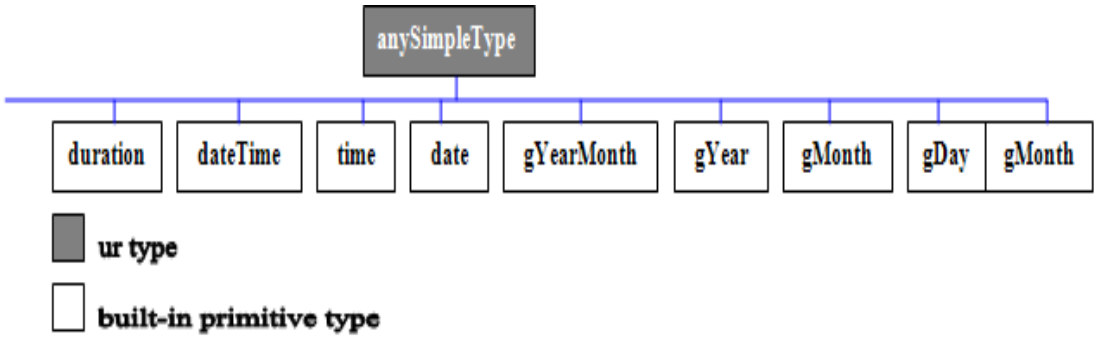
4.5 Αριθμητικός τύπος δεδομένων

Οι αριθμητικοί τύποι δεδομένων έχουν «χτιστεί» πάνω σε τέσσερις αρχικούς τύπους: xs:decimal για όλους τους δεκαδικούς τύπους, xs:double και xs:float για την μονή και διπλή ακρίβεια και xs:boolean για τιμές δύο καταστάσεων true ή false. Το Whitespaces καταργούνται για όλους τους παραπάνω τύπους. Οι υποκατηγορίες φαίνονται στην παρακάτω εικόνα:



4.6 Ημέρας και Ώρας τύπος δεδομένων

Οι τύποι αυτού του είδος φαίνονται στην παρακάτω εικόνα:



4.7 Τύπος Λίστας

Αυτός ο τύπος είναι κατάλογοι whitespace-χωρισμένων στοιχείων. Ο τύπος αυτών των στοιχείων καθορίζεται κατά τη διάρκεια της διαδικασίας παραγωγής και ο κατάλογος τύπων μπορεί να προέλθει από οποιοδήποτε άλλο απλό τύπο. Τρεις προκαθορισμένοι τύποι είναι τύπου Λίστας: xs:NMTOKENS, xs:IDREFS, και xs:ENTITIES.

Για αυτούς τους τύπους, τα στοιχεία πρέπει να χωρίζονται από ένα ή περισσότερα whitespaces.

4.8 Ο τύπος anySimpleType

Έχουμε καλύψει περιεκτικά όλους τους προκαθορισμένους τύπους εκτός από έναν, τον anySimpleType. Αυτός ο τύπος είναι ένα είδος μπαλαντέρ, που σημαίνει ότι γίνεται αποδεκτή οποιαδήποτε τιμή και δεν προσθέτει οποιοδήποτε περιορισμό στο λεξικολογικό διάστημα.

Οι τύποι anySimpleType έχουν δύο άλλα χαρακτηριστικά που τους καθιστούν μοναδικούς μεταξύ των υπολοίπων:

α) Οι απλοί τύποι των χρηστών δεν μπορούν να προέλθουν από αυτόν και τις ιδιότητές του, και

β) Η κανονική μορφή του δεν καθορίζεται από τη σύσταση.

Αυτά τα χαρακτηριστικά τον καθιστούν ως έναν τύπο προς αποφυγήν, εκτός από τις περιπτώσεις που απαιτείται η χρήση του.

5. Δημιουργία Τύπων

5.1 Δημιουργώντας Απλούς Τύπους Δεδομένων

Μέχρι τώρα, έχουμε χρησιμοποιήσει μόνο προκαθορισμένους τύπους δεδομένων. Στη συνέχεια θα δούμε πώς να δημιουργήσουμε τους νέους απλούς τύπους, που εκμεταλλεύονται τους διαφορετικούς μηχανισμούς παραγωγής και προέλευσης από τον περιορισμό.

Το W3C XML σχήμα έχει καθορίσει τρεις ανεξάρτητους και συμπληρωματικούς μηχανισμούς για τη δημιουργία των δικών μας τύπων, χρησιμοποιώντας τους ήδη υπάρχοντες τύπους. Αυτοί οι νέοι τύποι δημιουργούνται από προκαθορισμένους τύπους ή από άλλους τύπους χρηστών. Το φαινόμενο αυτό λέγεται Αναπαραγωγή.

Οι τρεις μέθοδοι Αναπαραγωγής είναι:

α) Αναπαραγωγή με περιορισμό, όπου οι περιορισμοί προστίθενται σε ένα τύπο χωρίς να αλλάζει η αρχική του σημασία. Η διαδικασία αυτή γίνεται προσθέτοντας νέους περιορισμούς στις πιθανές

τιμές του τύπου. Ο τύπος στον οποίο εφαρμόζεται ο περιορισμός λέγεται βασικός τύπος (base datatype), μπορεί να παραπεμφθεί είτε σαν <base> ιδιότητα είτε να καθοριστεί με ένα στοιχείο xs:restriction.

```
<xs:simpleType name=”myInteger”>
  <xs:restriction base=”xs:integer”>
    <xs:minInclusive value=”-2”/>
    <xs:maxExclusive value=”5”/>
  </xs:restriction>
</xs:simpleType>
```

Επιπροσθέτως, μπορεί να οριστεί σε μία διαδικασία δύο βημάτων ενσωματώνοντας τον ανώνυμο ορισμό xs:simpleType(global definition).

```
<xs:simpleType name=”myInteger”>
  <xs:restriction>
    <xs:simpleType>
      <xs:restriction base=”xs:integer”>
        <xs:maxExclusive value=”5”/>
      </xs:restriction>
    </xs:simpleType>
    <xs:minInclusive value=”-2”/>
  </xs:restriction>
</xs:simpleType>
```

β) Αναπαραγωγή από Λίστα, όπου οι νέοι τύποι ορίζονται να είναι λίστες τιμών που ανήκουν σε έναν τύπο και παίρνουν τη σημασία του τύπου λίστας. Όλα τα στοιχεία της λίστας πρέπει να έχουν τον ίδιο τύπο. Οι τύποι λίστας είναι ειδικές περιπτώσεις, στις οποίες μία δομή καθορίζεται μέσα στο περιεχόμενο ενός στοιχείου ή μιας ιδιότητας. Αυτή η πρακτική συνήθως αποθαρρύνεται, καθώς οι εφαρμογές δεν έχουν πρόσβαση στις ατομικές τιμές μέσω των τρεχόντων εκδόσεων XML.

Παράδειγμα:

<commaSeperated> 1, 2, 25 </commaSeperated>

<valueWithUnit> 10 em </valueWithUnit>

Προτιμώνται λίστες διαχωρισμένες με whitespaces και διασπώμενα XML στοιχεία και ιδιότητες, παράδειγμα:

<commaSeperated> 1 2 25 </commaSeperated>

<valueWithUnit unit=”em”> 10 </valueWithUnit>

<valueWithUnit> 10em </valueWithUnit>

γ) Παραγωγή από ένωση, όπου οι νέοι τύποι ορίζονται ως επιτρεπόμενες τιμές από ένα σύνολο άλλων τύπων και χάνουν το μεγαλύτερο μέρος της σημασίας τους. Η διαδικασία αυτή μας επιτρέπει να ορίσουμε τύπους με το «πάντρεμα» (merging) των λεξικολογικών διαστημάτων από διάφορους προκαθορισμένους ή ορισμένους από τον χρήστη τύπους.

Ο καθορισμός ενός τύπου ένωσης που αναφέρεται στους υπάρχοντες τύπους γίνεται μέσω μιας ιδιότητας memberType, που περιέχει έναν διαχωρισμένο με whitespace τύπο λίστας.Παράδειγμα:

```
<xs:simpleType name=”integerOrData”>
    <xs:union memberTypes=”xs:integer xs:date”/>
</xs:simpleType>
```

Ο καθορισμός ενός τύπου ένωσης μπορεί επίσης να επιτευχθεί με την συγχώνευση ενός ή περισσοτέρων στοιχείων <xs:simpleType>:

```
<xs:simpleType name=”myIntegerUnion”>
  <xs:union>
    <xs:simpleType>
      <xs:restriction base=”xs=integer”/>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base=”xs=NMTOKEN”/>
        <xs:enumeration value=”undefined”/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

Τα δύο παραπάνω παραδείγματα μπορούν να αναμιχθούν και να αναπαρασταθούν ως εξής:

```
<xs:simpletype name=”myIntegerUnion”>
  <xs:union memberTypes=”xs:integer”>
    <xs:simpleType>
      <xs:restriction base=”xs:NMTOKEN”>
        <xs:enumeration value=”undefined”/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

5.2 Δημιουργώντας Σύνθετους Τύπους Δεδομένων

Μέχρι τώρα έχουμε δει πώς να δημιουργήσουμε απλούς τύπους δεδομένων που μπορούν να εφαρμοστούν σε ιδιότητες. Παρακάτω γίνεται λόγος για πώς μπορούν να δημιουργηθούν σύνθετοι τύποι δεδομένων.

Οι απλοί τύποι που είδαμε πιο πριν περιγράφουν το περιεχόμενο ενός κόμβου κειμένου ή την τιμή μιας ιδιότητας. Είναι απολύτως ανεξάρτητοι από τους άλλους κόμβους, επομένως, ανεξάρτητοι και από τη σήμανση (markup). Οι σύνθετοι τύποι, αντίθετα, είναι μια περιγραφή της δομής σήμανσης. Χρησιμοποιούν απλούς τύπους για να περιγράψουν τους κόμβους στοιχείων και να αποδώσουν τις τιμές, αλλά δεν έχουν καμία άλλη σύνδεση με αυτούς. Ακόμα κι αν τα ονόματα (και

στοιχεία) είναι μερικές φορές τα ίδια, η έννοια, η χρήση, και το περιεχομένου τους διαφέρουν.

Υπάρχει ένα είδος «χάσματος» μεταξύ των απλών τύπων και του σύνθετου περιεχομένου, όπου η διάκριση μεταξύ των στοιχείων και της σήμανσης γίνεται πιο συγκεχυμένη για το W3C XML σχήμα. Αυτή η ασάφεια είναι μια συχνή πηγή σύγχυσης και πολυπλοκότητας για τους ανθρώπινους αναγνώστες, αλλά και για W3C βοηθούς σύνταξης λογισμικού και αναφοράς σχημάτων XML.

Το W3C XML σχήμα έχει εισαγάγει πολλούς διαφορετικούς τρόπους επίτευξης των πληροφοριακών μας στόχων, και θα προσπαθήσουμε να δώσουμε μια σφαιρική εικόνα του τοπίου για την αποφυγή μπερδέματος. Καταρχάς, πρέπει να κάνουμε δύο βασικές επιλογές: ποιο πρότυπο να χρησιμοποιήσουμε και πότε είναι σωστό να δημιουργήσουμε νέους τύπους εκ νέου ή να τους παράγουμε από τους προκαθορισμένους τύπους.

Παρακάτω δίνεται ένας πίνακας, ο οποίος παρουσιάζει τη σχέση μεταξύ κόμβων περιεχομένου- προτύπων κειμένου και στοιχείων παιδιών:

Κόμβοι περιεχομένου	Μεικτός	Σύνθετος	Απλός	Κενό
Παιδιά στοιχείων	Ναι	Ναι	Όχι	Όχι
Πρότυπα κειμένου	Ναι	Όχι	Ναι	Όχι

Το W3C XML σχήμα παρέχει δύο κύριους τρόπους καθορισμού σύνθετων τύπων: ένας για τα σύνθετα πρότυπα κειμένου και ένας για τα απλά πρότυπα κειμένου. Προσφέρει επίσης διάφορα τεχνάσματα καθορισμού του μικτού και κενού περιεχομένου αυτών των ορισμών.

5.3 Πρότυπα περιεχομένου

5.3.1 Πρότυπα απλού περιεχομένου

Θα αρχίσουμε με την εξέταση σύνθετων τύπων που περιέχουν απλό περιεχόμενο επειδή είναι πιο κοντά στους απλούς τύπους και παρέχουν επίσης μια ευκολότερη μετάβαση στον πιο σύνθετο κόσμο του σύνθετου περιεχομένου. Θα εστιάσουμε στο σύνθετο τύπο προτύπων απλού περιεχομένου, δηλαδή στοιχεία που έχουν μόνο κόμβους κειμένου και ιδιότητες.

- Δημιουργία προτύπων απλού περιεχομένου

Οι σύνθετοι αυτοί τύποι δημιουργούνται με την προσθήκη ενός καταλόγου ιδιοτήτων σε έναν απλό τύπο. Η λειτουργία της προσθήκης των ιδιοτήτων σε έναν απλό τύπο για να δημιουργήσει έναν δείγμα σύνθετου τύπου περιεχομένου καλείται επέκταση του απλού τύπου, πχ:

```
<xs:element name=”title”>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base=”string255”>
        <xs:attribute ref=”lang”/>
```

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

Η παραγωγή προτύπων απλού περιεχομένου γίνεται με δύο τρόπους:

α) Με επέκταση όπως φαίνεται στο ακόλουθο παράδειγμα:

```
<xs:element name="title">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extention base="tokenWithLang">
        <xs:attribute name="note" type="xs:token"/>
      <xs:extention>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

β) Και με περιορισμό όπως στο παράδειγμα που ακολουθεί:

```
<xs:element name="title">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="tokenWithLangAndNote">
```

```
            <xs:maxLength value="255"/>
              <xs:attribute name="lang" type="xs:language"/>
            <xs:attributename="note" type="xs:token"/>
          </xs:restriction>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
```

5.3.2 Πρότυπα σύνθετου περιεχομένου

Τα σύνθετα περιεχόμενα δημιουργούνται με τον καθορισμό της λίστας των στοιχείων και ιδιοτήτων τους. Ακολουθεί ένα παράδειγμα:

```
    <xs:element name="author">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="name"/>
          <xs:element ref="born"/>
          <xs:element ref="dead" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute ref="id"/>
      </xs:complexType>
    </xs:element>
```

Η παραγωγή σύνθετων προτύπων περιεχομένου γίνεται με δύο τρόπους:

α) Με επέκταση όπως φαίνεται στο ακόλουθο παράδειγμα:

```
<xs:element name=”author”>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base=”baseperson”>
        <xs:sequence>
          <xs:element ref=”dead” minOccurs=”0”/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

β) Και με περιορισμό όπως στο παράδειγμα που ακολουθεί:

```
<xs:element name=”author”>
  <xs:complexType>
    <xs:complexContent>
      <xs:restriction base=”person”>
        <xs:sequence>
          <xs:element ref=”name”/>
          <xs:element ref=”born”/>
```

```
                <xs:element ref=”dead” minOccurs=”0”/>
              </xs:sequence>
            </xs:restriction>
          </xs:complexContent>
        </xs:complexType>
      </xs:element>
```

5.3.3 Πρότυπα μεικτού περιεχομένου

Τη δημιουργία μεικτού προτύπου θα την αποδώσουμε με το παραπάνω παράδειγμα του στοιχείου title, προσθέτοντας μία μεικτή ιδιότητα με τιμή “true”:

```
      <xs:element name=”a”>
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base=”xs:string”>
              <xs:attribute name=”href” type=”xs:anyURI”/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>

      <xs:complexType name=”markedText” mixed=”true”>
        <xs:choise minOccurs=”0” maxOccurs=”unbounded”>
```

```
      <xs:element name=”em” type=”xs:token”/>
      <xs:element ref=”a”/>
    </xs:choise>
    <xs:attribute ref=”lang”/>
  </xs:complexType>
  <xs:element name”title” type=”markedText”/>
```

Όπως και στα προηγούμενα δύο πρότυπα, έτσι και στο μεικτό πρότυπο περιεχομένου, η δημιουργία του γίνεται με δύο τρόπους: α) με επέκταση και β) με περιορισμό.

5.3.4 Πρότυπα κενού περιεχομένου

Τα κενά πρότυπα είναι στοιχεία που μπορούν να δεχτούν μόνο ιδιότητες. Το W3C XML σχήμα δεν περιλαμβάνει οποιαδήποτε υποστήριξη για τα κενά πρότυπα περιεχομένου, τα οποία μπορούν να θεωρηθούν είτε ως σύνθετα πρότυπα χωρίς στοιχεία είτε ως απλά πρότυπα με την τιμή τους να περιορίζεται σε μία κενή σειρά χαρακτήρων. Για τη δημιουργία τους χρησιμοποιούμε τη μέθοδο του περιορισμού, όπως στο ακόλουθο παράδειγμα:

```
  <xs:simpleType name=”empty”>
    <xs:restriction base=”xs:string”>
      <xs:enumeration value=””/>
    </xs:restriction>
  </xs:simpleType>
```

```
<xs:complexType name=”emptyBr”>
  <xs:simpleContent>
    <xs:extension base=”empty”>
      <xs:attribute name=”id” type=”xs:ID”/>
      <xs:attribute name=”class” type=”xs:NMTOKEN”/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name=”allmostEmptyBr”>
  <xs:simpleContent>
    <xs:restriction base=”emptyBr”>
      <xs:whitespace value=”collapse”/>
      <xs:attribute name=”id” type=”xs:ID”/>
      <xs:attribute name=”class” type=”xs:NMTOKEN”/>
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
```

6. NAMESPACES

Η πολυπόθητη υποστήριξη των namespaces έκανε την εμφάνισή της στον κόσμο της XML, έχοντας ως αποτέλεσμα ένα καλοδουλεμένο επιπρόσθετο εργαλείο στα «χέρια» των W3C συντακτών σχημάτων XML.

Ωστόσο, τα Namespaces προκάλεσαν δύο προβλήματα στα DTD. Το ένα ήταν πώς θα γίνει η αναγνώριση των namespaces, που καθορίζονται χρησιμοποιώντας διαφορετικά προθέματα. Το άλλο εμπόδιο ήταν πώς θα διευκολυνθεί ο καθορισμός σχημάτων με πολλαπλά namespaces. Το W3C XML σχήμα αντιμετώπισε αυτά τα προβλήματα και προχώρησε ένα βήμα παραπέρα όσον αφορά στη χρήση των namespaces. Αυτό επιτεύχθηκε με το συσχετισμό ενός namespace με όλα τα αντικείμενα, δηλαδή στοιχεία, ιδιότητες, απλούς και σύνθετους τύπους, που καθορίζονται σε ένα σχήμα.

6.1 Τα συστατικά μέρη ενός Namespace

Τα namespace μας δίνουν τη δυνατότητα να δημιουργήσουμε μοναδικά ονόματα που εμφανίζονται σε xml αρχεία. Συσχετίζουμε ένα URI με τα ονόματα που χρησιμοποιούμε στα αρχεία μας. Όταν ένα όνομα συσχετίζεται με ένα URI, αυτό το λέμε «προσδιοριστικό όνομα» και υπάρχει μέσα σε αυτό το namespace. Επιπλέον μπορούμε να έχουμε μέσα στο αρχείο μας ονομασίες που δεν υπάρχουν σε κανένα namespace. Αυτές οι τις ονομασίες τις λέμε «μη προσδιοριστικές». Ο

συσχετισμός των ονομάτων, που βρίσκονται στα αρχεία, με ένα namespace γίνεται με τη χρήση ενός προθέματος (prefix). Το πρόθεμα αυτό είναι ένας συνδυασμός από χαρακτήρες γραμμάτων, που λειτουργεί ως συντομογραφία για το namespace, ούτως ώστε τα ονόματα να μην είναι υπερβολικά μακροσκελή. Ωστόσο το πρόθεμα είναι προαιρετικό, καθώς μπορούμε να δηλώσουμε ένα namespace ως default, εννοώντας ότι όλα τα μη-προσδιοριστικά ονόματα βρίσκονται στο namespace με το να μην εμφανίζονται.

ΠΡΟΣΟΧΗ

Όλα τα χαρακτηριστικά χωρίς πρόθεμα είναι μη-προσδιοριστικά, δηλαδή δεν είναι μέρος κάποιου namespace.

6.2 Δηλώνοντας Namespaces

Μέχρι τώρα, έχουμε δει σχήματα για έγγραφα, που δεν είχαν κανενός είδους δήλωση namespace και, επομένως, δεν ανήκαν σε κανένα namespace. Για να αντιστοιχήσουμε τα στοιχεία και τις ιδιότητες που ανήκουν σε ένα namespace, πρέπει να συσχετίσουμε αυτό το namespace με το σχήμα μας μέσω της targetNamespace ιδιότητας ενός xs:schema στοιχείου. Εάν τροποποιήσουμε τη βιβλιοθήκη μας για να χρησιμοποιήσουμε ένα ενιαίο namespace, τότε το παράδειγμα του κώδικα διαμορφώνεται ως εξής:

<library xmlns=”http://dyomeda.com/ns/library” >

<book id=”0-552-14951-9” available=”yes”>

<isbn> 0-552-14951-9</isbn>

<title lang=”en”> The Da Vinci Code</title>

…/…

</book>

</library>

Είναι ανάγκη να τροποποιήσουμε το σχήμα μας για να δηλώσουμε ένα namespace και να το καθορίσουμε ως το target namespace:

<xs:schema targetNamespace=”http://dyomeda.com/ns/library” elementFormDefault=”qualified” attributeFormDefault=”unqualified” xmlns:lib=”http://dyomeda.com/ns/library” xmlns:xs=”http://w3.org/2001/XMLSchema”>

…/…

</xs:schema>

Ο καθορισμός των namespaces είναι απόλυτα σημαντικός, δεδομένου ότι το W3C XML σχήμα τα χρησιμοποιεί για να συσχετίσει μια αναφορά URI με ένα πρόθεμα, το οποίο προσδιορίζει ένα namespace. Στο παράδειγμά μας, έχουμε δύο τέτοιες δηλώσεις:

xmlns: xs=" http://www.w3.org/2001/XMLSchema" και

xmlns: lib=" http://dyomedea.com/ns/library".

Η πρώτη δήλωση συσχετίζει το namespace του W3C XML σχήματος με το πρόθεμα xs. Θα μπορούσαμε, φυσικά, να έχουμε επιλέξει οποιοδήποτε πρόθεμα, ή ακόμα και να έχουμε χρησιμοποιήσει αυτό το namespace ως προεπιλογή, καθώς η επιλογή xs είναι απλώς κοινή χρήση.

Η δεύτερη δήλωση καθορίζει το namespace που χρησιμοποιείται στο έγγραφο μας, xmlns: lib=" http://dyomedea.com/ns/library". Εδώ επιλέξαμε να χρησιμοποιήσουμε το πρόθεμα lib, ακόμα κι αν αυτό το namespace δεν χρησιμοποιείται ποτέ για κανένα στοιχείο ή ιδιότητα του σχήματος. Θα μπορούσαμε επίσης να έχουμε επιλέξει οποιοδήποτε πρόθεμα για αυτό το namespace, ή και ακόμα να το έχουμε καθορίσει ως προεπιλεγμένο namespace.

6.3 Qualified names(QNames)

Η δήλωση ενός target namespace μας δίνει τη δυνατότητα να καθορίσουμε τα στοιχεία και τις ιδιότητες που ανήκουν στο target namespace, που λέγεται qualified και στοιχεία και ιδιότητες που δεν ανήκουν σε κάποιο namespace, που λέγεται unqualified.

Η διάκριση μεταξύ των qualified και unqualified στοιχείων και των ιδιοτήτων γίνεται μέσω των form ιδιοτήτων. Για παράδειγμα:

<xs:element name=”book” form=”qualified”/>

<xs:element name=”isbn” form=”qualified”/>

<xs:element name=”lang” form=”unqualified”/>

<xs:element name=”character” form=”unqualified”/>

Οι προκαθορισμένες τιμές για τις παραπάνω form ιδιότητες, έχουν οριστεί πιο πριν, στις elementFormDefault και attributeFormDefault ιδιότητες που προσθέσαμε στο στοιχείο xs:schema. Θυμίζω ότι, οι ιδιότητες elementFormDefault και attributeFormDefault ορίζουν προκαθορισμένες (Default) τιμές και ότι μπορούμε να διευκρινίσουμε κάθε στοιχείο και κάθε ιδιότητα αν είναι ικανή/ αρμόδια ή όχι.

Εάν θέλαμε να αποφύγουμε τη σύγχυση γράφοντας ένα σχήμα, έτσι ώστε αυτό να είναι πιο κατανοητό και ευανάγνωστο, θα μπορούσαμε να ορίσουμε στο σχήμα μας προθέματα και για τα δύο μας targetNamespaces, αλλά και για το namespace του W3C XML σχήματος. Κάνοντας αυτά, το σχήμα μας με το παράδειγμα της βιβλιοθήκης θα ξεκινούσε κάπως έτσι:

<?xml version=”1.0”?>

<xs:schema targetNamespace=”http://dyomeda.com/ns/library”

elementFormDefault=”qualified”

attributeFormDefault=”unqualified”

```
        xmlns:lib="http://dyomeda.com/ns/library"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="library">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="book" type="lib:bookType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="person">
.../...
```

Όπως δηλώθηκε το στοιχείο library, με παρόμοιο τρόπο θα δηλωθούν και τα υπόλοιπα στοιχεία του σχήματος, person, character και ο σύνθετος τύπος bookType.

6.4 Δηλώνοντας άλλα Namespaces

Ένας από τους στόχους της προδιαγραφής namespaces είναι να επιτραπεί η χρήση των εγγράφων, αναμιγνύοντας τα στοιχεία και τις ιδιότητες από διαφορετικά λεξιλόγια. Το W3C XML σχήμα αφήνει να εκμεταλλευθούμε πλήρως αυτήν την δυνατότητα. Παρακάτω δίδεται παράδειγμα μέρους της βιβλιοθήκης λεξιλογίου που περιγράφει το στοιχείο person:

```
<element name=”person”>
 <complexType>
   <sequence>
     <element name=”name” type=”string”/>
     <element name=”born” type=”date”/>
     <element name=”dead” type=”date” minOccurs=”0”/>
     <element name=”qualification” type=”string” minOccurs=”0”/>
   </sequence>
 <attribute name=”id” type=”ID” use=”required”/>
</complexType>
</element>
```

Αυτό θα μπορούσε να επαναχρησιμοποιηθεί από άλλες εφαρμογές, και αν θελήσουμε μπορούμε να καθορίσουμε ένα συγκεκριμένο namespace και να του δώσουμε την URI αναφορά “http://dyomedea.com/ns/people”. Αν το κάνουμε αυτό, το σχήμα μας θα ξεκινάει κάπως έτσι:

```
<?xml version=”1.0”?>
<library xmlns:ppl=“http://dyomedea.com/ns/people”
xmlns=”http://dyomedea.com/ns/library”>
<book id=”0-552-14951-9” available=”yes”>
<isbn> 0-552-14951-9</isbn>
…/…
<author>
```

```
    <ppl:person id="DB">
        <ppl:name> Dan Brown</ ppl:name>
        < ppl:born> 1965-11-26 </ ppl:born>
        < ppl:dead> undead </ ppl:dead>
    </ppl:person>
</author>
…/…
```

Για να διαχειριστούμε και τα δύο namespaces πρέπει να ορίσουμε και δύο σχήματα, ένα για κάθε namespace. Το σχήμα που περιγράφει το namespace xmlns:ppl="http://dyomedea.com/ns/people" είναι:

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://dyomeda.com/ns/people"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified" xmlns:ppl
    ="http://dyomeda.com/ns/people"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="person">
    <xs:complexType>
    <xs:sequence>
    <xs:element name="name" type="string"/>
    <xs:element name="born" type="date"/>
    <xs:element name="dead" type="date" minOccurs="0"/>
```

```
        <xs:element name=”qualification” type=”string” minOccurs=”0”/>
        </xs:sequence>
        <xs:attribute name=”id” type=”ID” use=”required”/>
        </xs:complexType>
        </xs:element>
</xs:schema>
```

Το σχήμα αυτό δεν περιλαμβάνει τίποτα από οποιοδήποτε άλλο namespace. Επίσης, είναι παρόμοιο με τα παραδείγματα που έχουμε δει πιο πριν. Μπορεί να χρησιμοποιηθεί μόνο του με έγγραφα που χρησιμοποιούν μόνο αυτό το namespace, αλλά μπορεί επίσης να "εισαχθεί" από σχήματα που περιγράφουν άλλα namespaces αλλά θα επιθυμούσαν να χρησιμοποιήσουν κάποιους από τους ορισμούς του.

6.5 Εισαγωγή σχήματος που δεν έχει Namespaces

Μέχρι στιγμής έχουμε δει πως μπορούμε να δηλώσουμε global στοιχεία και ιδιότητες, και τα στοιχεία και οι ιδιότητες αυτές πρέπει πάντα να είναι qualified (προσδιοριστικά). Επομένως, η μόνη λύση στην εισαγωγή των στοιχείων και των ιδιοτήτων χωρίς το namespace είναι να εισαχθεί ένα σχήμα χωρίς οποιοδήποτε targetNamespace.

Η εισαγωγή του σχήματος χωρίς namespace γίνεται όπως έχουμε δει και πιο πριν με σχήματα που έχουν targetNamespace, εκτός από τώρα, που παραλείπουμε τις ιδιότητες namespace. Πρέπει να επισημανθεί ότι για να είμαστε σε θέση να αναφερθούμε στα συστατικά

που περιλαμβάνονται στο σχήμα εισαγωγής, θα πρέπει να διατηρήσουμε την προεπιλογή namespace.

```
<?xml version=”1.0”?>
<xs:schema targetNamespace=”http://dyomeda.com/ns/library”
        elementFormDefault=”qualified”
        attributeFormDefault=”unqualified” xmlns:lib
        =”http://dyomeda.com/ns/library”
        xmlns:xs=”http://www.w3.org/2001/XMLSchema”>
        <xs:import schemaLocation=”SchemaWithoutNamespace.xsd”/>
        <xs:element name=”library”>
        <xs:complexType>
        <xs:sequence>
        …/…
```

Και το σχήμα SchemaWithoutNamespace.xsd, το οποίο γίνεται εισαγωγή είναι:

```
        <?xml version=”1.0”?>
        <xs:schema xmlns:xs=”http://www.w3.org/2001/XMLSchema”>
          <xs:element name=”person” type=”personType”/>
            <xs:complexType name=”personType”>
              <xs:sequence>
                  <xs:element name=”name” type=”xs:string”/>
              </xs:sequence>
               <xs:attribute name=”id” type=”xs:ID” use=”required”/>
```

```
    </xs:complexType>
</xs:schema>
```

Βλέπουμε λοιπόν, ότι για να καθορίσουμε μη-προσδιοριστικά στοιχεία και ιδιότητες σε ένα σχήμα με ένα targetNamespace, πρέπει να επιλέξουμε έναν από τους δύο τρόπους: είτε να τα καθορίσουμε τοπικά στο σχήμα, είτε να τα καθορίσουμε σε ένα ξεχωριστό σχήμα χωρίς targetNamespace.

7. Επεκτασιμότητα σχημάτων

Το X από το XML αντιπροσωπεύει τη λέξη "extensible", δηλαδή επεκτάσιμη. Ο στόχος οποιασδήποτε γλώσσας σχημάτων είναι να ελέγχει και να περιορίζει αυτή την επεκτασιμότητα και να βοηθήσει τις εφαρμογές να την αντιμετωπίσουν. Η επεκτασιμότητα και τα σχήματα έχουν δύο άκρως αντίθετους στόχους. Τα κακοδουλεμένα σχήματα μπορεί να οδηγήσουν σε σημαντική μείωση της επεκτασιμότητας, γεγονός που πρέπει να λαμβάνεται σοβαρά υπόψη κατά την σύνταξή τους.

Εδώ ερχόμαστε αντιμέτωποι με την δυαδικότητα μεταξύ του σχήματος και των δειγμάτων εγγράφων, και πρέπει να διακρίνουμε αυτές τις δύο διαφορετικές μορφές επεκτασιμότητας. Η επεκτασιμότητα του σχήματος, είναι στην ουσία η ικανότητα του σχήματος να επαναχρησιμοποιεί τα συστατικά του για να δημιουργεί νέα σχήματα, ενώ η επεκτασιμότητα του λεξιλογίου, είναι η ικανότητα να προστεθούν

ή και να τροποποιηθούν τα πρότυπα περιεχομένου με ελάχιστη επίδραση στις εφαρμογές, κάτι που στην πραγματικότητα είναι η διεύρυνση του σχήματος.

Η επεκτασιμότητα ενός σχήματος καθορίζεται ουσιαστικά από το στυλ του, του οποίου η επιλογή (στοιχεία και ιδιότητες, ομάδες στοιχείων και ιδιοτήτων, και απλοί και σύνθετοι τύποι) έχει γίνει καθολική(global), η χρήση των τελικών και σταθερών ιδιοτήτων, και ο προαιρετικός διαχωρισμός αυτών των συστατικών σε διαφορετικά έγγραφα σχημάτων.

Ο αντίκτυπος της προσπάθειας για σχήματα επαναχρησιμοποιήσιμα και ευέλικτα περιορίζεται στην εδραίωση των σχημάτων και ασκεί αμελητέα επίδραση στα ίδια τα έγγραφα. Εάν καθορίσουμε την διεύρυνση ενός σχήματος ως την ικανότητα να τροποποιεί το πρότυπο περιεχομένου του, το οποίο ορίζεται ρητά από το αρχικό σχήμα χωρίς τη δημιουργία ενός νέου σχήματος, τότε ο τρόπος για να συμβεί αυτό είναι η χρήση του xsi: type και των εργαλείων μπαλαντέρ.

Η ιδιότητα xsi:type ενδείκνυται ως μια αρχική προσέγγιση για τη διεύρυνση του σχήματος έχοντας υπό τον έλεγχο μας τις απρόσμενες εξελίξεις. Σε αυτήν την περίπτωση, η ιδιότητα xsi:type μπορεί να διεκπεραιωθεί μέσω των εφαρμογών για να αποφασιστεί ποιο είδος επέκτασης θα επιλεχθεί.

8. Εργαλεία Σύνταξης XML Σχημάτων

Μέχρι στιγμής έχουμε αναφέρει τη λέξη «σύνταξη σχημάτων» αρκετές φορές και έχουμε αναλύσει τη διαδικασία σύνταξης και τα συστατικά μέρη ενός σχήματος. Σ' αυτό το κεφάλαιο, θα αναφερθούμε στα εργαλεία σύνταξης σχημάτων. Πληροφοριακά παραθέτουμε κάποια ονόματα τέτοιων εργαλείων και στη συνέχεια θα ασχοληθούμε πιο λεπτομερειακά με κάποια από αυτά. Ονόματα τέτοιων εργαλείων που μπορούμε να βρούμε απλά με ένα κλικ στο διαδίκτυο είναι:

- xmlDraft
- XMLwriter
- XMLFox
- OXygen
- Smartdraw
- Editx5
- Stylus Studio
- xmlBlueprint
- XMLEditor
- XMLSpyEnt2007
- xmlViewer

Στις επόμενες σελίδες θα παραθέσουμε τα White papers (Επίσημα Έγγραφα από τις ιστοσελίδες των εταιρειών) τριών εργαλείων: xmlDraft, OXygen και XMLFox.

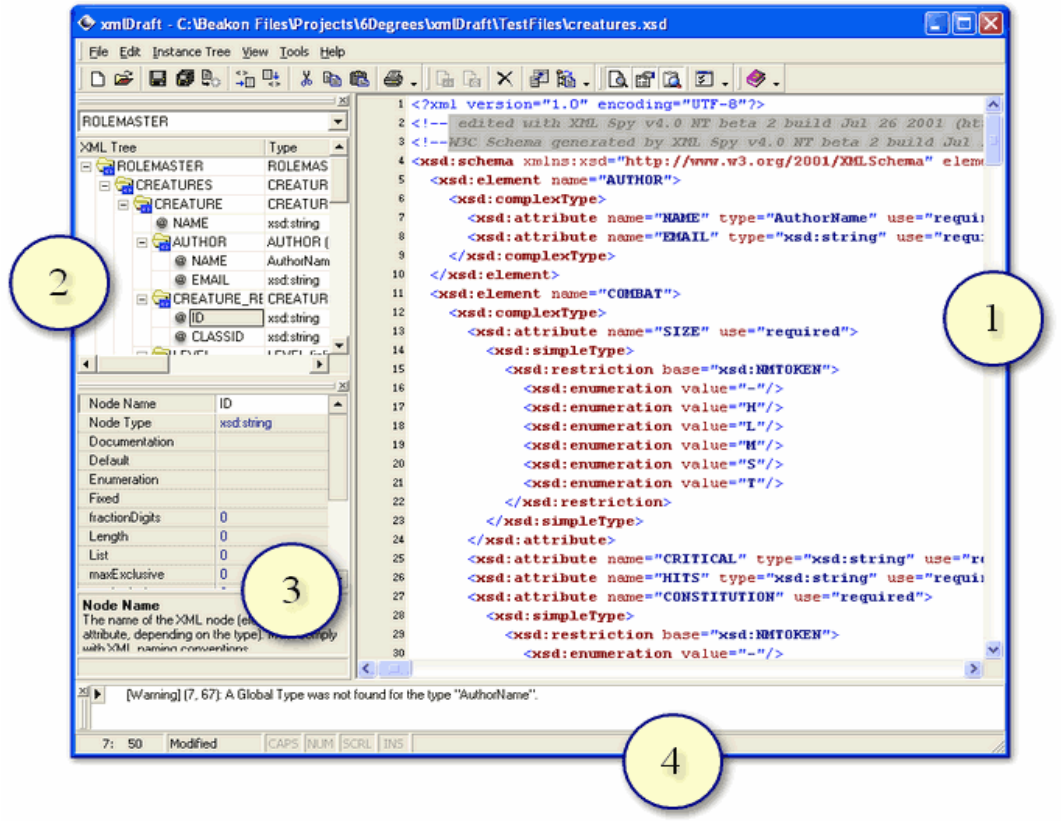
8.1 xmlDraft

Πολλά έγγραφα XML είναι μακροσκελή και δυσνόητα. Η ανάγκη να γίνουν πιο κατανοητά τα σχήματα γίνεται όλο και πιο έντονη.

Η απλοϊκότητα/ λιτότητα του xmlDraft

Το xmlDraft είναι μια προτεινόμενη λύση στην αναγκαιότητα για εξάλειψη της πολυπλοκότητας που αναφέρθηκε πιο πάνω. Το xmlDraft μας δίνει την ικανότητα καθώς τροποποιούμε ένα XML σχήμα, να παρατηρούμε τις επιπτώσεις των αλλαγών αυτών επάνω στο έγγραφο.

Παρακάτω παρατίθεται μια μικρή περιγραφή του xmlDraft:



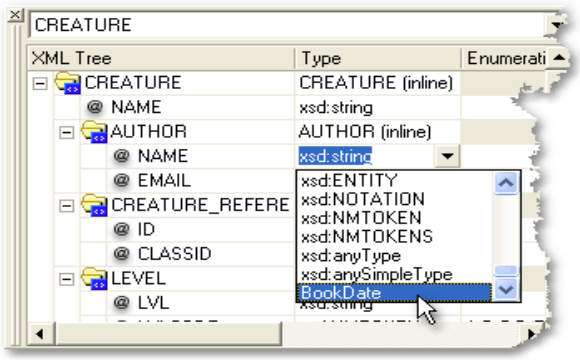
1. Ο κειμενογράφος, μας δείχνει το XML σχήμα.
2. Παράθυρο απεικόνισης του «δέντρου» με τους κόμβους.
3. Παράθυρο απεικόνισης ιδιοτήτων του επιλεγμένου κόμβου.
4. Παράθυρο απεικόνισης μηνυμάτων από το πρόγραμμα.

Δες τι ορίζεις

Το xmlDraft προσφέρει τη μοναδική ικανότητα απεικόνισης, αναπαράστασης και παρουσίασης του εγγράφου XML που το σχήμα προσπαθεί να ορίσει. Με άλλα λόγια, το παράθυρο απεικόνισης κόμβων δεν έχει πολλαπλούς κόμβους του ιδίου ονόματος, μόνο έναν.

Παράθυρο απεικόνισης κόμβων

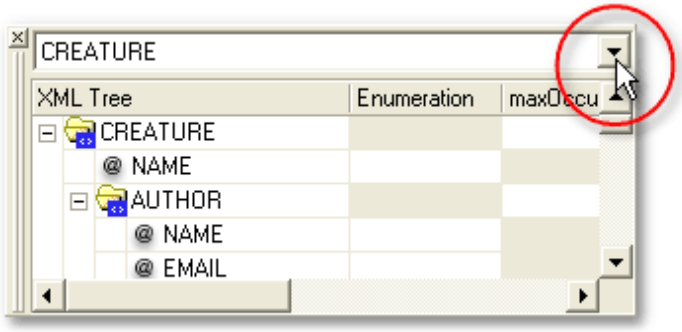
Το παράθυρο απεικόνισης κόμβων/ δέντρου, προσφέρει μία ιεραρχική άποψη του εγγράφου XML.



Όπως υποδεικνύει το παραπάνω σχήμα, έχουμε μια δενδροειδή αναπαράσταση του πως είναι το έγγραφο. Πώς δηλώνεται ο κάθε κόμβος με βάση τα στοιχεία του πλάτους του πλέγματος.

Αναπαρίστανται όχι μόνο τα στοιχεία και τα χαρακτηριστικά του, αλλά και οι ιδιότητες του.

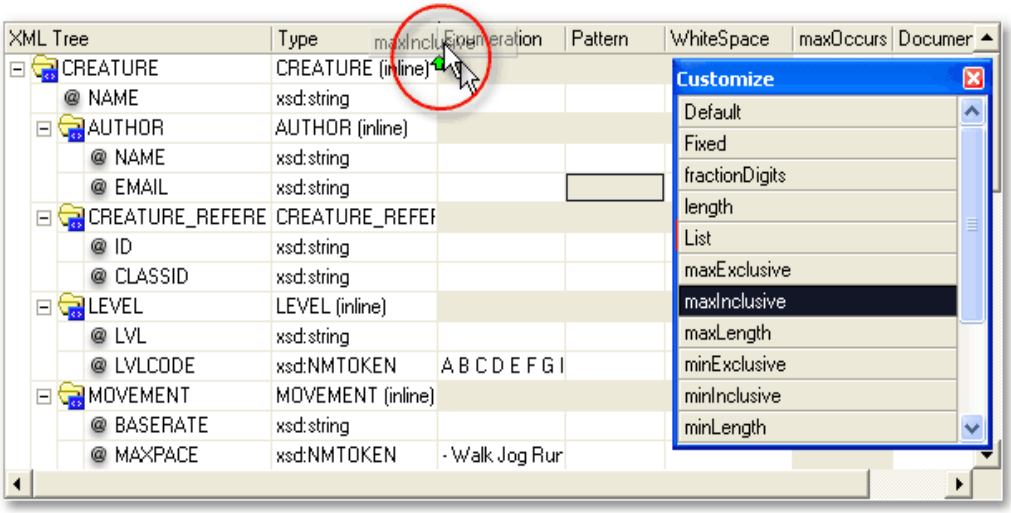
Ένα σχήμα ορίζει πολλαπλά XML έγγραφα



Το xmlDraft προσφέρει έναν εύκολο τρόπο να διατηρήσουμε μία απλή εικόνα του εγγράφου, και ταυτόχρονα επιτρέπει έναν τρόπο απεικόνισης διαφορετικών εγγράφων, τόσων όσων το σχήμα ορίζει. Όταν το σχήμα ορίζει περισσότερα από ένα έγγραφα, ένα drop-down βελάκι θα εμφανιστεί περιέχοντας τα ονομαστικά στοιχεία του κάθε εγγράφου. Όταν ο χρήστης επιλέξει ένα διαφορετικό έγγραφο το παράθυρο δενδροειδούς απεικόνισης δείχνει εκείνο το έγγραφο.

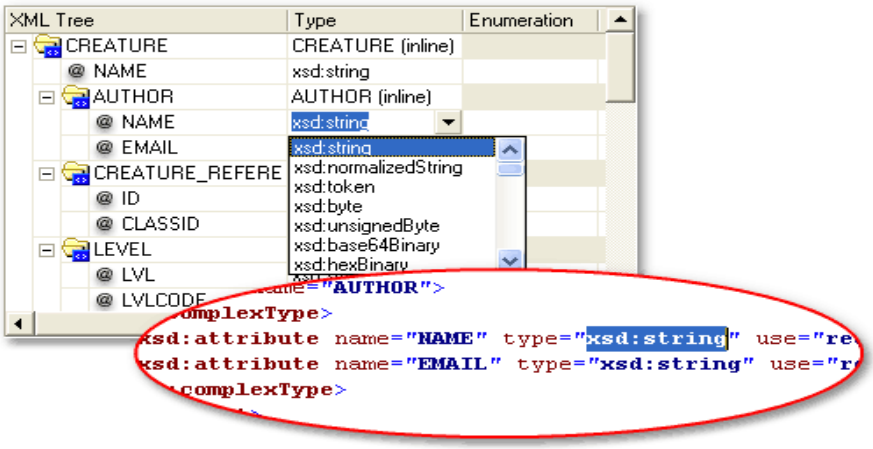
Ιδιότητες κόμβου στη δενδροειδή απεικόνιση

Στη δενδροειδή απεικόνιση, εκτός από τους κόμβους φαίνονται και οι ιδιότητες του κάθε κόμβου ξεχωριστά. Τύπος, μέγεθος, περιορισμοί κλπ.



Παράδειγμα δενδροειδούς απεικόνισης με όσες στήλες επιθυμεί ο χρήστης. Όταν όλες οι ιδιότητες ενός κόμβου είναι εμφανείς, γίνεται δύσκολη η χρήση, έτσι το xmlDraft προσφέρει την δυνατότητα να επιλέξουμε ποιες ιδιότητες του κόμβου θα εμφανίζονται.

Επιπλέον το xmlDraft προσφέρει τη δυνατότητα να κάνουμε άμεσες αλλαγές στην απεικόνιση αυτή και έτσι να έχουμε ένα σχήμα δυναμικά ενημερωμένο. Μετονομάζοντας έναν κόμβο, αλλάζοντας τον τύπο του, μπορούμε να προσθέσουμε ή να αφαιρέσουμε κόμβους, επιπρόσθετοι περιορισμοί όπως πχ για το εύρος τιμών, ενημέρωση τεκμηρίωσης, ακόμη και η μετακίνηση κόμβων είναι επιτρεπτά και πλήρως υποστηριζόμενα από την δενδροειδή απεικόνιση, κάνοντας τις απαραίτητες αλλαγές στο κείμενο/ κώδικα του σχήματος. Το ίδιο ισχύει και αντιστρόφως, οποιαδήποτε αλλαγή γίνει στο κείμενο, ενημερώνεται αυτόματα και η δενδροειδή απεικόνιση.



Όπως φαίνεται και παραπάνω, αλλάζοντας τον τύπο στην δενδροειδή απεικόνιση ενημερώνεται αμέσως και ο τύπος στο κείμενο. Στην πραγματικότητα θα μπορούσαμε να δουλέψουμε εξολοκλήρου στη δενδροειδή απεικόνιση χωρίς να χρειαστεί να πειράξουμε τον κώδικα του σχήματος.

Εξάλειψη πολυπλοκότητας

Το xmlDraft εξαφανίζει την πιθανή πολυπλοκότητα των σχημάτων, όσον αφορά στην κατανόηση του σχήματος αυτού καθαυτού. Επίσης, είναι εύκολο στην κατανόηση του πώς το XML έγγραφο θα πρέπει να είναι μορφολογικά για το σχήμα το οποίο σχετίζεται. Επιπλέον μπορεί να μειώσει την ταχύτητα μάθησης σύνταξης και κατανόησης των σχημάτων.

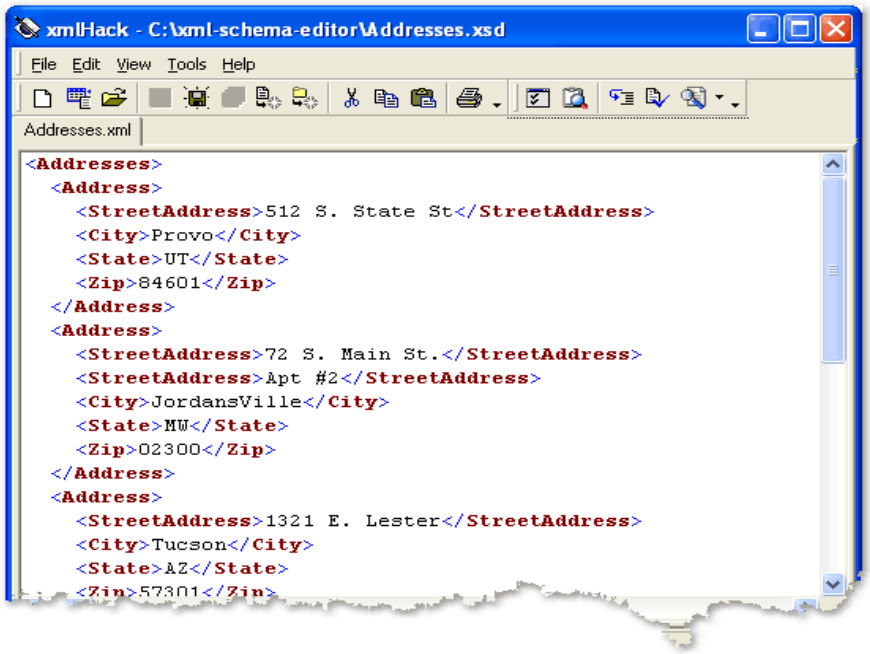
Εισαγωγή και Εξαγωγή XML σε/ από σχήματα

Ένα σημαντικό χαρακτηριστικό ενός XML Schema editor είναι η ικανότητα του να εισάγει ένα έγγραφο και να δημιουργεί το σχήμα του εγγράφου για μας. Επιπλέον, θα ήταν χρήσιμο να μπορούμε να

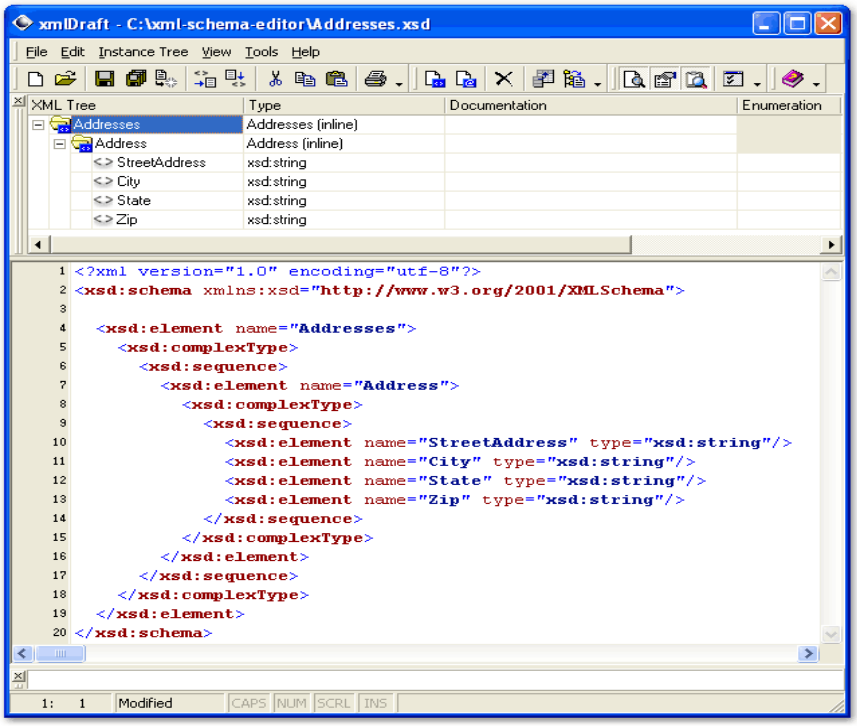
φτιάξουμε ένα δείγμα εγγράφου από ένα σχήμα. Το xmlDraft κάνει και τις δύο αυτές ενέργειες.

Η εισαγωγή ενός XML εγγράφου

Το xmlDraft επιτρέπει στον χρήστη να εισάγει οποιοδήποτε XML έγγραφο και να δημιουργήσει ένα βασικό XML σχήμα. Για παράδειγμα, εάν έπρεπε να εισάγουμε το ακόλουθο έγγραφο



Τότε το σχήμα που θα προέκυπτε θα ήταν



Το έγγραφο είναι αρκετά απλό. Το σχήμα που προκύπτει αυτόματα είναι δείγμα σχήματος, αλλά είναι αρκετό να αρχίσουμε να δουλεύουμε με αυτό.

Η εξαγωγή ενός δείγματος XML εγγράφου

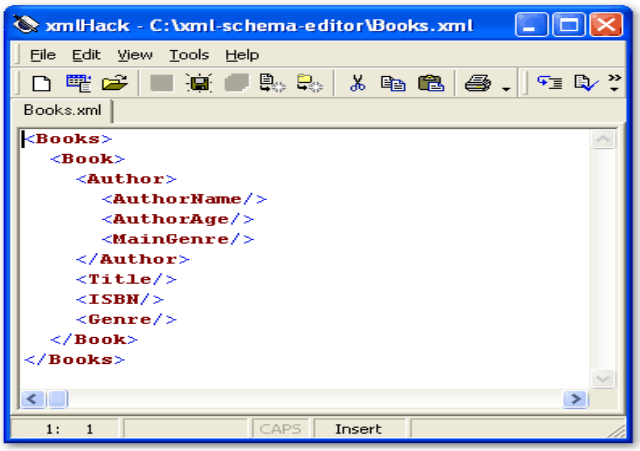
Εκτός από την εισαγωγή ενός XML εγγράφου, υπάρχουν αρκετές φορές που θα ήταν καλό να έχουμε ένα δείγμα του εγγράφου ενός συγκεκριμένου σχήματος. Για παράδειγμα, όταν αναπαράγουμε ένα Stylesheet (είναι οι οδηγίες για τη διάταξη και τα χρώματα του εγγράφου), τυπικά εργαζόμαστε σε ένα “template”, εννοώντας ένα έγγραφο το οποίο δεν περιέχει δεδομένα, μόνο τους κόμβους. Εάν έχουμε ένα σχήμα και μπορούσαμε να έχουμε ένα δείγμα εγγράφου, τότε θα λέγαμε ότι αυτό είναι ένα καλό χαρακτηριστικό. Το xmlDraft το κάνει αυτό. Ας δούμε το παρακάτω σχήμα:

```xml
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="Books">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Author" type="AuthorType"/>
              <xsd:element name="Title" type="xsd:string"/>
              <xsd:element name="ISBN" type="xsd:string"/>
              <xsd:element name="Genre">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Sci-Fi"/>
                    <xsd:enumeration value="Fantasy"/>
                    <xsd:enumeration value="Mystery"/>
                    <xsd:enumeration value="Romance"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="AuthorType">
    <xsd:sequence>
      <xsd:element name="AuthorName" type="xsd:string"/>
      <xsd:element name="AuthorAge" type="xsd:string"/>
      <xsd:element name="MainGenre" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Είναι ένα απλό σχήμα το οποίο αντιπροσωπεύει ένα βιβλίο. Ας σημειωθεί ότι υπάρχει μια αναφορά καθολικής μεταβλητής σύνθετου τύπου. Όταν το εξάγουμε δημιουργείται το ακόλουθο έγγραφο:



Μπορεί να είναι απλό, αλλά προσφέρει ένα καλό σημείο εκκίνησης να αρχίσουμε να εργαζόμαστε με αυτό. Είναι πολύ χρήσιμο επίσης όταν χειριζόμαστε μεγαλύτερα και πιο πολύπλοκα σχήματα.

Τεκμηρίωση XML και xmlDraft

Ένα από τα πλεονεκτήματα του xmlDraft σχετικά με τα XML σχήματα είναι οι μέθοδοι τεκμηρίωσης που περιλαμβάνει. Τα σχήματα χειρίζονται την τεκμηρίωση μέσα από τον <σχολιασμό(annotation)> του κόμβου. Διαθέτει δύο μεθόδους τεκμηρίωσης, μία για χρηστική ευαναγνωσιμότητα που λέγεται «τεκμηρίωση» και μία που προορίζεται να είναι ευέλικτη. Το xmlDraft εστιάζει στην τεκμηρίωση κόμβου και έχει έναν εύκολο τρόπο να προσθέσουμε τεκμηρίωση σε ένα σχήμα.

<annotation> node in XMLSchema

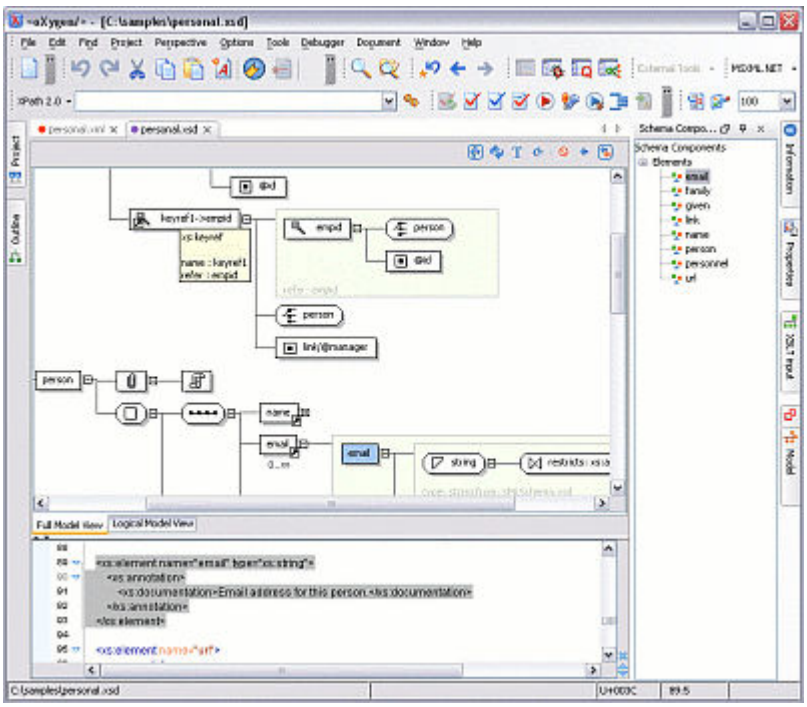
Ο σχολιασμός(annotation) κόμβου είναι ένα σχήμα από μόνο του για να υποστηρίξει την τεκμηρίωση.

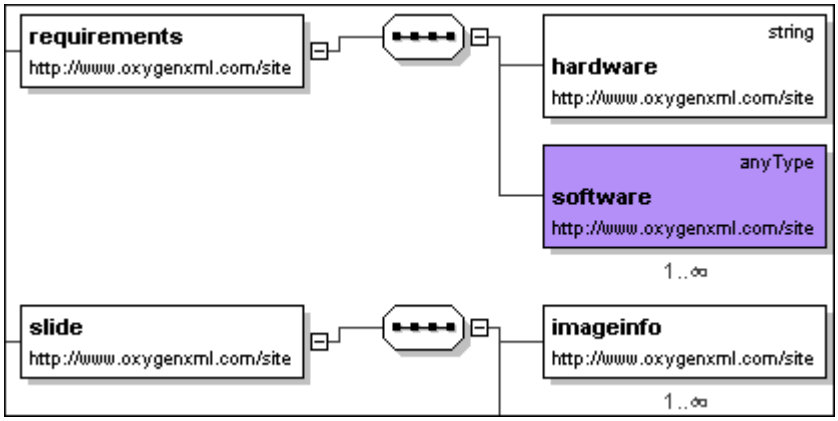
8.2 oXygen

Με το oXygen διάγραμμα σχημάτων απλοποιεί την δημιουργία και κατανόηση των αρχείων XML σχημάτων. Το oXygen προσφέρει μια παρουσίαση της πηγής και του διαγράμματος σχημάτων. Το διάγραμμα είναι συγχρονισμένο σε πραγματικό χρόνο με τον βασικό editor σχημάτων. Με την επιλογή ενός στοιχείου στο διάγραμμα δίνεται έμφαση στο αντίστοιχο στοιχείο στον editor, με αποτέλεσμα οποιαδήποτε αλλαγή στον editor να επιφέρει αλλαγή και στο διάγραμμα του σχήματος.

Το διάγραμμα σχημάτων αποδίδει όλα τα στοιχεία του σχήματος XML και μας δίνει τη δυνατότητα να κάνουμε μια γρήγορή πλοήγηση

στους ορισμούς των στοιχείων, των ιδιοτήτων, των τύπων, των ομάδων, των σχεδίων, κ.λπ.





Δύο τύποι οπτικής απεικόνισης διαγραμμάτων είναι διαθέσιμοι για ένα σχήμα: η πλήρης και η λογική απεικόνιση.

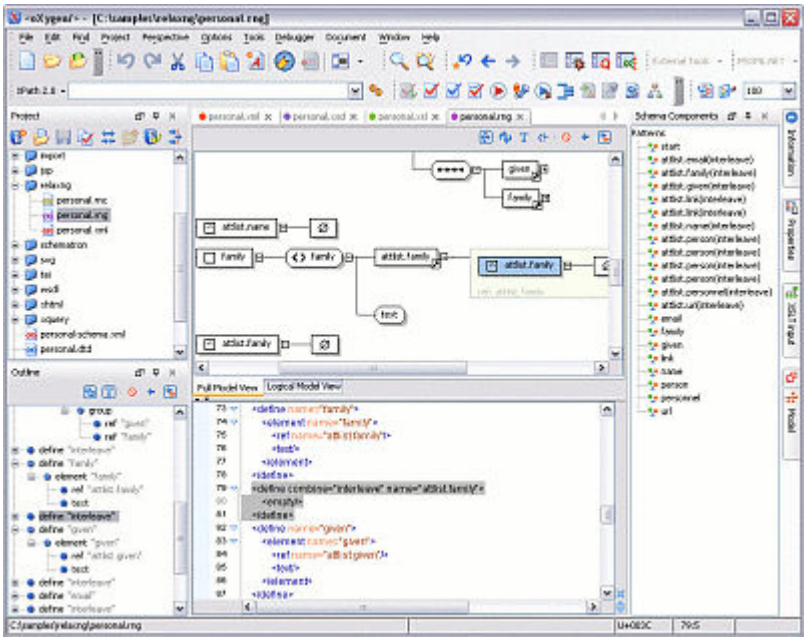
Πλήρης απεικόνιση. Η πλήρης απεικόνιση παρέχει μια ένα προς ένα επικοινωνία μεταξύ των τμημάτων των σχημάτων και των κόμβων

του γραφήματος. Οι αναφορές στα διάφορα συστατικά του σχήματος μπορούν να επεκτείνουν την ισχύ τους στο διάγραμμα (για παράδειγμα τις αναφορές στοιχείων ή ιδιοτήτων).

Λογική απεικόνιση. Η λογική απεικόνιση μας δείχνει ένα συμπαγέστερο διάγραμμα που αποκτάται με την επίλυση των αναφορών, των επεκτάσεων και των περιορισμών των τύπων, των επαναπροσδιορισμών κ.λπ. για το σχήμα XML με την εφαρμογή των κανόνων απλοποίησης.

Ένας κατάλογος με τα στοιχεία του σχήματος (στοιχεία, ιδιότητες, κ.λπ.) που παρουσιάζονται στο τμήμα Απεικόνιση Στοιχείων σε συνδυασμό με την περιληπτική προβολή του σχήματος απλοποιεί την πλοήγηση σε μεγάλα και πολύπλοκα σχήματα.

Ο Visual Schema Editor (Οπτικός Συντάκτης Σχημάτων) είναι ενσωματωμένος στον oXygen Editor και ενεργοποιείται κατά το άνοιγμα ενός XSD (σχήμα XML).

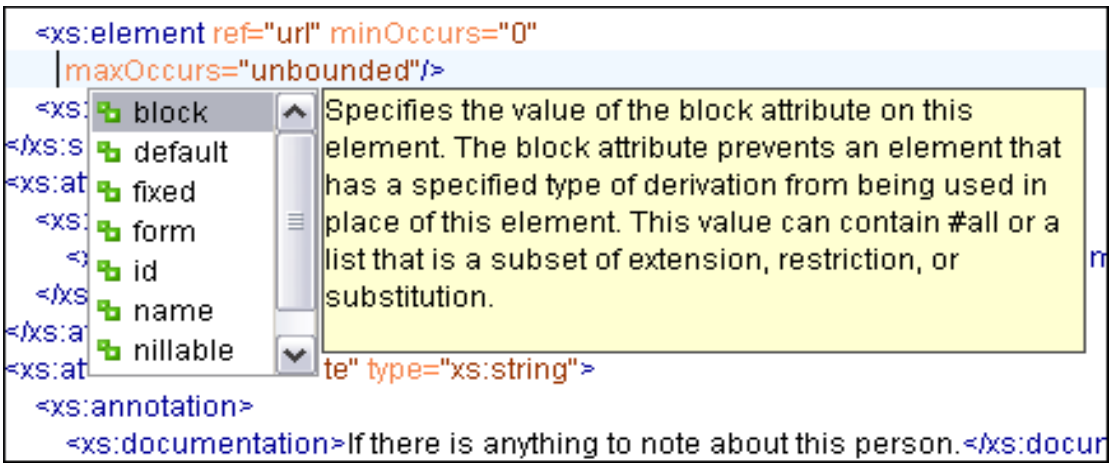


Οι ενέργειες που είναι διαθέσιμες στην πλήρη απεικόνιση επιτρέπουν την προσθήκη νέων στοιχείων, χωρίς όμως να θέσουν σε κίνδυνο την εγκυρότητα του σχήματος:

Υποστήριξη ζουμ στο διάγραμμα. Τα διαγράμματα μπορούν να γίνουν μικρότερα ή μεγαλύτερα ανάλογα με τις ρυθμίσεις ζουμ.

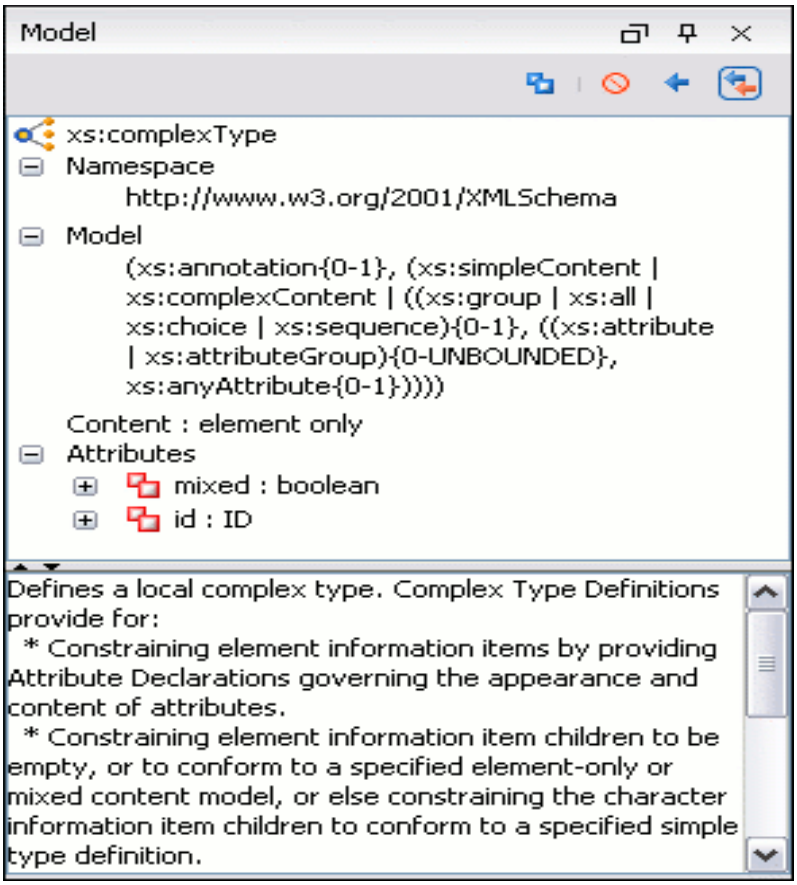
Υποστήριξη σχημάτων XML. W3C τα έγγραφα σχημάτων XML μπορούν να συνταχθούν και να επικυρωθούν με τον oXygen Editor.

Η ολοκλήρωση του περιεχομένου επιδεικνύει την τεκμηρίωση των σχημάτων.



Η τεκμηρίωση σχημάτων XML παρουσιάζεται από τη νέα υποστήριξη ολοκλήρωσης περιεχομένου. Η ολοκλήρωση περιεχομένου προσφέρει πρόσθετες πληροφορίες για τα στοιχεία και τις ιδιότητες κατά τρόπο απλό και κομψό.

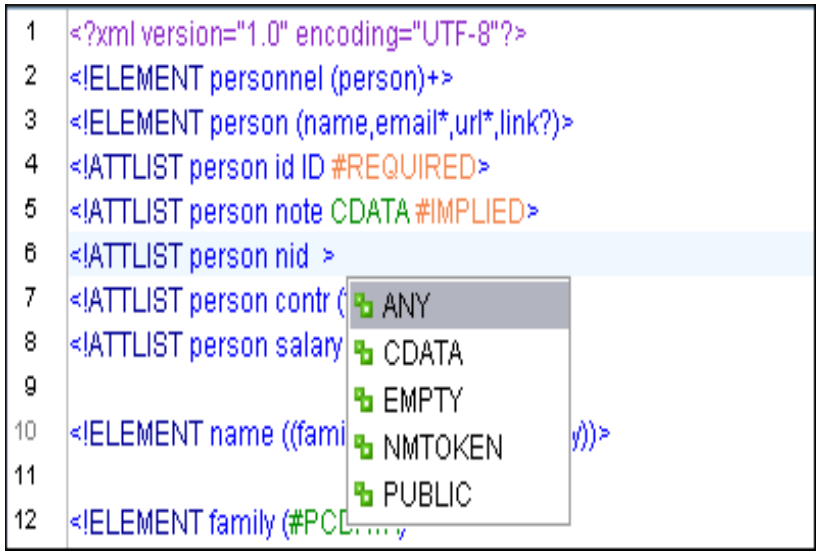
Πρότυπη απεικόνιση του σχήματος



Το πρότυπο αυτό μας παρέχει πληροφορίες (σχετικές με το σχήμα) για τα στοιχεία και τις ιδιότητες εγγράφων. Αυτές παρουσιάζονται κατά τρόπο συμπαγή, δίνοντας μας έτσι μια καλύτερη εικόνα του σχήματος εγγράφων, καθώς το συντάσσουμε.

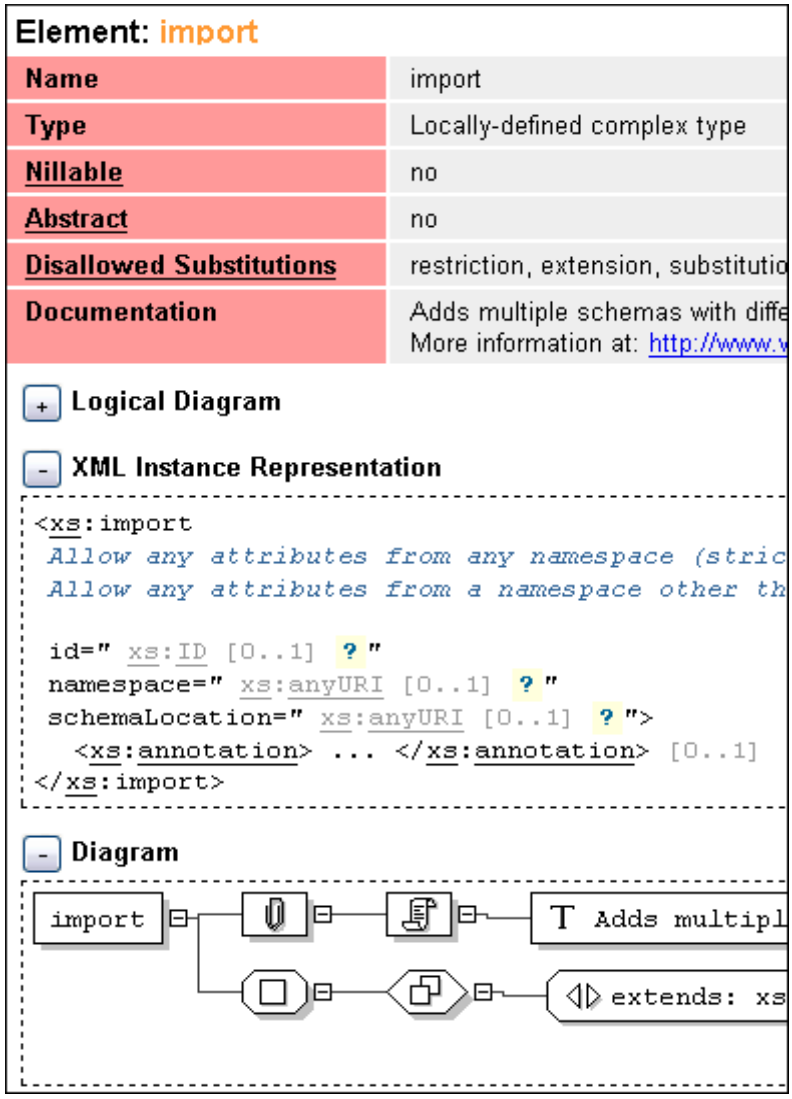
Υποστήριξη DTD.

Το DTDs μπορεί να εκδοθεί και να επικυρωθεί με τον oXygen Editor, που υποστηρίζει τον χρωματισμό και την επικύρωση σύνταξης.

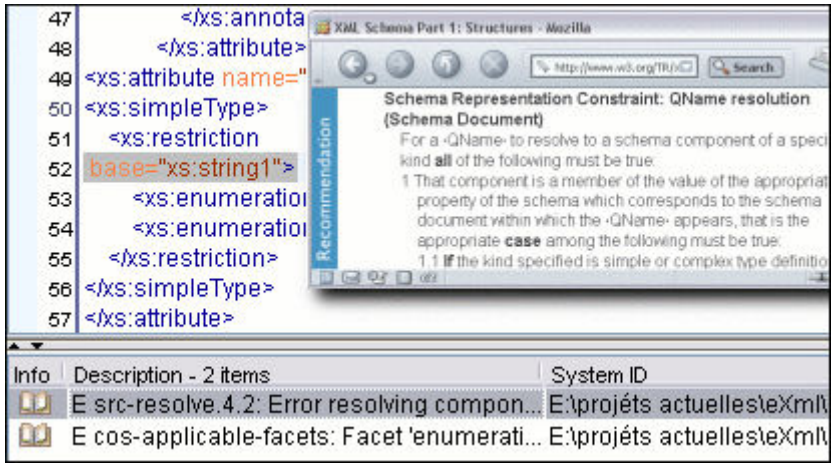


Ενισχυμένη τεκμηρίωση XML σχημάτων.

Είναι εφικτό να παράγουμε σελίδες τεκμηρίωσης HTML από τα αρχεία σχημάτων XML. Η τεκμηρίωση περιλαμβάνει τις εικόνες που αντιπροσωπεύουν τα πρότυπα για τα στοιχεία, ιδιότητες, τύπους κ.λ.π

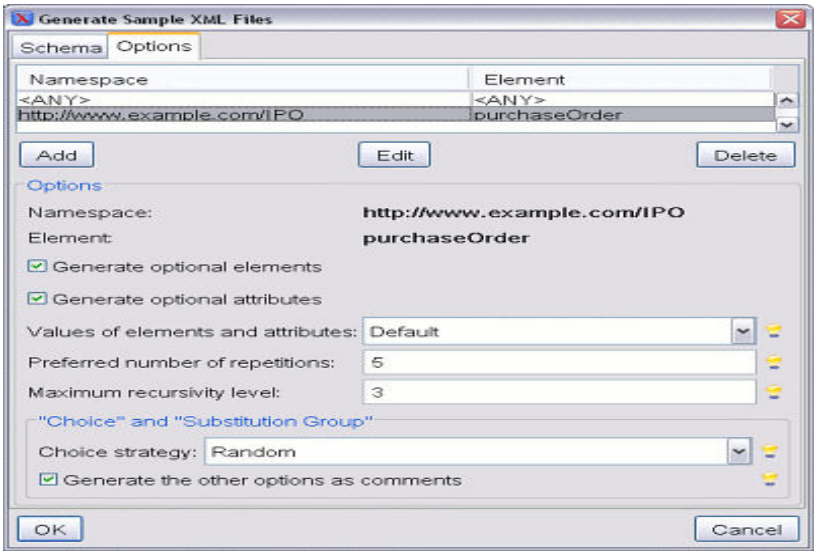


Διαδικτυακή Σύνδεση για τον προσδιορισμό συντακτικών σφαλμάτων σε σχήματα XML.



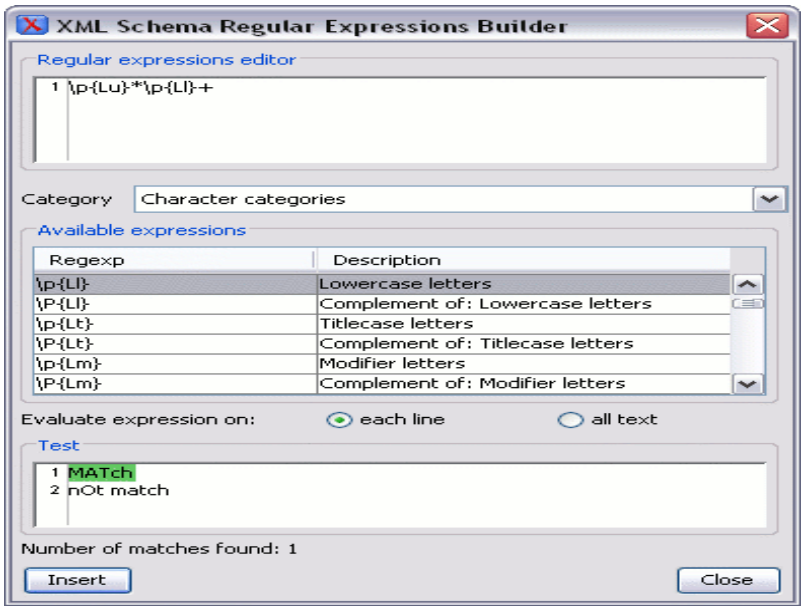
Γεννήτρια δειγμάτων εγγράφων σχημάτων XML.

Η γεννήτρια παραδειγμάτων σχημάτων XML μπορεί γρήγορα να παραγάγει ένα μεγάλο σύνολο δειγμάτων εγγράφων XML βασισμένων σε ένα δεδομένο σχήμα XML.



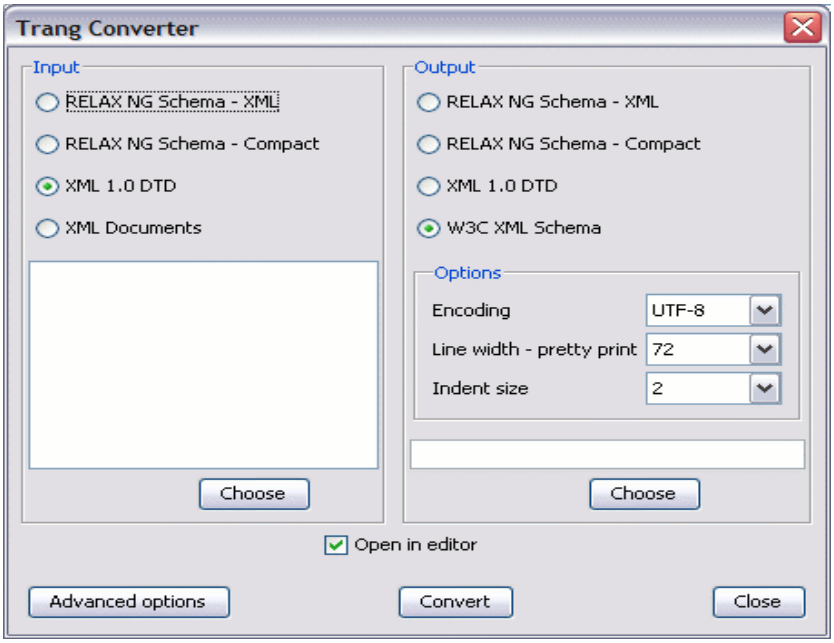
Δημιουργός κανονικών εκφράσεων XML σχημάτων.

Ο δημιουργός κανονικών εκφράσεων βοηθά στην κατασκευή και τον έλεγχο των κανονικών εκφράσεων που επιτρέπονται από τα μοτίβα περιορισμού των XML σχημάτων.



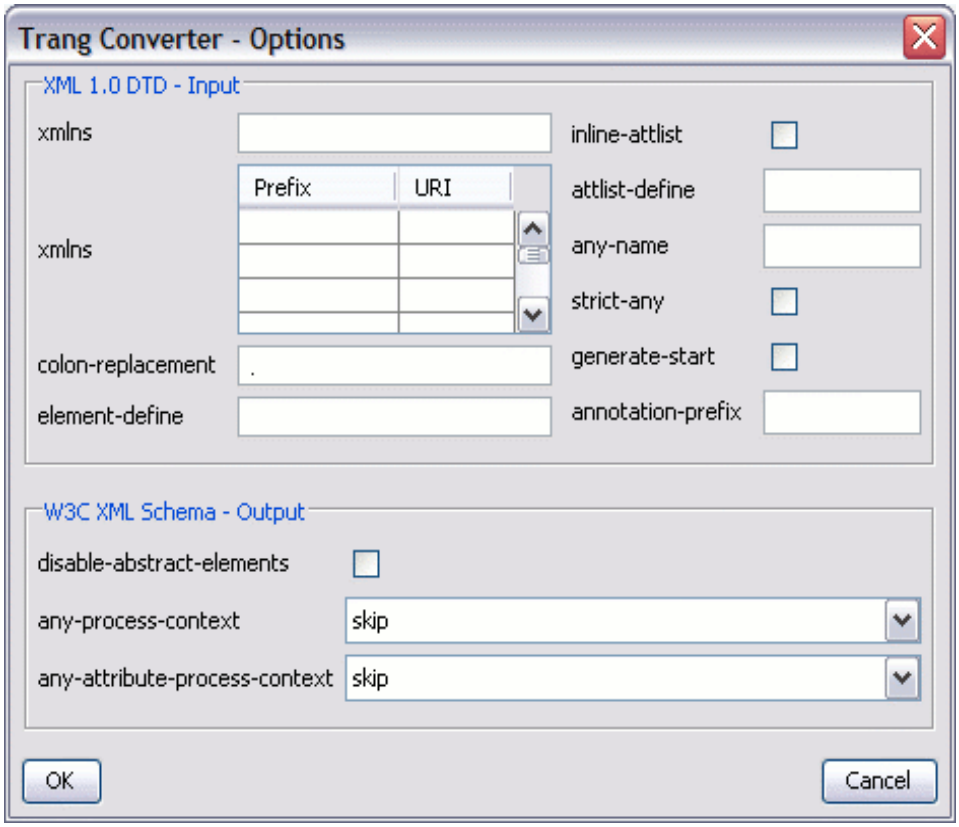
Αναζητώντας τις Αναφορές και τις Διακηρύξεις των συστατικών μερών των XML σχημάτων. Μπορούμε να βρούμε είτε τις δηλώσεις είτε τις αναφορές των τμημάτων σχημάτων XML. Το πεδίο αναζήτησης μπορεί να είναι το τρέχον αρχείο, το τρέχον πρόγραμμα ή ομάδα αρχείων.

Μετατροπή μεταξύ των γλωσσών γραμματικής. Ο μετατροπέας επιτρέπει να μετατρέψουμε ένα DTD ή ένα σύνολο αρχείων XML σε ένα ισοδύναμο XML σχήμα.



Μπορούμε να θέσουμε την επιθυμητή γλώσσα μετατροπής και του επιθυμητού ονόματος.

Οι επιλογές για προχωρημένους επιτρέπουν στον χρήστη να προσαρμόσει ποικίλες παραμέτρους μετατροπής.

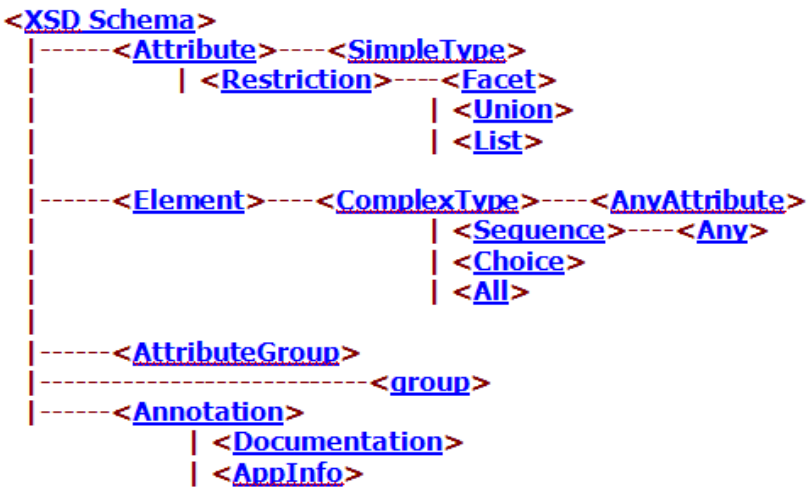


8.3 XMLFox

Ο XMLFox είναι ένας ακόμη editor XML σχημάτων, αλλά και συντάκτης του γενικότερου φάσματος της XML. Ο XMLFox αντιμετωπίζει το XML σχήμα σαν διαγραμματική απεικόνιση μιας αφηρημένης δομής ενός συνόλου στοιχείων.

Διευκρινίζει τη διάταξη των ετικετών στο έγγραφο XML, δείχνει τους τομείς που είναι υποχρεωτικοί ή που μπορούν να εμφανιστούν σε διαφορετικούς χρόνους, δίνει τα datatypes των τομέων και τα λοιπά. Το σχήμα είναι σε μεγάλο βαθμό ικανό να διασφαλίσει ότι οι τιμές στοιχείων στο αρχείο XML ισχύουν όσον αφορά στην εφαρμογή γονέων. Όπως

μας δείχνει και η παρακάτω εικόνα σε δενδρική απεικόνηση, τα συστατικά μέρη του XMLFox σχήματος, αλλά και γενικά του XML σχήματος, είναι:



Η γλώσσα καθορισμού σχημάτων XML (XSD) είναι η τρέχουσα τυποποιημένη γλώσσα σχημάτων για όλα τα έγγραφα και στοιχεία XML.

Η γλώσσα καθορισμού σχημάτων XML (XSD) επιτρέπει στους χρήστες να καθορίσουν τους τύπους δομών και στοιχείων για τα έγγραφα XML. Ένα σχήμα XML καθορίζει τα στοιχεία και τις ιδιότητες των δομών αυτών. Η αναφορά σχημάτων XML (XSD) είναι βασισμένη στις Προδιαγραφές Σύστασης W3C 2001 για τους τύπους δεδομένων και για τις δομές.

Ένα σχήμα XML αποτελείται από το κορυφαίο στοιχείο σχημάτων. Ο καθορισμός στοιχείων σχημάτων πρέπει να περιλάβει το ακόλουθο namespace: http://www.w3.org/2001/XMLSchema

Το στοιχείο σχημάτων περιέχει τους ορισμούς τύπων (στοιχεία simpleType και complexType) και τις δηλώσεις ιδιοτήτων και στοιχείων. Εκτός από τους ενσωματωμένους τύπους στοιχείων του (όπως ο ακέραιος αριθμός, η σειρά, και τα λοιπά), το σχήμα XML επιτρέπει επίσης τον καθορισμό των νέων τύπων στοιχείων χρησιμοποιώντας τα στοιχεία simpleType και complexType.

<Attribute> Ιδιότητες

Οι δηλώσεις ιδιοτήτων μπορούν να είναι παρούσες ως στοιχεία παιδιών του βασικού στοιχείου σχημάτων ή μέσα στους σύνθετους ορισμούς τύπων. Για τους σύνθετους τύπους, οι δηλώσεις ιδιοτήτων μπορούν να είναι παρούσες ως τοπικές δηλώσεις ή αναφορές στις ιδιότητες με σφαιρικό πεδίο ορισμού. Και οι σφαιρικές και τοπικές δηλώσεις ιδιοτήτων έχουν την προαιρετική ιδιότητα τύπων που αναφέρεται σε έναν υπάρχοντα απλό τύπο. Εάν η προαιρετική ιδιότητα τύπων δεν χρησιμοποιείται, η δήλωση ιδιοτήτων (σφαιρική ή τοπική) πρέπει να καθορίσει έναν τοπικό απλό τύπο.

Παραδείγματα:

1. Στο ακόλουθο παράδειγμα, μια ιδιότητα δηλώνεται από την αναφορά σε έναν ενσωματωμένο τύπο με μια προκαθορισμένη αξία και χρησιμοποιείται σε ένα στοιχείο complexType.

<xs:attribute name="MyAttribute" type="xs:string"

```
                default="test" />
    <xs:complexType name="myComplexType">
    <xs:attribute ref="MyAttribute"/>
    </xs:complexType>
```

2. Στο ακόλουθο παράδειγμα, μια απαραίτητη ιδιότητα δηλώνεται άμεσα μέσα σε ένα στοιχείο complexType.

```
    <xs:complexType name="myComplexType">
    <xs:attribute name="MyAttribute" type="xs:string"
                use="required"/>
    </xs:complexType>
```

3. Στο ακόλουθο παράδειγμα, μια ιδιότητα δηλώνεται με την προέλευση από τον ενσωματωμένο τύπο ακέραιων αριθμών (από τον περιορισμό) και τον περιορισμό της σειράς των τιμών μεταξύ "50" και "101", συμπεριλαμβάνουσας.

```
    <xs:attribute name="RightAge">
    <xs:simpleType>
    <xs:restriction base="xs:integer">
    <xs:minInclusive value="50"/>
    <xs:maxInclusive value="101"/>
    </xs:restriction>
    </xs:simpleType>
    </xs:attribute>
```

4. Στο ακόλουθο παράδειγμα, μια ιδιότητα δηλώνεται ως κατάλογος που περιέχει δεκαδικές τιμές. (Αυτό επιτρέπει σε μια ιδιότητα όπως PatientVital="120 70 8 11" να εμπεριέχει έναν κατάλογο των τιμών 120 ..70 ..8, και 11).

```
<xs:simpleType name="Vitals">
<xs:restriction base="xs:decimal">
<xs:enumeration value="120"/>
<xs:enumeration value="70"/>
<xs:enumeration value="8"/>
<xs:enumeration value="11"/>
</xs:restriction>
</xs:simpleType>
<xs:attribute name="PatientVital">
<xs:simpleType>
<xs:list itemType="Vitals"/>
</xs:simpleType>
</xs:attribute>
```

<simpleType>

Ένα στοιχείο σχημάτων XSD μπορεί να είναι των ακόλουθων τύπων:

- Απλός τύπος
- Σύνθετος τύπος, τύπος καθοριζόμενος από τον χρήστη.

Το simpleType είναι ένας τύπος ορισμού για μια τιμή που μπορεί να χρησιμοποιηθεί ως περιεχόμενο (textOnly) ενός στοιχείου ή μιας ιδιότητας. Αυτός ο τύπος δεδομένων δεν μπορεί να συμπεριλάβει στοιχεία ή να έχει ιδιότητες.

Παραδείγματα:

1) Ορισμοί απλών τύπων που χρησιμοποιούν ***Περιορισμούς***

```
<xsd:simpleType name="Amount">
<xsd:restriction base="xsd:integer">
<xsd:restriction base="xsd:integer">
<xsd:minInclusive value="1"/>
<xsd:maxInclusive value="999"/>
</xsd:restriction>
</xsd:simpleType>
```

2) Ορισμοί απλών τύπων που χρησιμοποιούν ***Λίστα***

```
<xs:simpleType name="CalendarDates">
<xs:list itemType="xs:date">
</xs:simpleType>
```

3) Ορισμοί απλών τύπων που χρησιμοποιούν ***Ένωση***

```
<xs:schema
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:attribute name="allMonitorSize">
```

```
<xs:simpleType>
<xs:union>
<xs:simpleType>
<xs:restriction base="CRTmonitorSize"/>
</xs:simpleType>
<xs:simpleType>
<xs:restriction base="LCDmonitorSize"/>
</xs:simpleType>
</xs:union>
</xs:simpleType>
</xs:attribute>
<xs:simpleType name="CRTmonitorSize">
<xs:restriction base="xs:positiveInteger">
<xs:enumeration value="15"/>
<xs:enumeration value="17"/>
<xs:enumeration value="19"/>
<xs:enumeration value="21"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="LCDmonitorSize">
<xs:restriction base="xs:string">
<xs:enumeration value="small"/>
<xs:enumeration value="medium"/>
```

<xs:enumeration value="large"/>

</xs:restriction>

</xs:simpleType>

</xs:schema>

<Facet> Άποψη

Το *Facet* είναι ένα χαρακτηριστικό ενός τύπου στοιχείων που μπορούμε να χρησιμοποιήσουμε για να περιορίσουμε τις τιμές ενός τύπου. Επιτρέπει τις συντομεύσεις στην δημιουργία απλών τύπων με τον περιορισμό ενός άλλου τύπου δεδομένων.

Κάθε Facet ορίζεται ως ένα στοιχείο, καθένα από τα οποία έχει μια σταθερή ιδιότητα που είναι μια τιμή Boolean. Όλες μαζί τα facet για έναν απλό τύπο καθορίζουν το σύνολο νόμιμων τιμών για αυτόν τον απλό τύπο.

Η συντακτική δήλωση ενός facet είναι:

<xsd:facetName value="facetValue">

Παράδειγμα:

```
<xsd:simpleType name="simpleTypeName">
 <xsd:restriction base="xsd:typeName">
  some facet statements
 </xsd:restriction>
</xsd:simpleType>
<xsd:element name="elementName" type="simpleTypeName"/>
```

elementName – Το όνομα του στοιχείου XML.

xsd:typeName – Ένας προκαθορισμένος τύπος δεδομένων που χρησιμοποιείται ως ο βασικός τύπος δεδομένων.

simpleTypeName – Νέος τύπος, επέκταση του βασικού τύπου δεδομένων.

Προαιρετικές facet δηλώσεις μπορεί να περιλαμβάνονται στην περιοριστική δήλωση με σκοπό να μας εφοδιάσουν με πρόσθετους περιορισμούς σε εκτεταμένους τύπους δεδομένων.

xsd:enumeration – έγκυρη τιμή. Η επανάληψη αυτής της δήλωσης παρέχει πολλαπλές τιμές

xsd:minInclusive – Συμπεριλαμβάνοντας τις ελάχιστες τιμές

xsd:minExclusive – Αποκλείοντας τις ελάχιστες τιμές

xsd:maxInclusive - Συμπεριλαμβάνοντας τις μέγιστες τιμές

xsd:maxExclusive - Αποκλείοντας τις μέγιστες τιμές

xsd:pattern – Μία κανονική έκφραση στην οποία να ταιριάζει η τιμή

xsd:length – Μέγεθος χαρακτήρων

xsd:minLength – Ελάχιστο μέγεθος χαρακτήρων

xsd:maxLength – Μέγιστο μέγεθος χαρακτήρων

xsd:totalDigits – Μέγιστος αριθμός ψηφίων

xsd:fractionDigits - Μέγιστος αριθμός ψηφίων στο κλάσμα

xsd:whiteSpace – Διευκρινίζει τη διαχείριση των κενών διαστημάτων: αντικατάσταση - διατήρηση – κατάργηση

<Element> Στοιχείο

Ο όρος «Στοιχείο» είναι το θεμελιώδες στοιχείο κάθε σχήματος XML. Ένα στοιχείο καθορίζει μια οντότητα σε ένα αρχείο XML. Πώς να διευκρινίσουμε το στοιχείο με το απλό περιεχόμενο:

Στοιχείο με ενσωματωμένο τύπο:

<xsd:element name="Numbers" type="xsd:positiveInteger"/>

Στοιχείο μέσα σε ένα καθορισμένο από το χρήστη simpleType:

<xsd:simpleType name="colors">

<xsd:restriction base="xsd:string">

<xsd:enumeration value="red"/>

<xsd:enumeration value="white"/>

<xsd:enumeration value="black"/>

</xsd:restriction>

</xsd:simpleType>

<xsd:element name="paints" type="colors"/>

Στοιχείο που αντλεί έναν σύνθετο τύπο από έναν απλό τύπο:

<xsd:element name="price">

<xsd:complexType>

<xsd:simpleContent>

<xsd:extension base="xsd:decimal">

<xsd:attribute name="money" type="xsd:string"/>

```
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>
```

<complexType>

Σε ένα σχήμα XSD, εάν ένα στοιχείο περιλαμβάνει ένα ή περισσότερα στοιχεία παιδιών ή εάν ένα στοιχείο περιέχει ιδιότητες, επομένως ο τύπος στοιχείων πρέπει να είναι "Σύνθετος Τύπος".

Το complexType είναι ένας καθορισμός τύπων για τα στοιχεία που μπορούν να περιλάβουν ιδιότητες και στοιχεία. Παραδείγματα:

1) Καθορίζοντας τον τύπο Address= Διεύθυνση

```
<xsd:complexType name="Address" >
<xsd:sequence>
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="street" type="xsd:string"/>
<xsd:element name="city" type="xsd:string"/>
<xsd:element name="state" type="xsd:string"/>
<xsd:element name="zip" type="xsd:decimal"/>
</xsd:sequence>
<xsd:attribute name="country" type="xsd:NMTOKEN"
                    fixed="GR"/>
</xsd:complexType>
```

2) Καθορίζοντας το στοιχείο myAddress έχοντας τον Σύνθετο Τύπο καθορισμένο μέσα στο στοιχείο.

<xs:element name='myAddress'>

<xs:complexType>

<xs:simpleContent>

<xs:extension base='xs:decimal'>

<xs:attribute name='nikas' type='xs:string' />

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

<Sequence>

Η *ακολουθία* XSD απαιτεί την εμφάνιση των στοιχείων που εμφανίζονται σε ρητή ακολουθία μέσα στο περιέχον στοιχείο. Παράδειγμα:

Ο καθορισμός ενός στοιχείου " Colors ", που μπορεί να έχει μηδέν ή περισσότερα από τα ακόλουθα στοιχεία: " black ", " white ", " blue ", στο στοιχείο ακολουθίας.

<xs:element name="Colors">

<xs:complexType>

<xs:sequence minOccurs="0" maxOccurs="unbounded">

<xs:element name="black"/>

<xs:element name="white"/>

<xs:element name="blue"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<Choice>

Το στοιχείο επιλογής XSD επιτρέπει στο αρχείο XML να συμπεριλάβει ένα από τα στοιχεία μέσα στο στοιχείο επιλογής. Παράδειγμα:

Ο σύνθετος τύπος καθορίζει ένα στοιχείο με μία ιδιότητα και ένα και μόνο ένα στοιχείο από μία **επιλογή** από πέντε συγκεκριμένα στοιχεία.

<xs:complexType name="residentState">

<xs:choice minOccurs="1" maxOccurs="1">

<xs:element ref="enable"/>

<xs:element ref="available"/>

<xs:element ref="disable"/>

<xs:element ref="visible"/>

<xs:element ref="hidden"/>

</xs:choice>

<xs:attribute name="resident" type="residentType"/>

</xs:complexType>

<ALL>

Το στοιχείο **All** επιτρέπει τα στοιχεία της ομάδας να εμφανιστούν (ή να μην εμφανιστούν) σε οποιαδήποτε διάταξη στο περιέχον στοιχείο. Όταν ένας σύνθετος τύπος περιλαμβάνει το στοιχείο All, τα αντίστοιχα στοιχεία XML μπορούν να συμπεριλάβουν μερικά ή όλα τα απαριθμημένα στοιχεία και σε οποιαδήποτε διάταξη. Παράδειγμα: Μια ομάδα του στοιχείου “ALL”

```
<xsd:complexType name="OrderType">
<xsd:all>
<xsd:element name="shipTo" type="Address"/>
<xsd:element name="billTo" type="Address"/>
<xsd:element ref="comment" minOccurs="0"/>
</xsd:all>
<xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
```

<AnyAttribute>

Επιτρέπει οποιεσδήποτε ιδιότητες από συγκεκριμένα namespace ή namespaces να εμφανιστούν στο περιέχον στοιχείο complexType.

Παράδειγμα:

```
<xsd:element name="root">
<xsd:complexType>
<xsd:anyAttribute namespace="namespaceName"
```

```
        processContents="skip"/>
    </xsd:complexType>
    </xsd:element>
```

<Any>

Επιτρέπει οποιοδήποτε στοιχείο από συγκεκριμένα namespace ή namespaces να εμφανιστούν στο περιέχον στοιχείο complexType.

Παράδειγμα:

```
<xs:element name='htmltxt'>
<xs:complexType>
<xs:sequence>
<xs:any namespace='http://www.w3.org/1999/xhtml' inOccurs='1'
                maxOccurs='unbounded' processContents='loop'/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

<AttributeGroup>

Ομαδοποιεί ένα σύνολο δηλώσεων ιδιοτήτων έτσι ώστε να μπορούν να ενσωματωθούν ως ομάδα στους σύνθετους ορισμούς τύπων.

Μια ομάδα ιδιοτήτων μπορεί να καθοριστεί μόνο ως παιδί του στοιχείου σχημάτων. Σε αυτήν την περίπτωση, οι ιδιότητες του

ονόματος πρέπει να είναι παρούσες και να περιλαμβάνουν τις ιδιότητες, τα στοιχεία attributeGroup, ή anyAttribute που αποτελούν την ομάδα ιδιοτήτων. Όταν ένα complexType ή ένα στοιχείο attributeGroup περιλαμβάνει μια ομάδα ιδιοτήτων, οι ιδιότητες REF πρέπει να είναι παρούσες και η ιδιότητα του ονόματος δεν επιτρέπεται.

Παράδειγμα:

Το παράδειγμα απεικονίζει μια ομάδα ιδιοτήτων, που καθορίζεται (anAttributeGroup) και χρησιμοποιείται σε σύνθετους τύπους (anElementType).

```
<xs:attributeGroup name="anAttributeGroup">
<xs:attribute name="intAttribute" type="xs:integer"/>
<xs:attribute name="strAttribute" type="xs:string"/>
<xs:attribute name="dateAttribute" type="xs:date"/>
</xs:attributeGroup>
<xs:complexType name="anElementType">
<xs:attributeGroup ref="anAttributeGroup"/>
</xs:complexType>
```

<Group>

Καθορίζει μια ομάδα σε επίπεδο σχημάτων που αναφέρεται στους σύνθετους τύπους. Ομαδοποιεί ένα σύνολο δηλώσεων στοιχείων έτσι ώστε να μπορούν να ενσωματωθούν ως ομάδα σε ορισμούς σύνθετων τύπων.

Παράδειγμα μίας ακολουθίας τριών στοιχείων και μία ομάδα στοιχείων σε έναν ορισμό σύνθετου τύπου.

```
<xs:element name="black" type="xs:string"/>
<xs:element name="white" type="xs:string"/>
<xs:element name="blue" type="xs:string"/>
<xs:attribute name="myAttribute" type="xs:decimal"/>
<xs:group name="groupOfColors">
<xs:sequence>
<xs:element ref="black"/>
<xs:element ref="white"/>
<xs:element ref="blue"/>
</xs:sequence>
</xs:group>
<xs:complexType name="myComplexType">
<xs:group ref="groupOfColors"/>
<xs:attribute ref="myAttribute"/>
</xs:complexType>
```

<Annotation>

Ο σχολιασμός χρησιμοποιείται για να αποθηκεύσει τις πρόσθετες πληροφορίες για το σχήμα ή τα στοιχεία του. Ο σχολιασμός μπορεί να είναι το πρώτο στοιχείο των περισσότερων σχημάτων ή οπουδήποτε κάτω από τα στοιχεία των σχημάτων.

Ένα στοιχείο σχολιασμών μπορεί να περιέχει μια ή περισσότερες περιπτώσεις AppInfo (πληροφορίες για τις εφαρμογές) και τις περιπτώσεις τεκμηρίωσης (σχόλια ή κείμενο). Παράδειγμα:

<xsd:complexType>

<xsd:annotation>

<xsd:documentation xml:lang="en">

Εδώ τοποθετούμε κάποιες λέξεις που εξηγούν τον σκοπό μιας οντότητας XSD.

</xsd:documentation>

</xsd:annotation>

.....

</xsd:complexType>

<Documentation>

Διευκρινίζει τις πληροφορίες που διαβάζονται ή που χρησιμοποιούνται μέσα σε έναν σχολιασμό. Οι πληροφορίες που παρέχονται μέσα στην ετικέτα τεκμηρίωσης δεν χρησιμοποιούνται στην επικύρωση. Παράδειγμα:

<xsd:complexType>

<xsd:annotation>

<xsd:documentation xml:lang="en">

Εδώ τοποθετούμε κάποιες λέξεις που εξηγούν τον σκοπό μιας οντότητας XSD.

</xsd:documentation>

</xsd:annotation>

...../....

</xsd:complexType>

<AppInfo>

Καθορίζει τις συγκεκριμένες πληροφορίες εφαρμογής μέσα σε έναν σχολιασμό.

Παράδειγμα:

<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Country">

<xs:annotation>

<xs:documentation>Country Name</xs:documentation>

<xs:appinfo>Application Information</xs:appinfo>

</xs:annotation>

</xs:element>

</xs:schema>

9. XML ΕΓΓΡΑΦΑ ΚΑΙ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Σε όλες σχεδόν τις ΒΔ δίνεται η δυνατότητα αποθήκευσης των δεδομένων με διάφορους τύπους καθιστώντας έτσι ευκολότερο τον τρόπο χειρισμού τους. Ευρέως χρησιμοποιούμενοι τέτοιοι τύποι είναι οι ακέραιοι αριθμοί (*Integer*), οι πραγματικοί αριθμοί (*Float*, *Double*), τα αλφαριθμητικά (*Varchar*), τύποι για ημερομηνίες και ώρες (*Date*, *Timestamp*), λογικοί τύποι δεδομένων (*Boolean*), τύποι για μεγάλα δυαδικά αρχεία και αρχεία χαρακτήρων (*BLOB*, *CLOB*) και άλλοι που ποικίλουν ανάλογα με το λογισμικό του ΣΔΒΔ. Γίνεται, λοιπόν, εμφανές ότι είναι δυνατό να εκμεταλλευτούμε τα εργαλεία και τις δυνατότητες που προσφέρει η Βάση Δεδομένων για να χειριστούμε μεγάλο όγκο αρχείων.

Έπειτα οι ΒΔ προσφέρουν μια πληθώρα εργαλείων υπό τη μορφή συναρτήσεων που επιδρούν στα δεδομένα αυτά και διευκολύνουν το χειρισμό τους.

Πολλές οργανώσεις και επιχειρήσεις καθιερώνουν διανεμημένα εργασιακά περιβάλλοντα, όπου οι διαφορετικοί χρήστες χρειάζονται να ανταλλάξουν πληροφορίες βασιζόμενοι σε ένα κοινό πρότυπο. Η XML χρησιμοποιείται ευρέως για τη διευκόλυνση αυτής της ανταλλαγής πληροφοριών.

Η επεκτασιμότητα των XML σχημάτων επιτρέπει τη δημιουργία γενικότερων προτύπων, που συγχωνεύουν στοιχεία από διαφορετικές πηγές. Προκειμένου να υποστηριχθεί αυτή η διαδικασία,

χρησιμοποιούμε συστήματα X-Database δεδομένων. Η βάση αυτών των συστημάτων είναι ένα αρχείο XML σχήματος.

Αρχικά, ένα τέτοιο σύστημα αναλύει τη σύνταξη του XML σχήματος και παράγει τη σχεσιακή βάση δεδομένων. Κατόπιν χειρίζεται την αποσύνθεση των έγκυρων αρχείων XML, καθώς και τη σύνθεση των εγγράφων XML από τις πληροφορίες στη βάση δεδομένων. Τέλος το σύστημα προσφέρει έναν ευέλικτο μηχανισμό για την τροποποίηση και εξέταση του περιεχόμενου των βάσεων χρησιμοποιώντας μόνο τα έγκυρα XML έγγραφα, τα οποία επικυρώνονται από τους κανόνες του αρχείου XMLSchema.

Από την XML σε Σχεσιακό Σχήμα

Για να πραγματοποιηθεί η μετάβαση από ένα XML σχήμα σε ένα Σχεσιακό σχήμα, κρίνεται απαραίτητη η χαρτογράφηση των δομών και οντοτήτων του σχήματος XML σε ένα Σχεσιακό σχήμα.

α) Χαρτογράφηση ιδιοτήτων

Κάθε «ιδιότητα» του XML σχήματος χαρτογραφείται στη βάση δεδομένων σε ένα πεδίο ενός πίνακα. Το ακόλουθο παράδειγμα παρουσιάζει έναν τύπο complexType ο οποίος περιέχει ένα σύνολο ιδιοτήτων και την απεικόνιση τους στη βάση.

<xs:complexType name="passport">

<xs:attribute name="id" type="ID" use="required"/>

<xs:attribute name="name" type="string" use="required"/>

</xs:complexType>

CREATE TABLE passport (id NUMBER NOT NULL, name

VARCHAR2(20) NOT NULL, PRIMARY KEY (id));

β) Χαρτογράφηση simpleType

Κάθε simpleType σε ένα XML σχήμα περιέχει μια απαρίθμηση από strings που αντιπροσωπεύουν τις πιθανές τιμές που μπορεί να έχει μια ιδιότητα.

passname VARCHAR2(20) NOT NULL CONSTRAINT CHECK (passname IN ('John', 'Chris', 'Andre'))

γ) Χαρτογράφηση complexType

Κάθε complexType σε ένα XML σχήμα χαρτογραφείται σε έναν χωριστό πίνακα στη βάση. Οι πρόσθετοι πίνακες δημιουργούνται για να απεικονίσουν πολλαπλές σχέσεις μεταξύ των complexTypes.

Κατά τη δημιουργία του XML σχήματος όλες οι φυσικές σχέσεις μεταξύ των διάφορων οντοτήτων έχουν εκφραστεί ως σχέσεις συγκράτησης (containment) ή αναφοράς (reference) μεταξύ των αντίστοιχων σύνθετων τύπων. Επομένως, όταν οι σύνθετοι τύποι

μετατρέπονται σε πίνακες της βάσης, είναι πολύ σημαντικό για τις σχέσεις τους να εκφραστούν σωστά οι περιορισμοί στη βάση δεδομένων.

δ) Ο χειρισμός της κληρονομικότητας των τύπων

Ένα ενδιαφέρον ζήτημα που παρατηρούμε κατά τη διάρκεια της παραγωγής του σχεσιακού σχήματος είναι η συντήρηση της αντικειμενοστρεφούς φύσης της XML. Οι χρήστες πρέπει να είναι σε θέση να έχουν πρόσβαση στο περιεχόμενο της βάσης. Η πρόσβαση αυτή πρέπει να βασίζεται στη λογική δομή των πληροφοριών παρά στο σχήμα της βάσης. Ένα αντικειμενοστρεφές χαρακτηριστικό γνώρισμα του XML σχήματος είναι η κληρονομικότητα μεταξύ των τύπων.

Καθορίζοντας τα χαρακτηριστικά γνωρίσματα σχεσιακής βάσης δεδομένων μέσα στο XML σχήμα.

Οι σχεσιακές βάσεις δεδομένων προσφέρουν πολλά χαρακτηριστικά γνωρίσματα που ελαχιστοποιούν τον αποθηκευτικό χώρο και επιταχύνουν τη διαδικασία ανάκτησης πληροφοριών. Τέτοια χαρακτηριστικά γνωρίσματα δεν υποστηρίζονται άμεσα στο XML σχήμα. Στον καθορισμό ιδιοτήτων μπορούμε να αντικαταστήσουμε τους βασικούς τύπους (δηλ. string, integer κ.λπ.) με simpleTypes που περιέχουν πληροφορίες σχετικά με το μέγεθος ή τη λίστα τιμών κ.λπ.

Προκειμένου να καθορίσουμε δείκτες καταχώρησης σε ένα σύστημα RDBMS πρέπει να ορίσουμε τις ιδιότητες ενός XML σχήματος. Μια απλή λύση αποτελεί η προσθήκη μιας ιδιότητας με το όνομα **indexed** που θα εμφανιστεί και θα φέρει την τιμή «yes» μόνο στις ιδιότητες που πρέπει να καταχωρηθούν.

```
<xsd:attribute name="id" type="xsd:ID"
use="required" db:indexed="yes"/>
```

Εντούτοις, η ιδιότητα **indexed** δεν υποστηρίζεται εξ ορισμού από το XML σχήμα του οργανισμού W3C και δεν μπορεί κανονικά να εμφανιστεί μέσα σε μία ιδιότητα xsd:, έτσι θα πρέπει να την δηλώσουμε σε ένα άλλο ξεχωριστό namespace, πχ **db:indexed**.

Η πολυπλοκότητα των βάσεων δεδομένων

Το κρισιμότερο σημείο X-Database δεδομένων είναι η δημιουργία της βάσης. Πρέπει να αναλυθεί η δομή του XML σχήματος και να δημιουργηθεί ο κατάλληλος αριθμός πινάκων μαζί με τα απαραίτητα ξένα κλειδιά και σκανδάλες (triggers) που θα εγγυηθούν την συνοχή της βάσης στις διαδοχικές εισαγωγές και διαγραφές. Διάφορες παράμετροι της βάσης του σχήματος έχουν ήδη μετρηθεί, όπως ο αριθμός πινάκων, τα ξένα κλειδιά και οι triggers που δημιουργούνται, καθώς επίσης και ο χρόνος δημιουργίας της βάσης για το αρχείο XML σχήματος

διαφορετικής πολυπλοκότητας. Όταν ο αριθμός των complexTypes, ο αριθμός αναφορών και ο αριθμός των επεκτάσεων σε ένα XML σχήμα αυξάνεται, ο αριθμός πινάκων σε μία βάση του σχήματος αυξάνεται αντίστοιχα. Ενδεικτικά αποτελέσματα της αύξησης φαίνονται στο σχήμα που ακολουθεί:

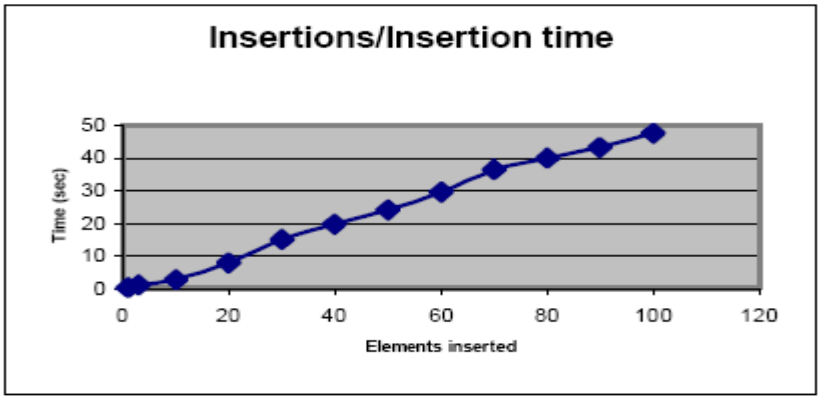
Complex Types	Unbounded refs	Extensions	Tables	Triggers	Foreign keys	Creation time (sec)	Drop time (sec)
68	35	18	140	77	260	52	31
63	31	14	111	51	197	41	20
56	29	14	99	44	172	38	20
50	27	13	88	40	154	31	20
45	26	9	81	36	139	29	11
37	23	8	69	30	93	19	9
28	18	5	50	21	64	15	8
15	9	5	26	10	31	7	4
5	5	0	11	6	12	3	3

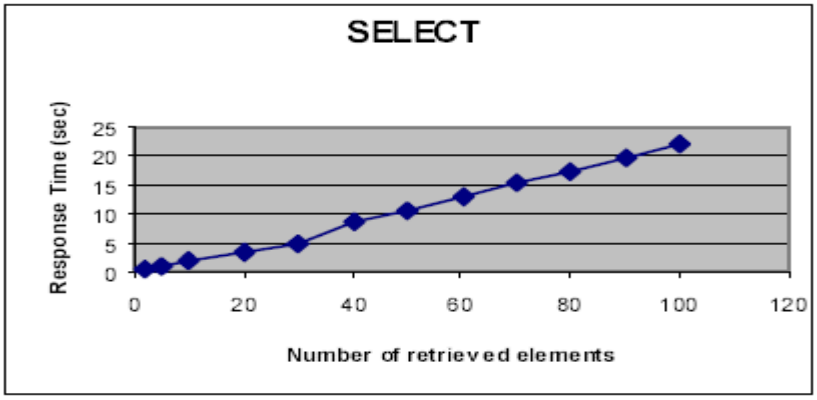
Αριθμός πινάκων συναρτήσει στοιχείων XML σχήματος

Χρόνος Εισαγωγή - Επιλογής

Προκειμένου να εξεταστεί η αποδοτικότητα ενός συστήματός X-Database, πρέπει να μετρηθεί ο απαιτούμενος χρόνος για ένα σύνολο επιλογών και εισαγωγών από τη βάση δεδομένων. Είναι εμφανές στη γραφική απεικόνιση που ακολουθεί ότι ο χρόνος των εντολών εισαγωγής, αλλά και επιλογής είναι ανάλογος με τον αριθμό των ανακτημένων στοιχείων, δεδομένου ότι όλα τα στοιχεία είναι του ιδίου τύπου. Όταν εισάγονται στοιχεία διαφορετικού τύπου, ο αριθμός

επακόλουθων εισαγωγών διαφέρει και η σχέση των στοιχείων αυτών δεν είναι ανάλογη με το χρόνο.





Η Βάση Δεδομένων της IBM DB2

Ανάμεσα σε μια πληθώρα λογισμικών πακέτων ΒΔ η υλοποίηση της IBM DB2 αποτελεί ένα από τα πιο ολοκληρωμένα εργαλεία ανάπτυξης εφαρμογών ΒΔ.

Η DB2 ανήκει στην οικογένεια συστημάτων διαχείρισης σχεσιακών βάσεων δεδομένων της IBM. Υπάρχουν διαφορετικές εκδόσεις της DB2, οι οποίες εκτελούνται από τον πιο απλό υπολογιστή μέχρι και τον πιο σύγχρονο mainframe. Οι διάφορες αυτές εκδόσεις «τρέχουν» σε διάφορα λειτουργικά συστήματα, όπως Windows, Linux, Unix.

Αξίζει όμως να κάνουμε μια μικρή ιστορική αναδρομή, καθώς η DB2 έχει μια μεγάλη ιστορία και θεωρείται από πολλούς το πρώτο προϊόν βάσεων δεδομένων που χρησιμοποίησε SQL, ένα προϊόν που επίσης αναπτύχθηκε πρώτα από τη IBM.

Το όνομα DB2 δόθηκε αρχικά στο σύστημα διαχείρισης βάσεων δεδομένων το 1983 όταν η IBM κυκλοφόρησε την DB2 σε πλατφόρμα κεντρικών υπολογιστών. Η DB2 έχει τις ρίζες της στην αρχή της δεκαετίας του '70 όταν ο E.F. Codd, που εργαζόταν για την IBM, περίγραψε τη θεωρία των σχεσιακών βάσεων δεδομένων και τον Ιούνιο του 1970 δημοσίευσε ένα πρότυπο για το χειρισμό στοιχείων. Για να εφαρμόσει το πρότυπο αυτό, ο Codd χρησιμοποίησε μια γλώσσα σχεσιακής βάσης δεδομένων που την ονόμασε *Alpha*.

Για μερικά έτη η DB2 ήταν αποκλειστικά διαθέσιμη μόνο σε κεντρικούς υπολογιστές της IBM. Αργότερα, τη δεκαετίας του '90, η IBM

μετέφερε την DB2 σε άλλες πλατφόρμες, συμπεριλαμβανομένων των OS/2, UNIX και Windows Servers, έπειτα Linux και PDAs.

Στα μέσα της δεκαετίας του '90, η IBM εξέδωσε μια έκδοση DB2 την οποία ονόμασε DB2 Parallel Edition, η οποία λειτούργησε αρχικά σε AIX. Αυτή η έκδοση έκανε χρήση μιας ιδιαίτερης τεχνικής, όπου μια ενιαία μεγάλη βάση δεδομένων χωρίζεται σε πολλαπλούς κεντρικούς υπολογιστές οι οποίοι επικοινωνούν μεταξύ τους με μια διασύνδεση υψηλής ταχύτητας. Σταδιακά, η έκδοση αυτή υποστηρίζονταν από όλες τις πλατφόρμες Linux, Unix και Windows.

Το 2001, η IBM αγόρασε την εταιρία Informix και στα επόμενα έτη ενσωμάτωσε την τεχνολογία Informix στην γενική πλατφόρμα της DB2. Σήμερα, DB2 μπορεί τεχνικά να θεωρηθεί ένα αντικείμενο-SQL DBMS (Σύστημα Διαχείρισης Βάσεων Δεδομένων).

Στα μέσα του 2006, αναγγέλθηκε από την IBM η έκδοση DB2 9. Η IBM ισχυρίσθηκε ότι η έκδοση αυτή θα είναι η πρώτη που θα υποστηρίζει την αποθήκευση XML στοιχείων σε σχεσιακή βάση δεδομένων.

Η DB2 είναι διαθέσιμη σε διάφορες εκδόσεις, κάποιες από τις οποίες θα αναφέρουμε ονομαστικά: DB2 Data Warehouse Enterprise Edition ή DB2 DWE, DB2 Version 8. Στις 30 Ιανουαρίου του 2006 η IBM εξέδωσε μία μη κερδοσκοπική έκδοση της DB2 με την ονομασία DB2 Express-c. Και ενώ οι εκδόσεις 8.2 και 9.1 της DB2 Express-C θέτουν περιορισμούς σε hardware στον server που εκτελούνται, η

έκδοση DB2 Express-C 9.5 εκτελείται σε περιβάλλον Windows και Linux χωρίς κανένα περιορισμό σε υλικό.

Η DB2 μπορεί να εκτελεστεί είτε από γραμμή εντολών είτε από μία GUI εφαρμογή. Η διεπαφή γραμμών εντολών απαιτεί καλή γνώση του προϊόντος, ενώ μέσω του γραφικού περιβάλλοντος οι χρήση της είναι απλή ακόμα και για αρχάριους. Υποστηρίζει και SQL και Xquery και διαθέτει μία εφαρμογή αποθήκευσης στοιχείων XML.

Η DB2 διαθέτει εφαρμογές για .NET, CLI, Java, Python, Perl, PHP, C++, C, COBOL, FORTRAN, και πολλές άλλες γλώσσες προγραμματισμού. Επίσης υποστηρίζει την ενσωμάτωση σε Eclipse και Visual Studio .NET.

Γιατί να επιλέξουμε την DB2 Express-C 9.5;

Η DB2 Express-C™ είναι η ελεύθερη έκδοση ενός από τα πιο προηγμένα συστήματα διαχείρισης βάσεων δεδομένων ανά τον κόσμο. Γιατί να πληρώνουμε όταν μπορούμε να έχουμε αυτό που χρειαζόμαστε δωρεάν; Η DB2 Express-C μας επιτρέπει να τη χρησιμοποιήσουμε χωρίς κανένα κόστος. Είναι ένα γρήγορο, ασφαλές και αξιόπιστο dataserver, ιδανικό για μικρομεσαίες επιχειρήσεις. Η DB2 Express-C 9.5 είναι διαθέσιμη σε Linux, Unix και Windows.

Επιτρέπει επίσης τον εύκολο χειρισμό της XML μέσω της εγγενούς τεχνολογίας αποθήκευσης που ονομάζεται pureXML™.

Μία καλή τεχνική χρήσης XML σε μια βάση δεδομένων είναι απλώς να τοποθετήσουμε το XML έγγραφο μέσα σε μια στήλη CLOB στη βάση. Αυτό ουσιαστικά μεταχειρίζεται την XML ως ένα γιγαντιαίο σύνολο στοιχείων.

Έπειτα μπορούμε να το διαβάσουμε από τη βάση δεδομένων, να το περάσουμε από έναν parser για την αναδημιουργία του XML εγγράφου. Αυτό είναι συνήθως πολύ ευκολότερο να το κάνουμε, αλλά δεν είναι τόσο ισχυρό. Δεν υπάρχει κανένας αποτελεσματικός τρόπος να αντλήσουμε πληροφορίες από τα στοιχεία του εγγράφου XML.

Οι βάσεις δεδομένων XML έχουν κερδίσει δημοτικότητα κατά τη διάρκεια των ετών επειδή εστιάζουν σε αυτό ακριβώς το πρόβλημα. Μας αφήνουν να αποθηκεύουμε τα στοιχεία μας γνήσια ως στοιχεία της XML. Μας αφήνουν να τα εξετάσουμε αποτελεσματικά χρησιμοποιώντας την XQuery, με αποτέλεσμα να μην χάνεται η δομή του εγγράφου.

Παρακάτω ακολουθούν κάποια παραδείγματα βημάτων για να γίνει όσο μπορεί πιο κατανοητός ο συνδυασμός BD2 και XML.

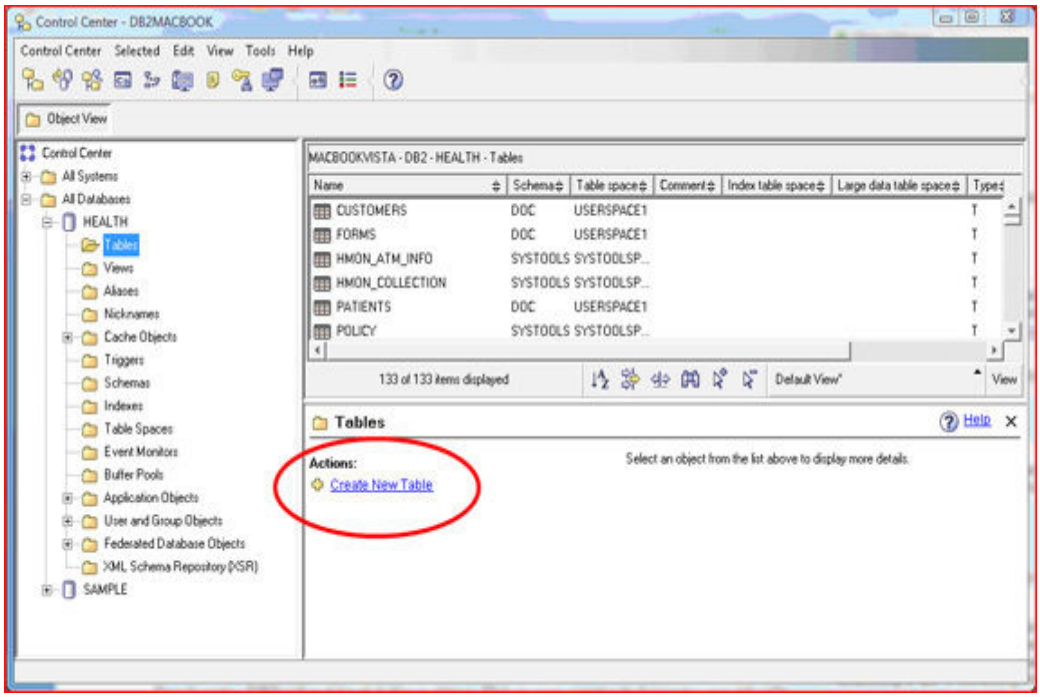
Ας υποθέσουμε ότι θα θέλαμε να χρησιμοποιήσουμε το επώνυμο ενός ασθενή για να ανακτήσουμε πληροφορίες. Μπορούμε να αποθηκεύουμε αυτές τις πληροφορίες μέσα στο έγγραφο XML και να γράψουμε ένα Xquery μόνον για να ανακτήσουμε έγγραφα βασισμένα στο επώνυμο του ασθενή.

Θα δημιουργήσουμε έναν πίνακα που θα χρησιμοποιεί εγγενή XML. Ένα πλεονέκτημα που έχει η DB2 είναι ότι επιτρέπει τη μίξη των συγγενικών στοιχείων και των στοιχείων XML. Αυτό που θα κάνουμε είναι ότι αντί να χαρτογραφήσουμε ένα έγγραφο XML άμεσα σε έναν πίνακα, το χαρτογραφούμε σε μια στήλη. Μια σειρά στον πίνακά μας θα μπορούσε επίσης να έχει πολλαπλά XML έγγραφα. Το παράδειγμα 1 παρουσιάζει ένα κομμάτι κώδικα που δημιουργεί έναν πίνακα που έχει μια μόνο στήλη XML.

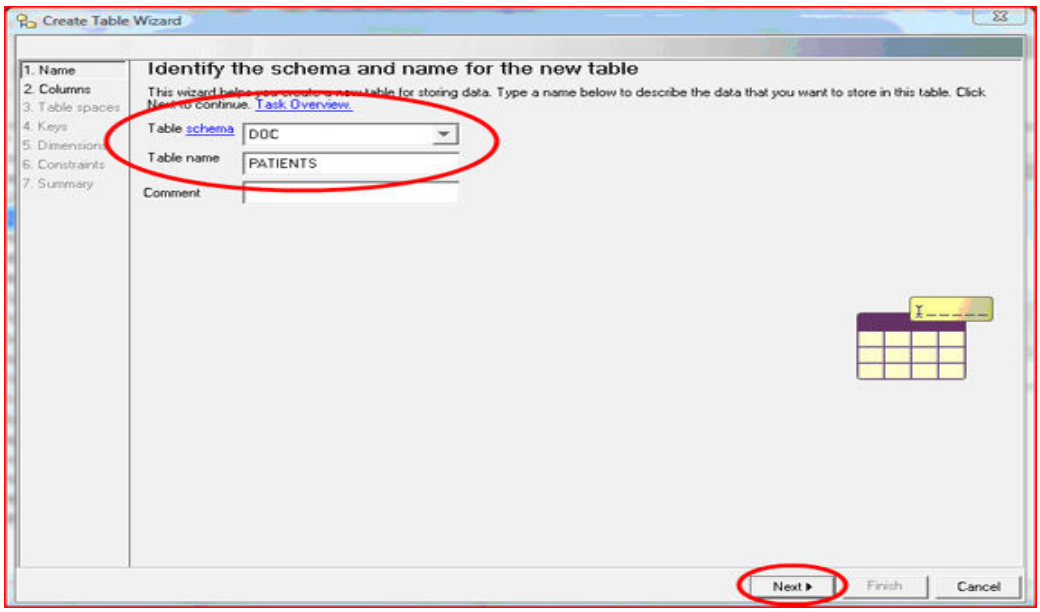
```
CREATE TABLE DOC.PATIENTS (
  ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (
        START with +1 INCREMENT BY +1),
INFORMATION XML NOT NULL);
```

Στο παράδειγμα, η στήλη ΠΛΗΡΟΦΟΡΙΕΣ είναι τύπου XML. Όπως μπορείτε να δούμε η σύνταξη είναι πολύ απλή.

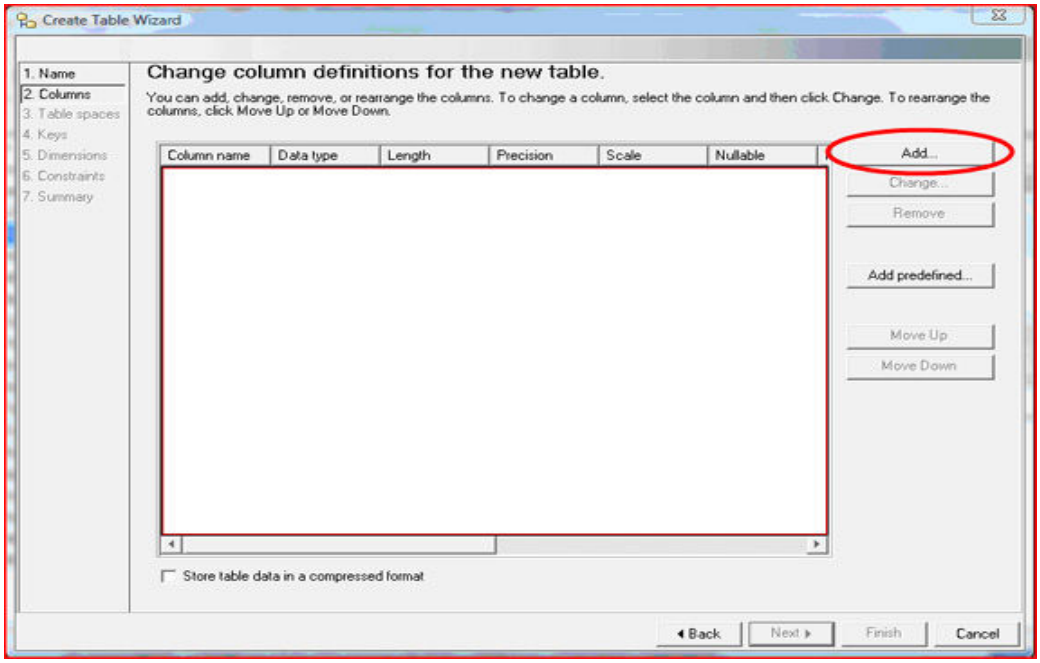
Εάν δεν θέλουμε να χρησιμοποιήσουμε τη γραμμή εντολών για τη δημιουργία των πινάκων, το Κέντρο Εντολών της DB2 μας προσφέρει μια εναλλακτική γραφική λύση που κάνει το ίδιο πράγμα, όπως φαίνεται στο σχήμα.



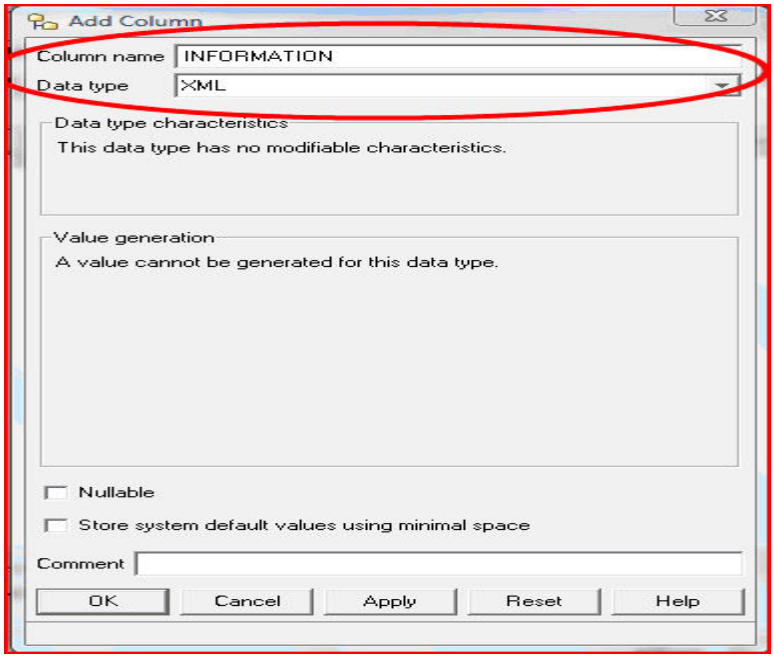
Κάνοντας κλικ πάνω στη Δημιουργία νέου πίνακα (Create new table) γίνεται εκκίνηση του wizard. Αυτό θα εμφανίσει την οθόνη πληροφοριών με τους πίνακες.



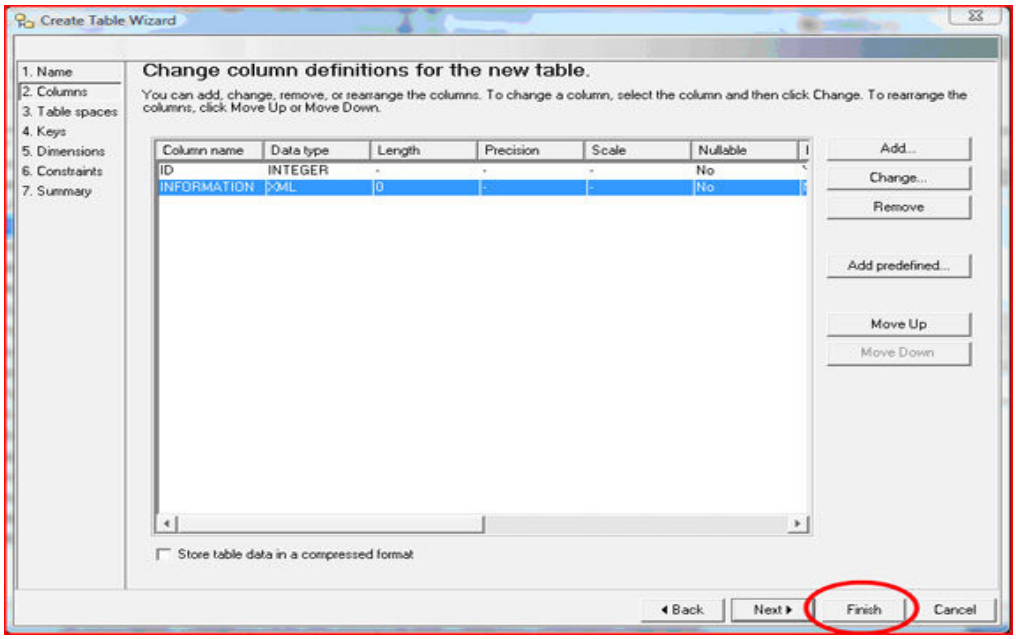
Ο πίνακας ονομάτων μας επιτρέπει να προσδιορίσουμε το σχήμα του πίνακά μας καθώς επίσης και την ονομασία του. Κάνοντας κλικ στο *Next* θα εμφανιστεί το παράθυρο καθορισμού στηλών.



Κάνοντας κλικ στο πλήκτρο *add* θα εμφανιστεί το παράθυρο στηλών.



Εδώ είναι το δύσκολο μέρος. Για να καθορίσουμε μια στήλη ως φέρουσα των στοιχείων XML , πρέπει να θέσουμε τον τύπο στοιχείων της στήλης σε XML. Μόλις τελειώσουμε, κάνουμε κλικ στο OK και θα εμφανιστεί η στήλη XML που καθορίσαμε.



Μπορούμε να προσθέσουμε περισσότερες στήλες, αρχικά και ξένα κλειδιά, και ακόμα περισσότερα στον wizard. Όταν τελειώσουμε, κάνουμε κλικ στο Finish και ο πίνακας έχει δημιουργηθεί. Με όποιον τρόπο και να δημιουργήσουνε τον πίνακα, μπορούμε να εισάγουμε τα στοιχεία στον πίνακα χρησιμοποιώντας SQL.

Μπορούμε να μεταχειριστούμε την XML ως ένα σύνολο κατά το γράψιμο του SQL όπως φαίνεται στο παράδειγμα.

INSERT INTO DOC.PATIENTS(INFORMATION) VALUES (‘<?xml version=”1.0” encoding=”UTF-8”?>

<Info>

<FirstName>Vasilis</FirstName>

<Age>26</Age>

<Insurer>IKA</Insurer>

<ID>234987547</ID>

</Info>’);

Μπορούμε επίσης να αντλήσουμε πληροφορίες από την XML κάνοντας χρήση της SQL και XQuery όπως φαίνεται στο παράδειγμα:

```
SELECT XMLQUERY('<Patients> {for $i in $x/Info
  where $i/Insurer = "IKA"
  return <Patient>{$i/FirstName/text()}</Patient>} </Patients>'
passing P.INFORMATION as "x")
  from DOC.PATIENTS P;
```

Η ανάμιξη SQL και XQuery μαζί είναι γνωστή ως SQL/XML. Η δήλωση της παραπάνω ερώτησης επιστρέφει το έγγραφο XML όπως παρουσιάζεται παρακάτω:

```
<?xml version="1.0" encoding="UTF-16"?>
<Patients>
    <Patient>Vasilis</Patient>
</Patients>
```

Είδαμε πόσο εύκολο είναι να χρησιμοποιήσουμε την DB2 για να αποθηκεύσουμε την XML μας εγγενώς. Έχοντας τουλάχιστον τις βασικές γνώσεις πάνω σε βάσεις δεδομένων και στη χρήση SQL εντολών/ ερωτημάτων, ο αναγνώστης του συγγράμματος είναι σε θέση να κάνει την πρώτη του επαφή με την DB2 σε συνδυασμό με Σχεσιακές Βάσεις Δεδομένων.

Ακολουθούν κάποια παραδείγματα SQL και XQuery, με κάποιες μικρές επεξηγήσεις για μια πιο καλύτερη κατανόηση.

1. Το πρώτο βήμα είναι ο ορισμός πίνακα.

```
create table clients (
        id int primary key not null,
      name varchar(50),
    status varchar(10),
  contactinfo xml );
```

2. Εισαγωγή μίας εγγραφής στον πίνακα Clients. Με τον ίδιο τρόπο μπορούμε να εισάγουμε και άλλες και δεν είναι υποχρεωτικό κάθε εγγραφή να έχει τα ίδια στοιχεία και το ίδιο μέγεθος, μπορούν να λείπουν κάποιες πληροφορίες.

```
insert into clients values (3227, 'Vasilis Nikas', 'Gold',
'<Client>
    <Address>
        <street>Konstantinoupoleos 8</street>
        <city>Thessaloniki</city>
        <state>GR</state>
        <zip>54406</zip>
    </Address>
    <phone>
        <work>2310358992</work>
        <home>---</home>
        <cell>69xxxxxx</cell>
    </phone>
    <fax>2310887556</fax>
    <email>test@yahoo.com</email>
</Client>'
  );
```

3. Ένα απλό XQuery ανάκτησης ενός ολόκληρου εγγράφου:

db2-fn:xmlcolumn('CLIENTS.CONTACTINFO')

Θα μας δώσει σαν αποτέλεσμα το XML έγγραφο με τις πληροφορίες όλων των εγγραφών μέσα στον πίνακα. Χρησιμοποιώντας SQL το ερώτημα θα ήταν κάπως έτσι:

select contactinfo from client

4. Ανάκτηση συγκεκριμένων πληροφοριών:

```
for $y in db2-fn:xmlcolumn('CLIENTS.CONTACTINFO')/Client/Address
    return $y
```

Το παραπάνω query θα έχει σαν αποτέλεσμα επιστροφής όλες τις διευθύνσεις όλων των εγγραφών.

Φτάνοντας στο τέλος του κεφαλαίου, θα αντιπαραβάλουμε την SQL/XML με την XQuery παραθέτοντας τα πλεονεκτήματα και τα μειονεκτήματα της μιας έναντι της άλλης.

SQL/XML πλεονεκτήματα

Η SQL/XML έχει τα ακόλουθα πλεονεκτήματα:

- Η SQL/XML είναι καλή εάν έχουμε μια υπάρχουσα εφαρμογή SQL και πρέπει να προσθέσουμε μια λειτουργία XML.
- Είναι καλή για όσους γνωρίζουν SQL και εάν θέλουμε να κρατήσουμε την SQL ως πρωταρχική γλώσσα.

- Είναι καλή εάν τα ερωτήματα μας πρέπει να επιστρέψουν στοιχεία από σχεσιακές και στήλες XML συγχρόνως.
- Είναι καλή εάν θέλουμε τα επιστρεφόμενα αποτελέσματα ως σύνολα στοιχείων XML και οι πληροφορίες που απουσιάζουν αναπαρίστανται με ΚΕΝΟ (NULL).
- Είναι καλή για εφαρμογές που πρέπει να ενσωματώσουν σχεσιακά στοιχεία και στοιχεία XML. Η ένωση αυτή γίνεται με πολύ εύκολο τρόπο.

```
select u.unitID from dept d, unit u
where XMLEXISTS('$d//employee[name = $m]'
        passing d.deptdoc as "d", u.manager as "m");
```

- Είναι καλή για την ομαδοποίηση στοιχείων XML.

SQL/XML μειονεκτήματα

Η SQL/XML έχει τα ακόλουθα μειονεκτήματα:

- Δεν είναι η πάντα η καλύτερη επιλογή για να μετασχηματιστεί ένα XML έγγραφο σε ένα άλλο έγγραφο XML.
- Μπορούμε να πούμε ότι η έκφραση μιας ένωσης μεταξύ δύο στηλών XML, ή γενικότερα μεταξύ δύο τιμών XML, μπορεί να είναι πιο εύστοχή και σαφής σε XQuery από ότι σε SQL/XML. Παραδείγματος χάριν, το παρακάτω ερώτημα ενώνει τις στήλες XML των πινάκων «Τμήμα» και «Έργο» για να επιστρέψει τους υπαλλήλους που εργάζονται σε οποιοδήποτε έργο.

SQL/XML

```
select XMLQUERY('$d/dept/employee' passing d.deptdoc as "d")
from dept d, project p
where XMLEXISTS('$d/dept[@deptID=$p/project/deptID]'
        passing d.deptdoc as "d", p.projectDoc as "p");
```

XQuery

```
for $dept in db2-fn:xmlcolumn("DEPT.DEPTDOC")/dept
  for $proj in db2-fn:xmlcolumn("PROJECT.PROJECTDOC")/project
      where $dept/@deptID = $proj/deptID
          return $dept/employee;
```

10. IBM DB2 και XML Schema

Η DB2 είναι ένα υβριδικό σύστημα βάσεων δεδομένων. Μπορεί να περιλαμβάνει σχεσιακά δεδομένα, αλλά και δεδομένα τύπου XML στην ίδια βάση. Τα XML δεδομένα αποθηκεύονται σε ιεραρχική μορφή στις στήλες ενός πίνακα, και μία υβριδική μηχανή επεξεργάζεται και τους δύο τύπους δεδομένων. Ένα XML έγγραφο πρέπει να είναι σωστά δομημένο (well-formed) για να εισαχθεί σε μία XML στήλη και με μέγεθος έως 2GB. Τα αντικείμενα XML δεδομένων αποθηκεύονται ξεχωριστά από τα αντικείμενα του πίνακα. Για κάθε γραμμή της στήλης

τύπου XML, υπάρχει ένας δείκτης XML δεδομένων (xds), ο οποίος αποθηκεύεται στον πίνακα. Ο δείκτης xds περιέχει τις πληροφορίες για την πρόσβαση των XML δεδομένων που είναι αποθηκευμένα στο δίσκο και χρησιμοποιείται επίσης για τις ενέργειες Import – Export.

Όπως έχει προαναφερθεί, τα XML σχήματα, όπως και τα DTDs, χρησιμοποιούνται για την τεκμηρίωση XML εγγράφων. Η DB2 9 υποστηρίζει μόνο XML σχήματα για τεκμηρίωση εγγράφων. Τα XML σχήματα, που χρησιμοποιούνται από την DB2 για τεκμηρίωση θα πρέπει να βρίσκονται καταχωρημένα σε μία αποθήκη XML σχημάτων της DB2, τη λεγόμενη XSR (**X**ml **S**chema **R**epository).

Τα XML έγγραφα που αποθηκεύονται σε στήλες τύπου XML, συνήθως περιέχουν κάποια URI (Uniform Resource Identifier) ή αναφορές που περιέχουν πληροφορίες για σχήματα, κάποιο DTD ή οποιαδήποτε άλλη εξωτερική οντότητα.

Η XSR διαχειρίζεται αυτές τις εξαρτήσεις σε εξωτερικές οντότητες χωρίς κάποια αλλαγή στην αναφορά του URI. Η αποθήκη αυτή βρίσκεται μέσα στον κατάλογο της βάσης και περιλαμβάνει τους καταλόγους των πινάκων, τους καταλόγους απεικονίσεων και κάποιες διαδικασίες συστήματος για την εισαγωγή στους καταλόγους των πινάκων αυτών.

Ένα XML σχήμα μπορεί να περιέχει ένα ή περισσότερα XML σχήματα. Ας υποθέσουμε ότι ένα σχήμα κάνει εισαγωγή δύο άλλων σχημάτων, εκ των οποίων το ένα από αυτά εισάγει με τη σειρά του

κάποιο άλλο. Όλα τα σχήματα, DTD και οποιαδήποτε εξωτερική οντότητα θα πρέπει να καταχωρηθούν στην αποθήκη XSR πριν τα χρησιμοποιήσουμε. Μία καταχώρηση ενός XSR αντικειμένου θεωρείται επιτυχής, όταν καταχωρηθούν όλα τα επιμέρους σχήματα που το απαρτίζουν.

Για να γίνει πιο κατανοητή η καταχώρηση σχημάτων και η διαχείριση XSR αντικειμένων, παρακάτω δίνεται ένα παράδειγμα.

Ακολουθεί το σχήμα pets.xsd το οποίο είναι πρωταρχικό σχήμα:

Pets.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.itso.org/pets"
xmlns:pe="http://www.itso.org/pets" xmlns:ca="http://www.itso.org/cat"
xmlns:do="http://www.itso.org/dog">
<xs:import namespace="http://www.itso.org/cat" schemaLocation="cat.xsd"
/>
<xs:import namespace="http://www.itso.org/dog" schemaLocation="dog.xsd"
/>
  <xs:element name="PETS">
        <xs:complexType>
              <xs:sequence>
                    <xs:element name="DOG" type="do:DOG"/>
                    <xs:element name="CAT" type="ca:CAT"/>
              </xs:sequence>
        </xs:complexType>
  </xs:element>
</xs:schema>
```

Όπως φαίνεται και στον κώδικα, περιέχει δύο στοιχεία CAT και DOG (μπλε χαρακτήρες) όπου τα σχήματα που τα περιγράφουν γίνονται εισαγωγή πιο πριν (κόκκινοι χαρακτήρες). Τα σχήματα που χρησιμοποιούνται για την περιγραφή των δύο αυτών στοιχείων είναι:

Cat.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.itso.org/cat">
  <xsd:complexType name="CAT">
        <xsd:sequence>
              <xsd:element name="NAME" type="xsd:string" />
              <xsd:element name="AGE" type="xsd:integer" />
        </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Dog.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.itso.org/dog">
  <xsd:complexType name="DOG">
        <xsd:sequence>
              <xsd:element name="NAME" type="xsd:string" />
              <xsd:element name="AGE" type="xsd:integer" />
        </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Το σχήμα pets.xsd δεν μπορεί από μόνο του να περιγράψει ένα έγγραφο που να περιέχει στοιχεία τύπου cat και dog. Για την τεκμηρίωση ενός τέτοιου εγγράφου με αυτό το σχήμα θα πρέπει να

καταχωρηθούν στην XSR τα σχήματα Cat.xsd και Dog.xsd τα οποία εισάγει το πρωταρχικό μας σχήμα.

Τα βήματα για την καταχώρηση ενός σχήματος στην αποθήκη XSR είναι:

- *Καταχώρηση του πρωταρχικού σχήματος.* Η πληροφορίες που χρειάζονται για την καταχώρηση αυτή είναι το ακριβές όνομα του σχήματος και το πλήρες μονοπάτι διαδρομής όπου βρίσκεται αποθηκευμένο, όπως επίσης και έναν προσδιοριστή SQL. Το πλήρες μονοπάτι διαδρομής μπορεί να είναι ένα URL, μία διεύθυνση FTP ή ένα μονοπάτι στον τοπικό υπολογιστή. Μπορεί να δηλωθεί οποιαδήποτε λέξη ως προσδιοριστής, κάτι που να βοηθάει στην εύκολη κατανόηση σχετικά με ποιο σχήμα συνδέεται. Ο προσδιοριστής χρειάζεται κατά την καταχώρηση ενός σχήματος στην XSR, αλλά και κατά την τεκμηρίωση ενός εγγράφου με το συγκεκριμένο σχήμα. Ακολουθεί ένα παράδειγμα εντολής καταχώρησης ενός πρωταρχικού σχήματος.

register xmlschema http://sample from c:\pets.xsd as sample.pets

Στο παραπάνω παράδειγμα, το ***c:\pets.xsd*** είναι το όνομα με το πλήρες μονοπάτι διαδρομής και το ***sample.pets*** είναι ο SQL προσδιοριστής.

- Προσθήκη επιπρόσθετων σχημάτων. Πρέπει να προστεθούν όλα τα επιπρόσθετα σχήματα που εισάγει το πρωταρχικό σχήμα. Ένα παράδειγμα εντολών προσθήκης των cat.xsd και dog.xsd είναι:

add xmlschema document to sample.pets add cat.xsd from c:\cat.xsd

add xmlschema document to sample.pets add dog.xsd from c:\dog.xsd

Στο παραπάνω παράδειγμα τα ***cat.xsd*** και ***dog.xsd*** είναι τα σχήματα, το ***sample.pets*** είναι ο προσδιοριστής του πρωταρχικού σχήματος για να γίνει η σύνδεση με τα επιπρόσθετα σχήματα και τα ***c:\cat.xsd*** και ***c:\dog.xsd*** οι διαδρομές για το που βρίσκονται τα σχήματα. Σειρά για το ποιο επιπρόσθετο σχήμα θα εισαχθεί πρώτο δεν υπάρχει, μπορούσαν οι εντολές προσθήκης να ήταν και αντίστροφα.

- Ολοκλήρωση καταχώρησης στην XSR. Αφού καταχωρηθεί το πρωταρχικό σχήμα και προστεθούν όλα τα επιπρόσθετα σχήματα μπορεί να ολοκληρωθεί η καταχώρηση. Το βήμα αυτό ελέγχει όλες τις αναφορές για εισαγωγές άλλων σχημάτων να βρει εάν εισήχθησαν τα σχήματα αυτά. Αν κάποια αντιστοίχηση δεν είναι σωστή, τότε δεν ολοκληρώνεται η καταχώρηση. Επίσης θα πρέπει όλα τα έγγραφα σχημάτων να είναι σωστά δομημένα, εάν κάποιο δεν είναι τότε πάλι αποτυγχάνει η καταχώρηση. Η εντολή ολοκλήρωσης του προσδιοριστή ***sample.pets*** είναι:

complete xmlschema sample.pets

Μόλις ένα σχήμα καταχωρηθεί στην XSR, τότε είναι ένα αντικείμενο της. Μπορεί να καταργηθεί ένα σχήμα από την XSR, αλλά πρέπει να είναι γνωστό ότι μαζί με το πρωταρχικό σχήμα καταργούνται και τα επιπρόσθετα σχήματα που εισάγει το συγκεκριμένο σχήμα. Επίσης, εάν κάποιο σχήμα της XSR, είτε πρωταρχικό είτε επιπρόσθετο, πρέπει να τροποποιηθεί, τότε θα πρέπει να καταργηθεί όλο το

αντικείμενο της αποθήκης και να αναδημιουργηθεί ξανά από την αρχή με όλα τα σχήματα που το απαρτίζουν. Η εντολή κατάργησης ενός σχήματος είναι:

drop xsrobject sample.pets

10.1 Κάνοντας ερωτήματα στην XSR

Μπορούν να γίνουν SQL ερωτήματα στην XSR και να αντληθούν πληροφορίες σχετικά με τα αντικείμενα της αποθήκης. Για τον περιγραφικό πίνακα της αποθήκης γράφουμε SYSCAT.XSROBJECTS. Ένα παράδειγμα αποτελεσμάτων αυτής της εντολής είναι:

Column name	**Type schema**	**Type name**	**Length**	**Scale**	**Nulls**
OBJECTID	SYSIBM	BIGINT	8	0	No
OBJECTSCHEMA	SYSIBM	VARCHAR	128	0	No
OBJECTNAME	SYSIBM	VARCHAR	128	0	No
TARGETNAMESPACE	SYSIBM	VARCHAR	1001	0	Yes
SCHEMALOCATION	SYSIBM	VARCHAR	1001	0	Yes
OBJECTINFO	SYSIBM	XML	0	0	Yes
OBJECTINFO	SYSIBM	XML	0	0	Yes
OWNER	SYSIBM	VARCHAR	128	0	No
CREATE_TIME	SYSIBM	TIMESTAMP	10	0	No
ALTER_TIME	SYSIBM	TIMESTAMP	10	0	No
STATUS	SYSIBM	CHARACTER	1	0	No
DECOMPOSITION	SYSIBM	CHARACTER	1	0	No
REMARKS	SYSIBM	VARCHAR	254	0	Yes

Επίσης, μπορούν να αντληθούν πληροφορίες για συγκεκριμένα στοιχεία του παραπάνω περιγραφικού πίνακα. Όπως π.χ.

select SCHEMALOCATION, TARGETNAMESPACE, OBJECTSCHEMA, OBJECTNAME from SYSCAT.XSROBJECTS

Επίσης, μία άλλη ερώτηση που μπορεί να γίνει στον περιγραφικό πίνακα της αποθήκη είναι σχετικά με τα σχήματα, μπορούν να αντληθούν πληροφορίες σχετικά με τα σχήματα με την εντολή SYSCAT.XSROBJECTCOMPONENTS. Στη γενική εικόνα της παραπάνω εντολής μπορεί να γίνει ακόμη ένα ερώτημα για συγκεκριμένα στοιχεία του π.χ.

select TARGETNAMESPACE, SCHEMALOCATION, OBJECTSCHEMA, OBJECTNAME FROM SYSCAT.XSROBJECTCOMPONENTS

Ένα παράδειγμα αποτελέσματος είναι:

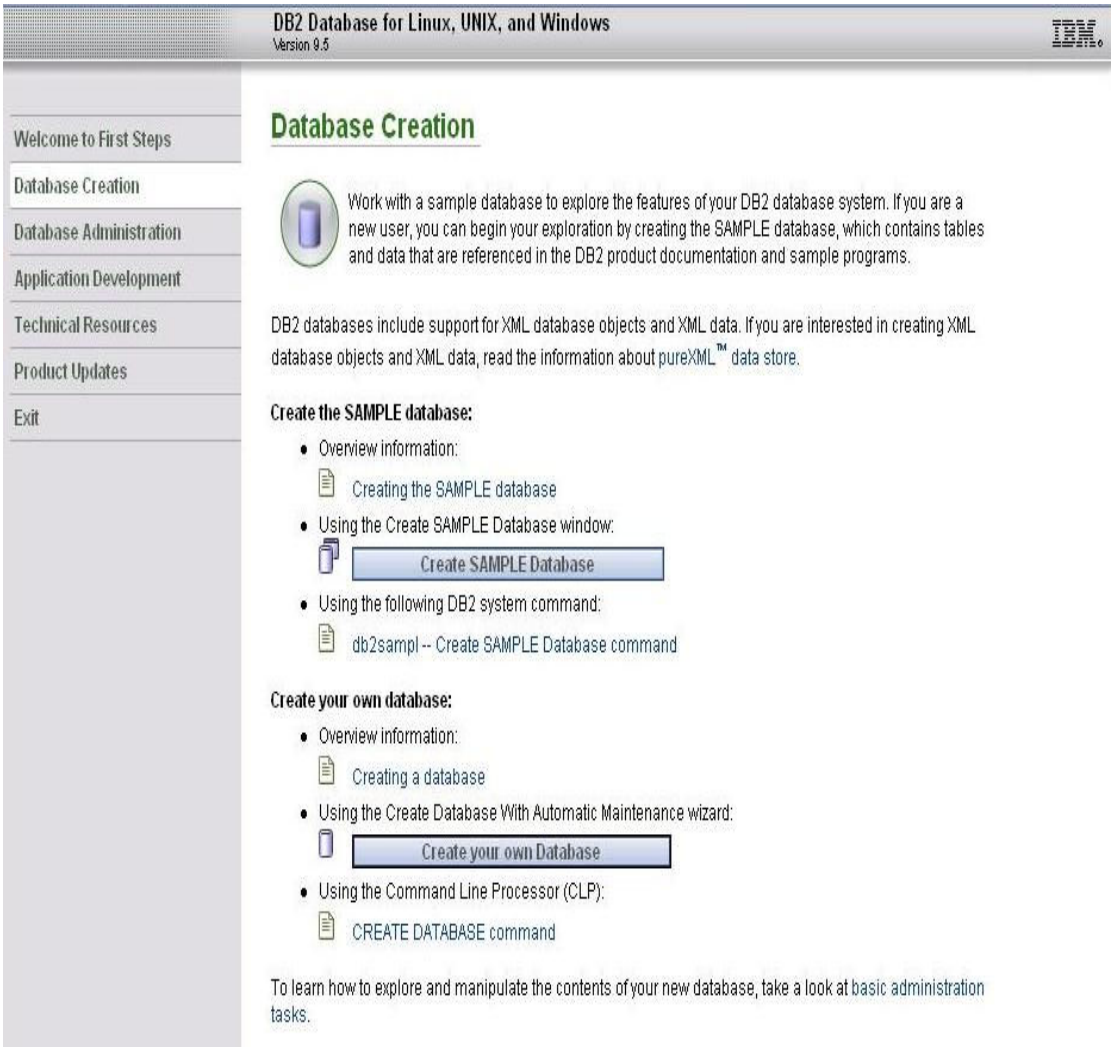
TARGETNAMESPACE	SCHEMALOCATION	OBJECTSCHEMA	OBJECTNAME
http://www.itso.org/sample	http://sample	SAMPLE	ORDER
http://person	http://person	JOHN	PERSON
http://www.itso.org/dog	dog.xsd	SAMPLE	PETS
http://www.itso.org/pets	http://sample	SAMPLE	PETS

10.2 Η IBM DB2 Υποστηρίζει XSR

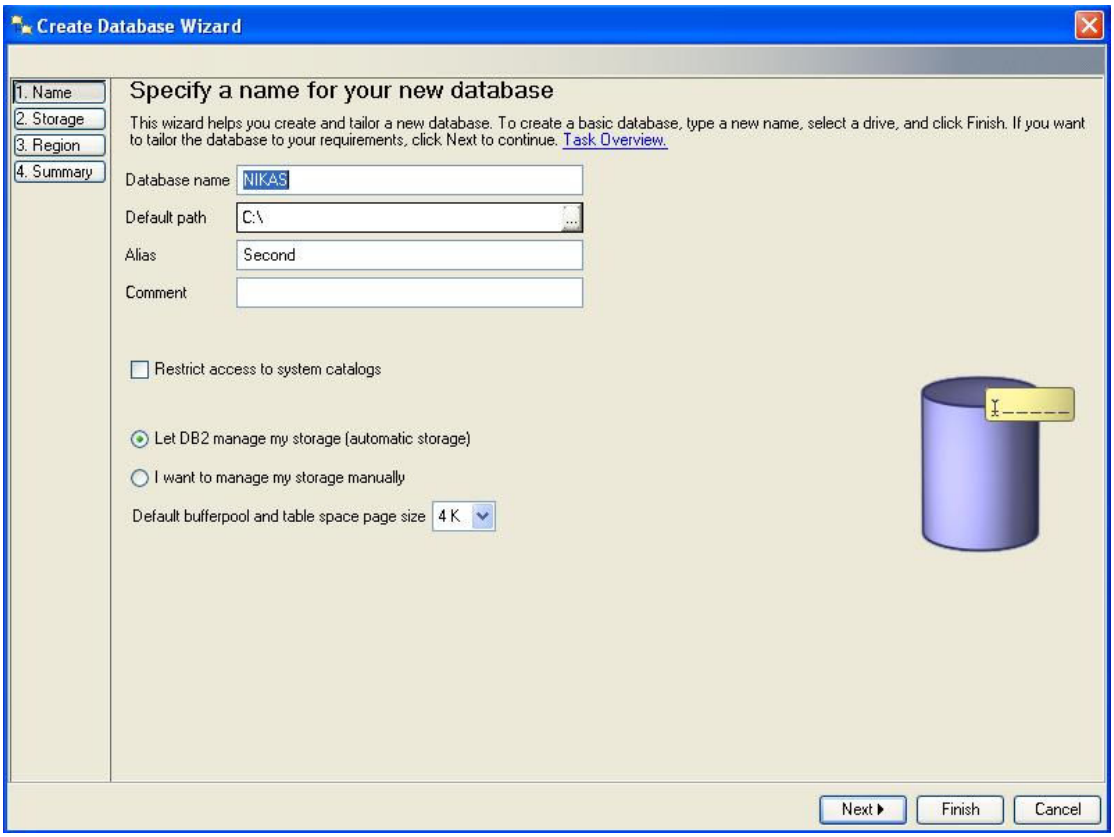
Ένας άλλος τρόπος διαχειρισμού της αποθήκης XSR και των αντικειμένων της, εκτός από εντολές SQL, είναι και το Κέντρο Ελέγχου της DB2., το οποίο υποστηρίζει καταχωρήσεις και καταργήσεις σχημάτων στην XSR.

Το κέντρο ελέγχου με αρκετούς βοηθούς (wizards) που διαθέτει είναι πολύ εύκολο στη χρήση του για τη δημιουργία μίας βάσης, καθώς

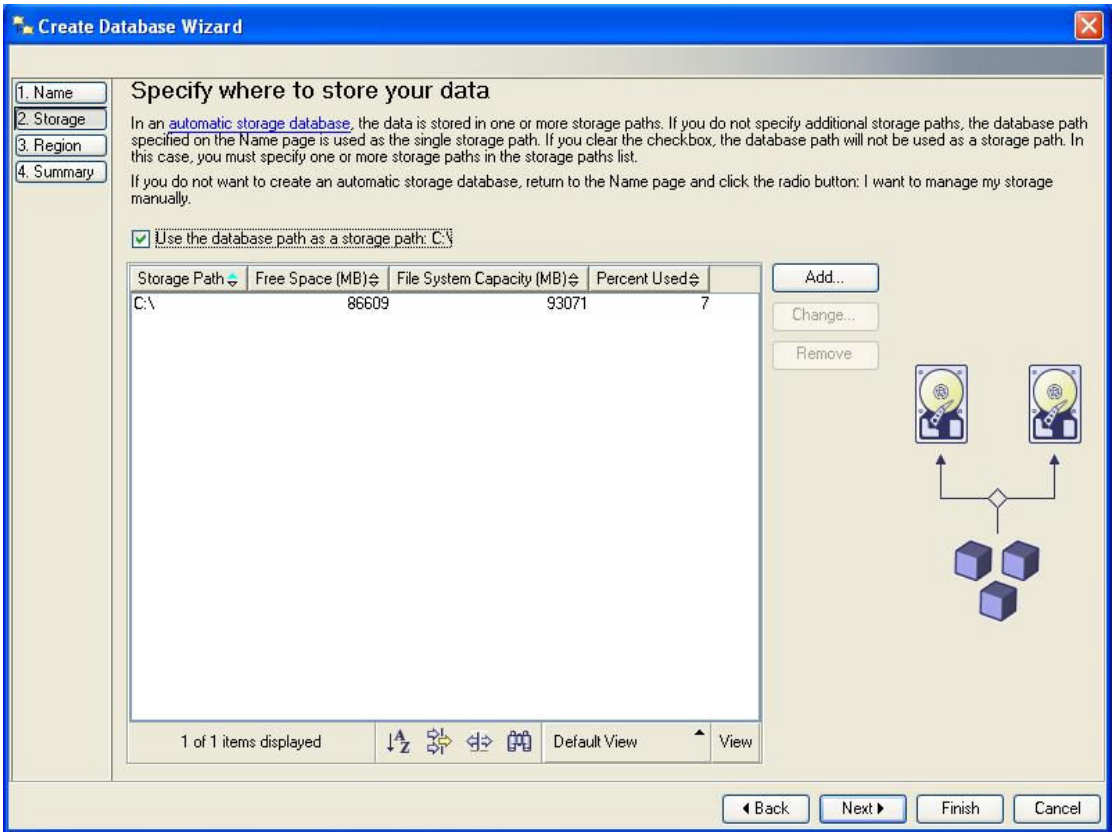
επίσης και στη διαχείριση των περιεχομένων της. Σαν πρώτο βήμα θα πρέπει να δημιουργηθεί η βάση με τον βοηθό. Παρακάτω ακολουθούν τα βήματα που πρέπει να γίνουν:

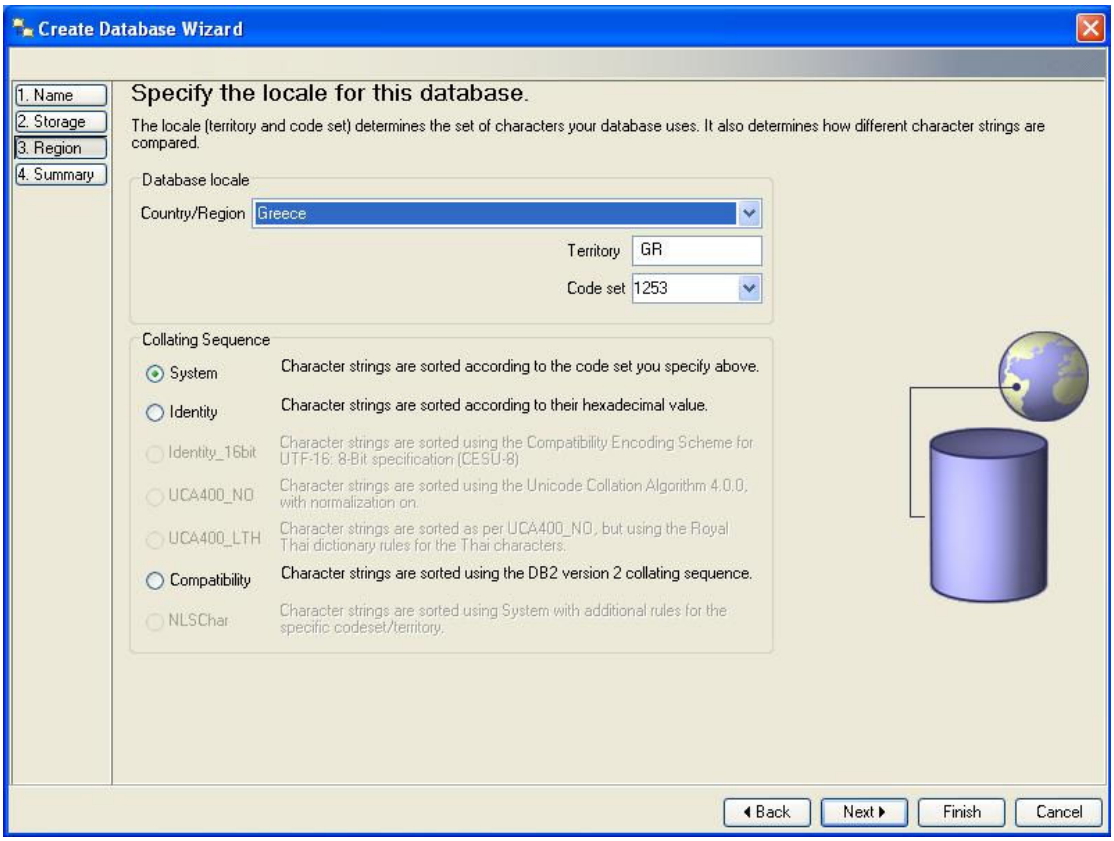


Ονομασία της βάση και ορισμός διαδρομής αποθήκευσης.

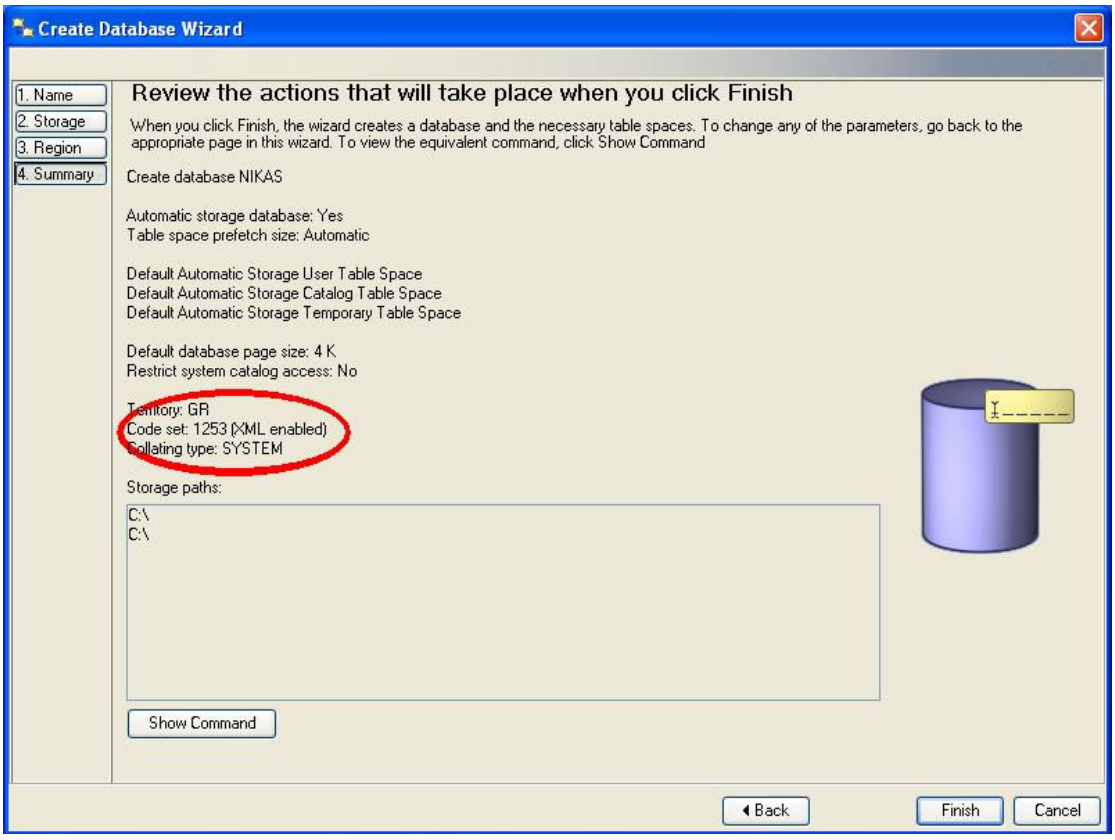


Ορισμός του μέγεθος της βάσης.

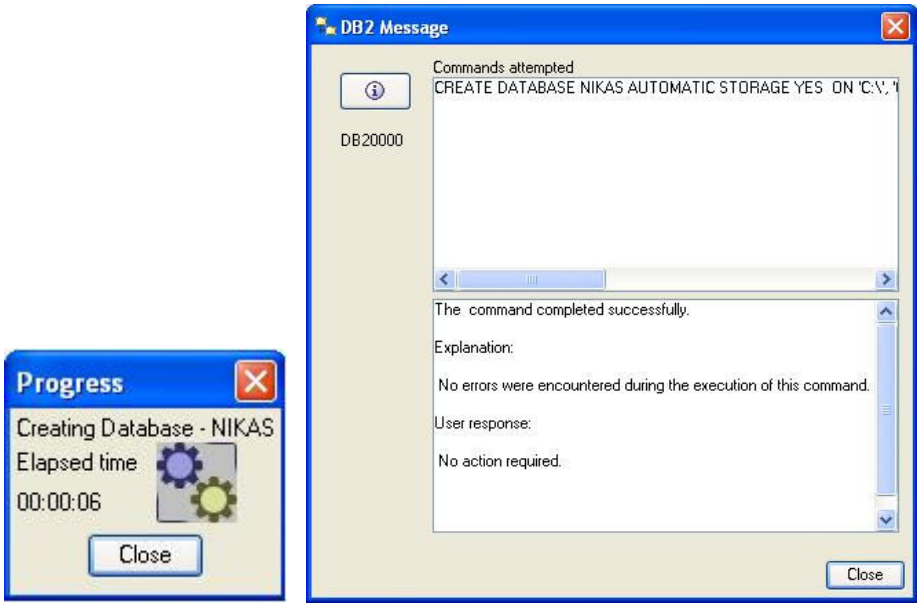




Παρατηρείται ότι επιτρέπεται η διαχείριση XML στη βάση.

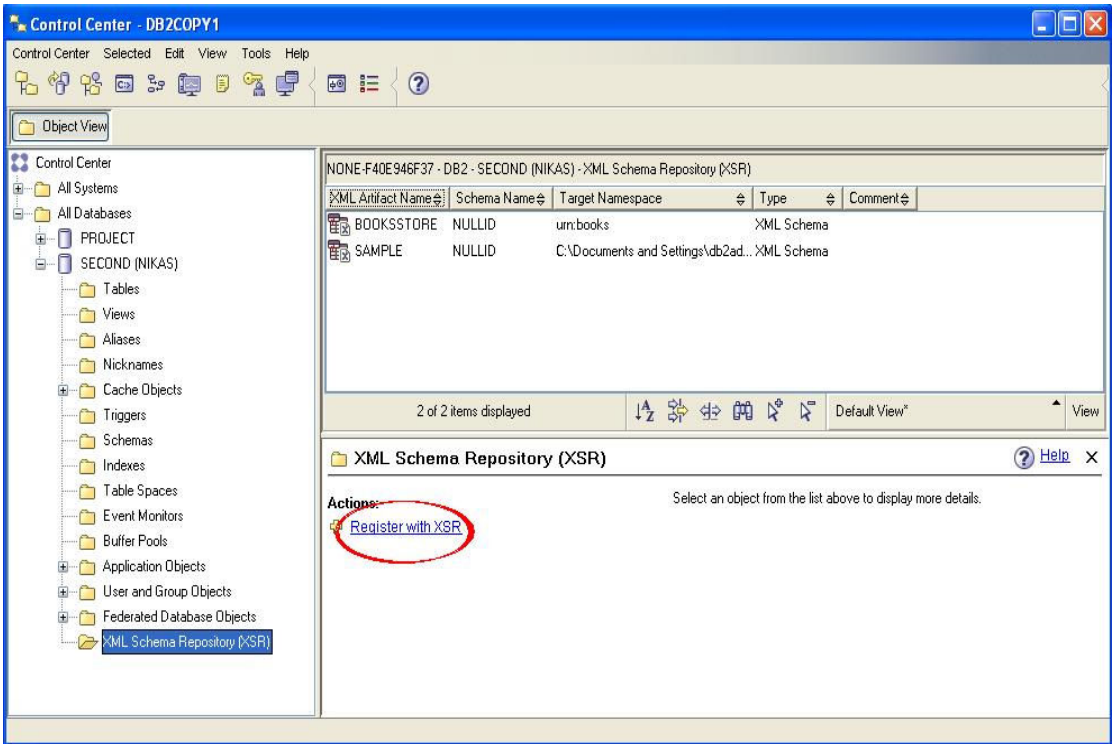


Μετά από λίγο έχει δημιουργηθεί η βάση.

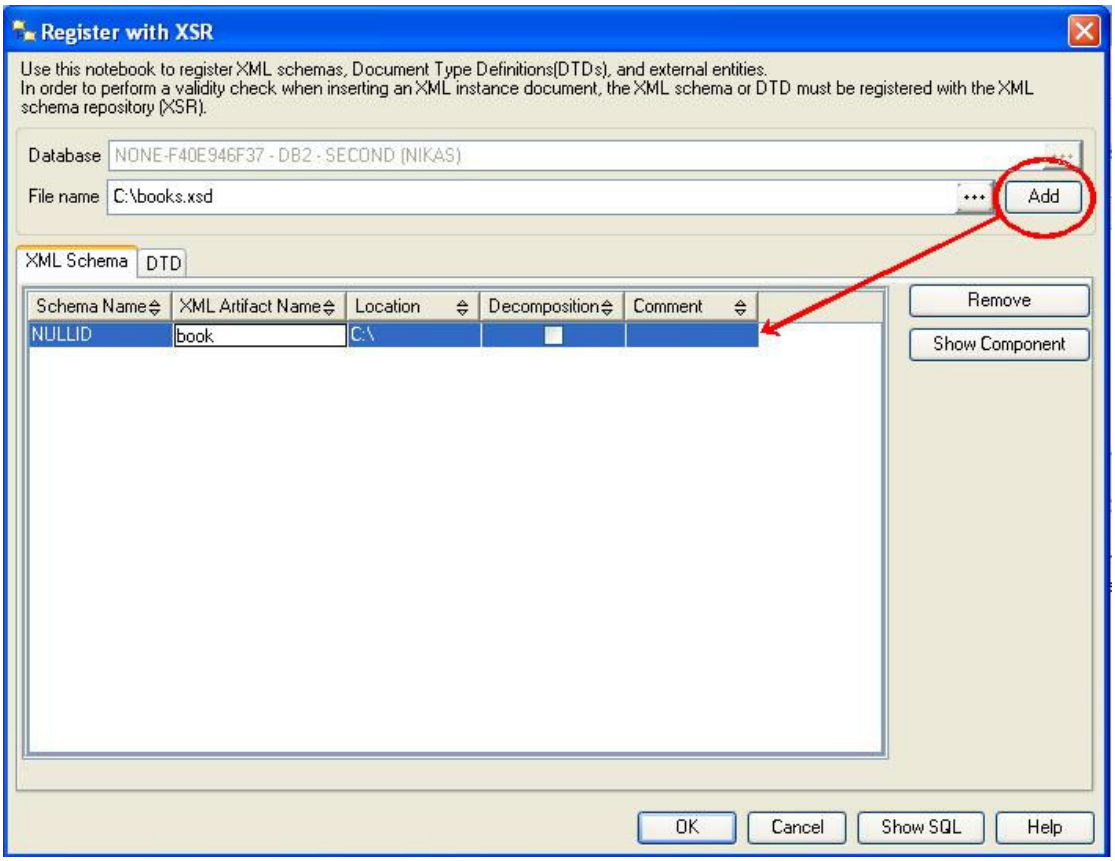


Το επόμενο βήμα είναι η καταχώρηση των σχημάτων που θα χρησιμοποιηθούν στην αποθήκη XSR της βάσης.

Κάνουμε κλικ στον σύνδεσμό Register with XSR.

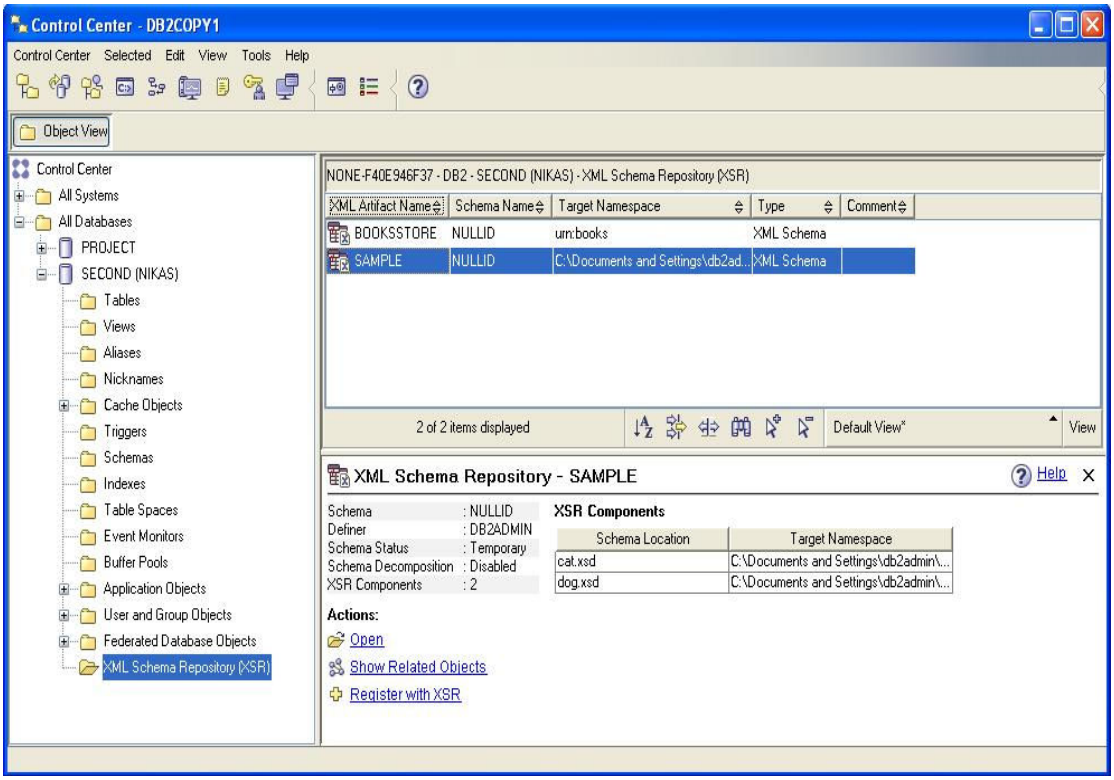


Όπως φαίνεται και από την παρακάτω εικόνα, ορίζεται η διαδρομή όπου βρίσκεται αποθηκευμένο το σχήμα και πατώντας το πλήκτρο Add είναι έτοιμο προς καταχώρηση στην XSR.



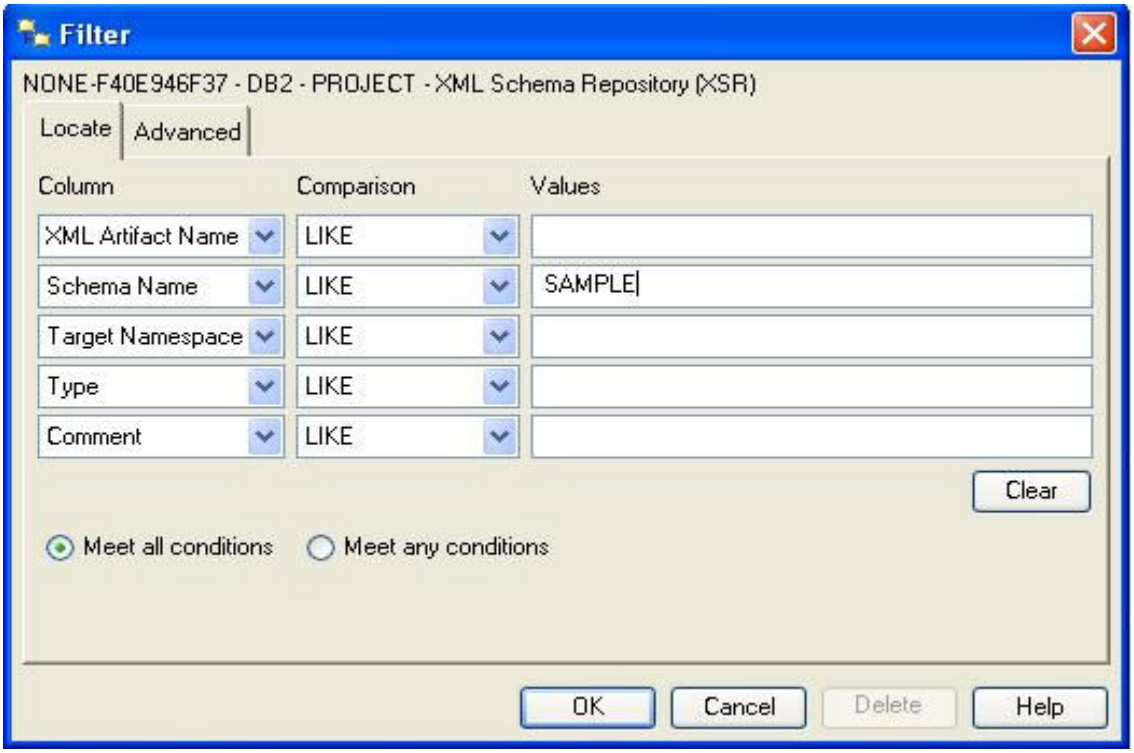
Πατώντας στη συνέχεια το πλήκτρο OK, περιμένουμε μέχρι να ολοκληρωθεί η καταχώρηση και εμφανίζεται το ακόλουθο μήνυμα.

Ένα παράδειγμα απεικόνισης του Κέντρου Ελέγχου με εμφανή τις καταχωρήσεις στην XSR είναι:



Επίσης, μπορεί να γίνει φιλτράρισμα των σχημάτων που θα εμφανίζονται κάθε φορά κάνοντας χρήση του φίλτρου που παρέχει το Κέντρο Ελέγχου.

Για το άνοιγμα του παραθύρου φίλτρων επιλέγουμε **Selected** → **Filter** → **Create** στο μενού του Κέντρου. Μπορεί να γίνει φιλτράρισμα των σχημάτων με βάση τα XML Artifact Name, Schema Name, Target Namespace, Type, and Comment.



10.3 Λογική Σχεδίαση Βάσης

Σε μία υβριδική βάση δεδομένων ο σχεδιασμός πινάκων περιλαμβάνει προβληματισμούς του τύπου τι είδους πληροφορίες θα έπρεπε να χρησιμοποιηθούν σε σχεσιακές στήλες πληροφοριών και τι πληροφορίες σε στήλες XML. Ένας άλλος προβληματισμός όσον αφορά στην αποθήκευση XML εγγράφων σε μία Βάση Δεδομένων είναι η τεκμηρίωση του εγγράφου XML. Σ' αυτό το κεφάλαιο θα επιχειρήσουμε να παραθέσουμε τους τύπους τεκμηρίωσης, καθώς και τα πλεονεκτήματα και μειονεκτήματα τους.

10.3.1 Τύπος Δεδομένων XML

Η αρχιτεκτονική αποθήκευσης και επεξεργασίας XML δεν θέτει όρια στο μέγεθος ενός εγγράφου XML. Όμως, το πρωτόκολλο

επικοινωνίας πελάτη-κεντρικού εξυπηρετητή (client – server) οριοθετεί το μέγεθος 2GB ανά έγγραφο.

Μπορούν να παρεμβληθούν μόνο σωστά δομημένα (well-formed) έγγραφα XML, τα οποία πρέπει να ικανοποιούν ορισμένους κανόνες σύνταξης που διευκρινίζονται στα πρότυπα για XML σύνταξη του οργανισμού W3C. Στις εφαρμογές, μπορούν να συνδυαστούν μία XML στήλη σε έναν δυαδικό τύπο, αλφαριθμητικό ή τύπο XML μεταβλητών. Μπορούν επίσης, να εισαχθούν-ενημερωθούν-διαγραφούν XML δεδομένα σε μία στήλη τύπου XML με SQL δηλώσεις διαχείρισης δεδομένων, όπως γίνεται και με δεδομένα σχεσιακού τύπου.

Ένα έγγραφο XML μπορεί να παρεμβληθεί σε μια στήλη τύπων LOB/LONG VARCHAR. Η επιλογή ενός ολόκληρου XML εγγράφου είναι γρήγορη επειδή οι πληροφορίες δεν αποθηκεύονται σε δενδρική μορφή, με αποτέλεσμα να μην υπάρχει λόγος για σειριακή ταξινόμηση (serialization). Γενικότερα, τα XML έγγραφα μπορούν να αποθηκευτούν ως τύποι LOB/LONG VARCHAR εάν ισχύει έστω και μία από τις ακόλουθες προτάσεις:

- ✓ Καμία διαδικασία δεν απαιτείται στο έγγραφο XML. Δεν είναι απαραίτητο να αναζητηθεί, να εξαχθεί, ή να ενημερωθεί μερικώς το έγγραφο. Παραδείγματος χάριν, το έγγραφο πρέπει να κρατηθεί άθικτο για επιχειρησιακούς κανόνες ή νομικούς λόγους.
- ✓ Το έγγραφο XML προέρχεται από μια έγκυρη πηγή που εγγυάται τη σωστή δομή του.

- ✓ Η καλή δομή δεν είναι σημαντική για το XML έγγραφο.
- ✓ Οι μόνες διαδικασίες στο XML έγγραφο είναι η εισαγωγή και η επιλογή ολόκληρου του εγγράφου.

10.3.2 Σχεσιακή Δομή vs XML Δομή

Όσον αφορά τον σχεδιασμό υβριδικής βάσης δεδομένων ένα ερώτημα που θα μπορούσε να προκύψει θα ήταν: Τι πληροφορίες θα έπρεπε να αποθηκευτούν σε σχεσιακή μορφή και τι σε μορφή XML;

Γενικά, τα στοιχεία που έχουν τις ακόλουθες ιδιότητες πρέπει να αποθηκευτούν σε μια σχεσιακή δομή:

- ✓ Όταν υποβάλλονται σε επεξεργασία με άλλα σχεσιακά στοιχεία.
- ✓ Όταν περιγράφονται καλύτερα με πινακοειδή μορφή.
- ✓ Όταν οι τιμές είναι ανεξάρτητες από τις ιεραρχικές δομές XML.
- ✓ Όταν πρέπει να προσπελαστεί από εφαρμογές που επεξεργάζονται σχεσιακά στοιχεία.

Εν αντιθέσει, τα στοιχεία που έχουν τις ακόλουθες ιδιότητες πρέπει να αποθηκευτούν ως στοιχεία XML:

- ✓ Όταν περιγράφονται καλύτερα σε ιεραρχική δομή. Η υψηλή πολυπλοκότητα των ιεραρχικών δομών απαιτεί μεγάλο αριθμό πινάκων και είναι δύσκολο να χαρτογραφηθούν σε σχεσιακή δομή. Ως XML είναι ο φυσικότερος τρόπος να αποθηκευτούν τέτοια στοιχεία.
- ✓ Πολλές ιδιότητες των στοιχείων είναι κενές ή άγνωστες. Εάν χαρτογραφηθούν τα στοιχεία σε σχεσιακούς πίνακες, θα υπάρξουν

πολλές μηδενικές τιμές στους πίνακες. Εάν το στοιχείο είναι περίπλοκο και μεγάλο, απαιτούνται πολλοί πίνακες. Το XML σχήμα είναι πιο ευέλικτο και η πρόσβαση με XQuery σε τέτοια στοιχεία είναι πολύ πιο απλή.

✓ Υπάρχει η περίπτωση να έχουμε μικρό όγκο δεδομένων με ιδιαίτερα σύνθετη δομή.

✓ Το σχήμα αλλάζει συνεχώς και εξελίσσεται. Οι επιχειρησιακοί κανόνες μπορούν να αλλάξουν και να έχουν επιπτώσεις στο σχήμα. Γενικά, είναι ευκολότερο να γίνει η εξέλιξη XML σχημάτων, παρά να αλλαχτεί το σχεσιακό σχήμα.

Μερικές φορές μία επιχείρηση μπορεί να αλλάξει κάποια στοιχεία στη δομή και γενικά στον τρόπο λειτουργίας της. Αυτοί οι κανόνες επηρεάζουν τα σχήματα που χρησιμοποιεί η επιχείρηση, με αποτέλεσμα τα σχήματα αυτά να πρέπει να τροποποιηθούν έτσι ώστε να ικανοποιούν τους νέους αυτούς κανόνες.

Παραδείγματος χάριν, η αποθήκη μιας επιχείρησης κάνει χρήση ενός σχήματος για την ικανοποίηση των παραγγελιών. Το στοιχείο προϊόν έχει όνομα, ποσότητα και τιμή ως στοιχεία μεταβλητών και τις ιδιότητες βάρος και χρώμα. Το σχήμα που ικανοποιεί το παραπάνω παράδειγμα είναι το order.xsd:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:complexType name="item">
      <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
```

```
                <xsd:element name="quantity" type="xsd:integer"/>
                <xsd:element name="price" type="xsd:integer"/>
        </xsd:sequence>
        <xsd:attribute name="weight" type="xsd:integer"/>
        <xsd:attribute name="color" type="xsd:string"/>
        </xsd:complexType>
    <xsd:element name="order" >
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="order" type="item" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Η επιχείρηση αποφάσισε να προσθέσει έναν κανόνα στο παραπάνω σχήμα. Προτείνει στους πελάτες να γίνουν εγγεγραμμένα μέλη και με κάθε αγορά τους ως μέλη να έχουν κάποια έκπτωση σε διάφορα προϊόντα. Η επιχείρηση αποφάσισε να γνωστοποιεί στους πελάτες με κάθε αγορά τους πόσο κερδίζουν από κάθε προϊόν. Αυτό έχει ως αποτέλεσμα την τροποποίηση του σχήματος order.xsd έτσι ώστε να ικανοποιεί τον νέο κανόνα. Όταν σε κάποια σχήματα γίνουν τέτοιες αλλαγές, η πράξη αυτή λέγεται εξέλιξη σχημάτων. Το νέο σχήμα είναι ίδιο με το προηγούμενο, με τη μόνη διαφορά την προσθήκη ενός στοιχείου μεταβλητής στο προϊόν με την ονομασία έκπτωση.

```
<xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="quantity" type="xsd:integer"/>
        <xsd:element name="price" type="xsd:integer"/>
        <xsd:element name="discount" type="xsd:integer" minOccurs="0"/>
</xsd:sequence>
```

Το στοιχείο αυτό περιέχει την ιδιότητα minOccurs=0 γιατί δεν έχουν όλα τα προϊόντα έκπτωση.

Το επόμενο βήμα μετά τη δημιουργία του νέου σχήματος είναι να καταργηθεί το παλαιό σχήμα από την XSR, να γίνει η καταχώρηση του νέου και να τεκμηριωθεί η εφαρμογή με το νέο αυτό σχήμα. Εάν δεν είναι επιθυμητό η τεκμηρίωση της εφαρμογής από την αρχή, μπορεί να καταχωρηθεί το νέο σχήμα στην XSR με τις ίδιες πανομοιότυπες πληροφορίες του παλαιού σχήματος.

Αποσύνθεση XML εγγράφων σε σχεσιακούς πίνακες

Τα έγγραφα XML μπορούν να αποσυντεθούν σε έναν σχεσιακό πίνακα, όπως επίσης τα αποσυντεθειμένα έγγραφα μπορούν να ανασυντεθούν ξανά σε έγγραφο XML. Όμως το νέο έγγραφο XML είναι διαφορετικό από το αρχικό έγγραφο. Κατά τη διάρκεια της διαδικασίας αποσύνθεσης, το XML έγγραφο χάνει την περισσότερη διάταξή του προκειμένου να χαρτογραφηθεί στον σχεσιακό πίνακα. Εάν η διάταξη του εγγράφου είναι σημαντική, τότε πρέπει να αποθηκευτεί σε στήλη τύπου CLOB/LONG VARCHAR ή τύπου XML στήλη.

Γενικά, τα XML έγγραφα μπορούν να αποσυντεθούν εάν τα στοιχεία τους έχουν τις ακόλουθες ιδιότητες:

- ✓ Η δομή του XML εγγράφου μπορεί να αποσυντεθεί σε έναν λογικό αριθμό σχεσιακών πινάκων.
- ✓ Το έγγραφο XML χρησιμοποιείται για την ανταλλαγή στοιχείων.

✓ Η μερική ενημέρωση στοιχείων είναι συχνή και σημαντική. Οι σχεσιακοί πίνακες έχουν καλύτερη απόδοση στις μεμονωμένες ενημερώσεις στηλών.

✓ Το σχήμα XML δεν αλλάζει. Το XML σχήμα είναι συνήθως πιο ευέλικτο από το σχεσιακό. Αυτό σημαίνει ότι η εξέλιξη σχημάτων είναι ευκολότερη σε XML. Εάν το σχήμα δεν αλλάζει, η αποσύνθεση του XML εγγράφου σε σχεσιακούς πίνακες είναι λογική επιλογή.

✓ Το έγγραφο XML πρέπει να χαρτογραφηθεί στους υπάρχοντες σχεσιακούς πίνακες.

✓ Το XML έγγραφο πρέπει να επεξεργαστεί από εφαρμογή που έχει τη δυνατότητα να έχει πρόσβαση μόνο σε στοιχεία σχεσιακού πίνακα.

10.4 Δείκτες XML

Οι δείκτες παρέχουν έναν τρόπο να επιταχυνθεί η εύρεση και η πρόσβαση στοιχείων. Η DB2 9 υποστηρίζει τη δημιουργία XML δεικτών. Ένας σχεσιακός δείκτης αποτελείται από μια ή περισσότερες στήλες του πίνακα. Κάθε δείκτης XML χρησιμοποιεί ένα συγκεκριμένο μοτίβο XML εκφράσεων στα μονοπάτια των δεικτών και τις τιμές τους στα έγγραφα XML, οι οποίοι αποθηκεύονται σε μία στήλη του σχεσιακού πίνακα.

Τα στοιχεία στα οποία γίνονται συχνές επερωτήσεις και δεν απαιτούν τροποποιήσεις είναι καλή επιλογή χειρισμού ως δείκτες.

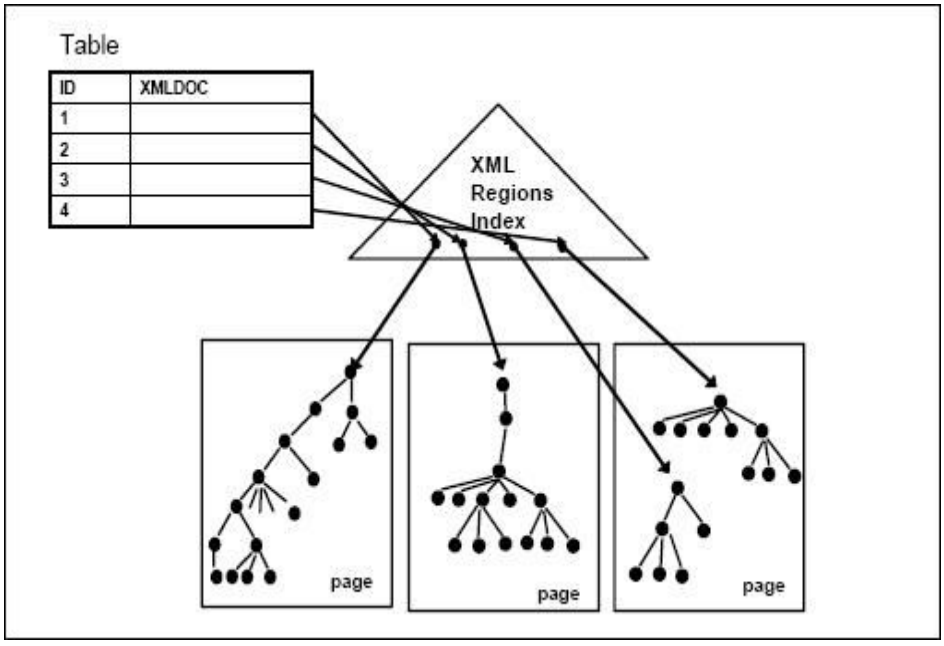
Παρομοίως με τους σχεσιακούς δείκτες, οι XML δείκτες που προσδιορίζουν XML δεδομένα, προσδιορίζουν ολόκληρη τη στήλη.

Υπάρχουν όμως και κάποιες διαφορές μεταξύ σχεσιακών και XML δεικτών, οι οποίες είναι:

- Δημιουργούνται σε στήλες τύπου XML, βασισμένοι σε εκφράσεις μονοπατιών/ διαδρομών (paths).
- Κατά τη δημιουργία του δείκτη μπορεί να προσδιοριστεί ποια διαδρομή θα δείχνει και τι τύπο δεδομένων θα χρησιμοποιεί ο δείκτης όταν χρησιμοποιείται σε XQueries.
- Ο δείκτης μπορεί να δείχνει σε μία μόνο XML στήλη, η σύνθεση δεικτών δεν επιτρέπεται.

Η DB2 σειράς 9 έχει τρεις τύπος δεικτών:

Δείκτης περιοχών XML ο οποίος περιέχει τις τοποθεσίες που βρίσκονται αποθηκευμένα όλα τα έγγραφα στην αποθήκη της DB2.



Δείκτης διαδρομών XML στηλών ο οποίος δημιουργείται αυτόματα από το σύστημα για κάθε στήλη XML που δημιουργείται σε

έναν πίνακα και δείκτης στήλης XML ο οποίος με τη χρήση του αυξάνει την αποδοτικότητα XQuery και SQL/XML ερωτημάτων.

10.5 Απεικονίσεις XML Αποτελεσμάτων

Μετά από την επεξεργασία XML στοιχείων είναι επιθυμητό η απεικόνιση των αποτελεσμάτων να γίνεται με πιο κατανοητό τρόπο. Στην DB2 9 μπορεί να προσαρμοστεί η απεικόνιση των αποτελεσμάτων κάνοντας χρήση της SQL εντολής XMLTABLE.

Η XMLTABLE είναι μια εντολή SQL/XML, η οποία εμφανίζει ως πίνακα το αποτέλεσμα μίας ερώτησης XQuery, αφού τα αποτελέσματα μίας XQuery είναι σε σειριακή διάταξη. Ο πίνακας που εμφανίζεται σαν τελικό αποτέλεσμα μπορεί να περιέχει στήλες οποιουδήποτε τύπου SQL, ακόμη και τύπου XML.

Αυτή η πινακοειδής έξοδος είναι πολύ χρήσιμη, γιατί παρέχει μία σχεσιακή άποψη των XML δεδομένων.

10.6 Βιομηχανικά Στάνταρντ και XML Σχήματα

Οι εταιρίες στις διάφορες βιομηχανίες λαμβάνουν, επεξεργάζονται, αποθηκεύουν, και στέλνουν στοιχεία καθημερινά. Τα σχήματα/η δομή των πληροφοριών μπορούν να διαφέρουν μεταξύ των εταιριών, ακόμη και μεταξύ τμημάτων της ίδιας εταιρίας. Εάν δύο εταιρίες θέλουν να ανταλλάξουν πληροφορίες, πρέπει να τα μετατρέψουν από τη μία δομή στην άλλη. Και επειδή η μετατροπή αυτή είναι συνήθως δαπανηρή και χρονοβόρα, θα ήταν χρήσιμο να υπάρξουν πρότυπα/τυποποιήσεις για τις ανταλλαγές αυτές.

Τα βιομηχανικά πρότυπα καθορίζουν προσυμφωνημένους τρόπους ανταλλαγής πληροφοριών μεταξύ των επιχειρήσεων και ενδοτμηματικά. Η ανάπτυξη των εφαρμογών χρησιμοποιώντας ένα τυποποιημένο σχήμα XML ή ένα DTD διασφαλίζει μία ανταλλαγή άνευ προβλημάτων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Anon. “Extesible Mark-up Language”. 11 Nov 2007.
< www.CRM2day.gr >

Anon. “What is XML?”. 11 Nov 2007.
< www.w3c.org >

Anon. “Introduction to XML”.
< www.w3schools.com/XML/xml_whatis.asp >

Anon. “<oXygen/> xml editor”. 3 May 2008
< www.oxygenxml.com >

Anon. “Free Xml Editor”. 8 May 2008
< www.xmlfox.com >

Anon. “xmlDraft Editor”. 8 May 2008
< www.sysonyx.com/Products/xmlDraft/index.asp >
< www.sysonyx.com/Products/xmlDraft/downloads.asp >

Benoit, Marchal. “Οδηγός της XML με Παραδείγματα” (2001). Ed Γκιούρδας Β. Μεταφρ Προκοπάκης – Φιαμπολης.
< www.in.gr/books >

Anon. “Πληροφορική και Νέες Τεχνολογίες”.
< http://dide.flo.sch.gr/Plinet/plinet.html >

Anon. “XML Mediator”.
< http://www.idealgroup.gr/IdealSystems/products/../EntireXML>

Van Der Vlist, Eric. “XML Schema”. 19 Sept 2007.
< www.xml.com/pub/a/2002/06/19/vdv-wxs.html >

Boudouri, Lina. “Text Encoding Initiative: επισκόπηση, προβλήματα και εφαρμογές”. 12 Oct 2007.
< www.ionio.gr/~boudouri/BOUNTOURI_TEI.pdf >

Hines, Matt. “Νέος Τύπος Xml για το Office 12”. 13 Oct 2007.
< www.e-pcmag.gr > or < www.cnetnews.com >

Anon. “Η Κοινοπραξία του Παγκόσμιου Ιστού Εκδίδει τα XSLT 2.0, XPath 2.0 και XQuery 1.0 ως Υποψήφιες Συστάσεις”. (2005).
< www.w3c.gr/office/pressreleases/2005/11/xslt-xquery-xpath-pressrelease.el.html >

Anon. “Το W3C Ορίζει Νέο Πρότυπο Διεθνοποιημένου Περιεχομένου στον Παγκόσμιο Ιστό”. (2007).

< www.w3c.gr/office/pressreleases/2007/04/its-pressrelease.el.html >

Peterson, David. Sperberg- McQueen, C.M. “Xml Schema 1.0 Part 2: Datatypes”. (16 Feb 2006).
< www.w3.org/TR/xmlschema11-2/ >

Prokopiadou, Georgia. “Διαλειτουργικότητα στον Δημόσιο Τομέα”.
< www.ionio.gr/~papatheodor/lessons/psi-interoperability.ppt >

Alon, Noga. Milo, Tova. “Typecheking XML Views of Relational Databases”. (2001).
< http://alpha.uhasselt.be/~lucg5503/lics2001.ps >

Anon. “ Information Management: Technical resources for IBM Information Management Software “. (2008).
< http://www.ibm.com/developerworks/db2>

Anon. “DB2 9 pureXML Guide”.
< http://www.redbooks.ibm.com>

ΠΑΡΑΡΤΗΜΑ:

ΑΛΛΗΛΟΓΡΑΦΙΑ ΕΛΛΗΝΙΚΗΣ ΑΣΤΥΝΟΜΙΑΣ police.xml

```
<?xml version="1.0"?>
<?xml-stylesheet href="..\Profile.xsl" type="text/xsl" ?>
<arx-basis xsql-timing="45" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="police.xsd">
<!--
45
-->
<application mode="DEBUG">
<header>
<logo href="bootscreen" image="logo.gif" title="Αρχική Σελίδα"/>
<navbar>
<link label="Αποσύνδεση" href="/arxeion-xml/pages/internal/arxeion/logout"/>
<link href="javascript:window.print();" label="Εκτύπωση" />
<link href="javascript:window.open('arxeion-
xml/static/police/prot_beta/help/index.html);" label="Οδηγίες" />
<link label="::" />
<link href="/arxeion-xml/pages/internal/pd/folder/browse" label="Προσωπικά
Έγγραφα" />
</navbar>
</header>
<tabs>
<tab label="Αλληλογραφία ΕΛ.ΑΣ." action="all=today" selected="yes">
<subtab label="Δημιουργία Εγγράφου" action="prot-esot-verify" />
<subtab label="Νέο Εισερχόμενο" action="prot-imp-verify" />
<subtab label="Νέο Απαντητικό" action="prot-find-apa" />
<subtab label="Ορθή Επανάληψη" action="prot-esot-dup" />
<subtab label="Σημερινά" action="all-today" />
<subtab label="Αναμονή Απάντησης" action="pending" />
<subtab label="Αναζήτηση" action="prot-find" />
<subtab label="Ενημέρωση Αξ. Υπ." action="prot-duty-off-all" />
</tab>
<tab label="Διαχείριση" action="deal-ext-all"/>
</tabs>
</application>
<page xsql-timing="30">
<!--
3
-->
<!--
4
-->
<!--
5
-->
<!--
2
```

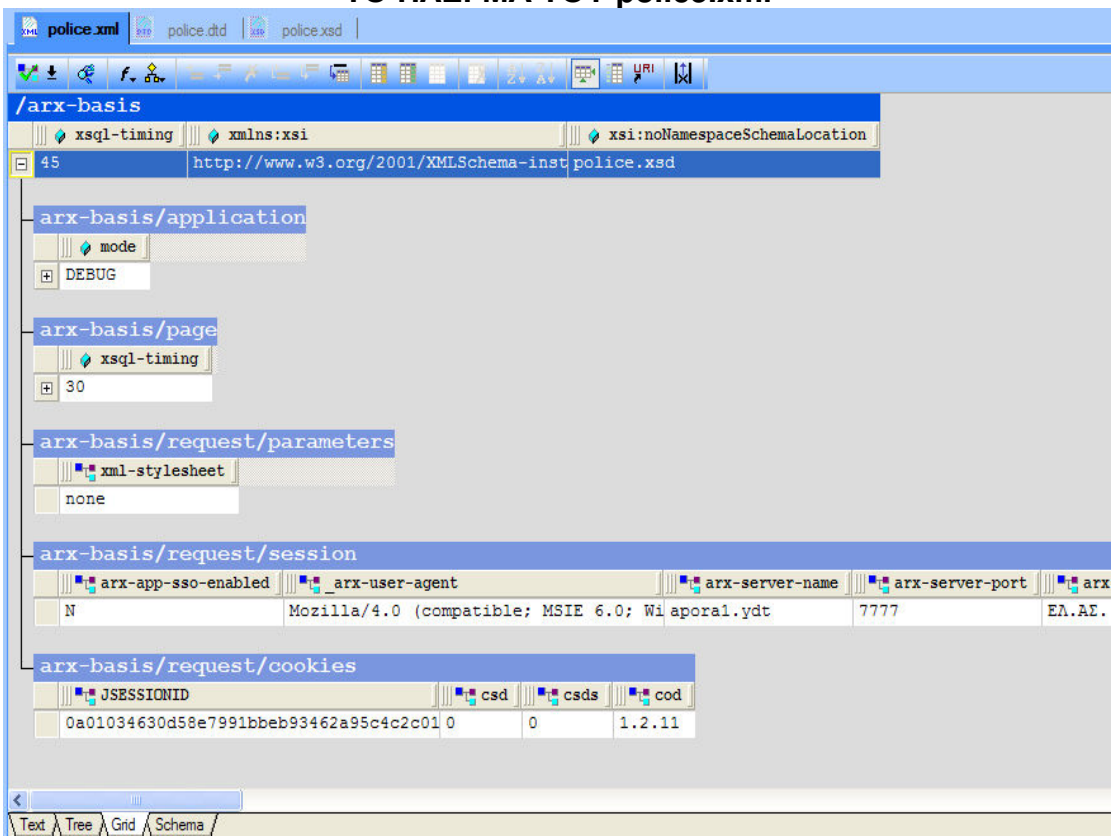
```
-->
<!--
2
-->
<!--
4
-->
<layout page-type="normal">
<column>
<!--
9
-->
<tree label="Ειδοποιήσεις Ροής Εργασιών">
<tree id="0">
<leaf id="1" label="Πρωτόκολλο P.O.L." image="wfgroup">
<leaf id="2" label="Εισερχόμενα (στο P.O.L.)" image="kp_in">
<leaf id="3" label="Εκκρεμότητες (0)" action="++++++++++++++++"
image="wfflow" style="font-weight: normal;" />

<leaf id="4" label="Κοινοποιήσεις (0)" action="+++++" image="wfntfs" />
<leaf id="5" label="Εκπρόθεσμες (0)" action="+++++" image="wfexpires" />
<leaf id="6" label="Προωθημένες (0)" action="+++++" image="wfprom" />
<leaf id="7" label="Προωθ. + Ολοκλ. (μη δβ. 0)" action="+++++"
image="wfcomplete" />
<leaf id="8" label="Εκκρεμότητες άλλων (0)" action="+++++" image="wfempl" />
<leaf id="9" label="Κοινοποιήσεις άλλων (0)" action="+++++" image="wfntfs" />
<leaf id="10" label="Ολοκληρωμένες (μη δβ. 0)" action="+++++"
image="wfcomplete" />
</leaf>
<leaf id="11" label="Οίκοθεν/ Απαντητικά (του P.O.L.)" image="esot" />
<leaf id="12" label="Εκκρεμότητες (12)" action="+++++" image="wfflow"
style="fond-weight: bold;" />
<leaf id="13" label="Κοινοποιήσεις (0)" action="+++++" image="wfntfs" />
<leaf id="14" label="Εκπρόθεσμες (0)" action="+++++" image="wfexpires" />
<leaf id="15" label="Προωθημένες (0)" action="+++++" image="wfprom" />
<leaf id="16" label="Προωθ. + Ολοκλ. (μη δβ. 0))" action="+++++" image="
wfcomplete" />
<leaf id="17" label="Εκκρεμότητες άλλων (0)" action="+++++" image="wfempl" />
<leaf id="18" label="Κοινοποιήσεις άλλων (0)" action="+++++" image="wfntfs" />
<leaf id="19" label="Ολοκληρωμένες (μη δβ. 0)" action="+++++" image="
wfcomplete" />
</leaf>
</tree>
</tree>
</column>
<column>
<titlebox width="auto">
<title style="text-align:center; font-style: italic;">Καλωσήρθατε στην Εφαρμογή
Αλληλογραφίας της ΕΛ.ΑΣ. </title>
<form name="intro" type="view" style="text-align:center; width:600px;">
```

<items>
<rows>
<row>
<item field="date" style="text-align:center; width:600px; font-weight: bold; font-size:12px;" />
</row>
</rows>
</items>
</form>
</titlebox>
</column>
<column />
</layout>
<data>
<data-set name="date">
<!--
1
-->
<ROWSET>
<ROW num="1">
<date>04/01/2008 12:27</date>
</ROW>
</ROWSET>
</data-set>
</data>
</page>
<request>
<parameters>
<xml-stylesheet>none</xml-stylesheet>
</parameters>
<session>
<arx-app-sso-enabled>N</arx-app-sso-enabled>
<_arx-user-agent>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)</_arx-user-agent>
<arx-server-name>apora1.ydt</arx-server-name>
<arx-server-port>7777</arx-server-port>
<arx-sub-title>ΕΛ.ΑΣ.</arx-sub-title>
<arx-sub-url-map>police</arx-sub-url-map>
<arx-user-roles-count>1</arx-user-roles-count>
<arx-user-id>gram-34113</arx-user-id>
<_arx-auth-sub-id>POLICE</_arx-auth-sub-id>
<arx-path-info>/police/prot_beta/bootscreen</arx-path-info>
<arx-action>bootscreen</arx-action>
<arx-request-uri>/arxeion-xml/pages/police/prot_beta/bootscreen</arx-request-uri>
<arx-request-url>http://apora1.ydt:7777/arxeion-xml/pages/police/prot_beta/bootscreen</arx-request-url>
<_arx-action-guid>++++++++++++++++++++++</_arx-action-guid>
<_arx-page-action>prot/bootscreen.xsql</_arx-page-action>
<_arx-session>id>++++++++++</_arx-session>id>
<arx-user-type>NORMAL</arx-user-type>

<arx-user-guid>++++++++++++++++++</arx-user-guid>
<arx-user-can-change-password>NO</arx-user-can-change-password>
<arx-user-name>gram-34113 gram-34113</arx-user-name>
<arx-sso-logout-url />
<arx-user-role-guid>POLICESECRETARY</arx-user-role-guid>
<arx-user-position-code>POLICEGRPDEPORG34113</arx-user-position-code>
<arx-user-role>Γραμματέας</arx-user-role>
<arx-user-position>Τ.Τ. ΧΑΛΚΗΔΟΝΑΣ ΘΕΣ/ΝΙΚΗΣ</arx-user-position>
<arx-user-organization-guid>++++++++++++</arx-user-organization-guid>
<arx-user-org-role-guid>+++++++++++++++</arx-user-org-role-guid>
<arx-sub-id>POLICE</arx-sub-id>
<arx-path-context>/arxeion-xml</arx-path-context>
<arx-app-url-map>prot_beta</arx-app-url-map>
<arx-app-id>PROT_BETA</arx-app-id>
<arx-app-title>Εφαρμογή Αλληλογραφίας</arx-app-title>
<arx-app-default-action-guid>+++++++</arx-app-default-action-guid>
<arx-app-default-action>+++++++</arx-app-default-action >
<arx-app-guid>POLICEPOT_BETA</arx-app-guid>
<arx-path-base>/arxeion-xml/pages/police/prot_beta</arx-path-base>
<arx-app-static-dir>/arxeion-xml/static/police/prot_beta</arx-app-static-dir>
<arx-referer>http://apora1.ydt:7777/arxeion-
xml/pages/police/prot_beta/bootscreen</arx-referer>
<arx-prev-action>login-form</arx-prev-action>
<arx-action-title>Καλωσήρθατε στην Εφαρμογή Αλληλογραφίας της ΕΛ.ΑΣ.</arx-
action-title>
<prot_prev_list>all-today</prot_prev_list>
<main_org_guid>POLICEGRPDEPORG34113</main_org_guid>
<arx_user_guid>++++++++++++++</arx_user_guid>
<arx_user_name>gram-34113 gram-34113</arx_user_name>
<arx_user_role>Γραμματέας</arx_user_role>
<arx_user_position>Τ.Τ. ΧΑΛΚΗΔΟΝΑΣ ΘΕΣ/ΝΙΚΗΣ</arx_user_position>
</session>
<cookies>
<JSESSIONID>0a01034630d58e7991bbeb93462a95c4c2c01a3b1706.e38LbNeTahe
Sci0NaO0</JSESSIONID>
<csd>0</csd>
<csds>0</csds>
<cod>1.2.11</cod>
</cookies>
</request>
</arx-basis>

ΤΟ ΠΛΕΓΜΑ ΤΟΥ police.xml



ΚΟΜΜΑΤΙ ΤΟΥ ΔΙΑΓΡΑΜΜΑΤΟΣ police.xsd