ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Υλοποίηση εργαλείου μετατροπής μιας σχεσιακής και αντικειμενοσχεσιακής βάσης δεδομένων σε αντίστοιχη XML μορφής

Του φοιτητή
Μπέλλα Μιχάλη
Αρ. Μητρώου: 05/2799

Επιβλέπων καθηγητής
κος Κεραμόπουλος Ευκλείδης

Θεσσαλονίκη 2010

Στην οικογένεια μου

Στη Μαρία

Περίληψη

Η παρούσα πτυχιακή εργασία ασχολείται με την υλοποίηση ενός εργαλείου μετατροπής σχεσιακών και αντικειμενοσχεσιακών βάσεων δεδομένων σε αντίστοιχες XML μορφής με χρήση των δεδομένων και μεταδεδομένων τους. Η εφαρμογή αναπτύσσεται στη γλώσσα προγραμματισμού Java και το γραφικό περιβάλλον υλοποιείται με τη βοήθεια της Java Swing. Ως σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων χρησιμοποιείται η PostgreSQL και ως αντικειμενοσχεσιακό η IBM DB2. Αναλύουμε τις συναφείς τεχνολογίες και μελετάμε συνοπτικά το ConvRel [13], τον αλγόριθμο Holistic Constraint-Preserving Transformation Algorithm [22], το SilkRoute [26], το XPERANTO [27] και άλλες προσεγγίσεις [24], [25]. Τέλος περιγράφουμε αναλυτικά τους αλγορίθμους που χρησιμοποιεί η εφαρμογή για την μετατροπή και αναλύουμε την ίδια την εφαρμογή.

Abstract

This thesis addresses the issue of developing a mapping tool, that publishes relational and object-relational databases as XML; That means relational and object-relational schemata are transformed to the corresponding XML Schemata and then the database's tables are published as an XML document, using the database's data and metadata. The mapping tool is developed in Java, whereas the GUI is implemented, using the Java Swing API. We are using PostgreSQL as the relational testing dbms and IBM DB2 as the object-relational. We analyze the related technologies and talk about ConvRel [13], SilkRoute [26], the Holistic Constraint-Preserving Transformation algorithm [22], the XPERANTO [27] and other approaches [24], [25]. Finally we give a detailed -step by step- description of the algorithms that are used from the mapping tool, to perform the mapping and analyze the tool itself.

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον κύριο Ευκλείδη Κεραμόπουλο, επιβλέποντα καθηγητή αυτής της πτυχιακής εργασίας, για την όρεξη, την υπομονή, την ηρεμία του και την συμβολή του στο ανέβασμα του επιπέδου της εργασίας.

Επίσης, θέλω να ευχαριστήσω τους γονείς μου και τον αδερφό μου για τη συνεχή και αδιάκοπη υποστήριξη τους όλα αυτά τα χρόνια.

Περιεχόμενα

Εισαγωγή

Στις μέρες μας, η πληροφορία με τη γενική έννοια του όρου, είναι στην πλειοψηφία της οργανωμένη και αποθηκευμένη σε βάσεις δεδομένων. Οι επιχειρήσεις, οι τράπεζες, οι διαδικτυακές επιχειρήσεις, όλες διατηρούν τα επιχειρηματικά τους δεδομένα αποθηκευμένα σε βάσεις δεδομένων σχεσιακής, αντικειμενοσχεσιακής ή άλλης μορφής.

Με την εξέλιξη της τεχνολογίας γεννιέται και η ανάγκη της δημιουργίας διαδικασιών που θα μετατρέπουν τα δεδομένα σε μορφή που να μπορεί να αξιοποιηθεί από τις νεότερες τεχνολογίες. Οι διαδικασίες αυτές λέγονται αλγόριθμοι μετατροπής δεδομένων (mapping algorithms).

Σκοπός αυτής της εργασίας είναι η ανάπτυξη ενός εργαλείου μετατροπής των σχεσιακών και αντικειμενοσχεσιακών βάσεων δεδομένων σε αντίστοιχες XML μορφής. Αυτό σημαίνει έκδοση του σχήματος της βάσης ως XML Schema και των δεδομένων της ως XML έγγραφο έχοντας, όσο το δυνατό, μικρότερη απώλεια πληροφορίας. Για την ανάπτυξη του εργαλείου, χρησιμοποιούμε διαφορετικές τεχνολογίες τις οποίες παρουσιάζουμε αναλυτικά στην παρακάτω εργασία.

Στο κεφάλαιο 1, μιλάμε για τα μεταδεδομένα και τη χρήση τους. Αναλύουμε, τους τύπους των μεταδεδομένων γενικά και τις πληροφορίες που μας δίνουν τα μεταδεδομένα μιας βάσης δεδομένων συγκεκριμένα.

Στο κεφάλαιο 2, περιγράφουμε αναλυτικά το JDBC API , δίνοντας έμφαση σε ιδιαίτερα συστατικά στοιχεία του, όπως ο διαχειριστής οδηγών JDBC. Παρουσιάζουμε τους διαφορετικούς τύπους JDBC οδηγών, καθώς και τις βασικές κλάσεις και μεθόδους τους που συμβάλλουν στην ανάκτηση δεδομένων και μεταδεδομένων από την βάση, παραθέτοντας και παραδείγματα κώδικα.

Στο κεφάλαιο 3, εστιάζουμε στην αυτοπεριγραφόμενη γλώσσα σήμανσης XML (eXtensible Markup Language) , μιλάμε για το XML Schema και τα έγκυρα XML έγγραφα. Επίσης, παραθέτουμε κάποια από τα βασικά στοιχεία της XML δομής.

Στο κεφάλαιο 4, κάνουμε μια ιστορική αναδρομή στα πρώτα Συστήματα Διαχείρισης Βάσεων Δεδομένων. Ακολουθώντας αυτή την πορεία, φτάνουμε από τα σχεσιακά στα αντικειμενοσχεσιακά συστήματα και κάνουμε μια σύγκριση ανάμεσα τους.

Στο κεφάλαιο 5, παραθέτουμε τους αλγόριθμους που προέκυψαν από την έρευνα στα πλαίσια αυτής της εργασίας. Μελετάμε συνοπτικά τους αλγορίθμους ConvRel, SilkRoute , τον αλγόριθμο Holistic Constraint-Preserving Transformation algorithm και το XPERANTO και παραθέτουμε και άλλες προσεγγίσεις.

Στο κεφάλαιο 6, παρουσιάζουμε τις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής καθώς και -βήμα προς βήμα- τους αλγόριθμους μετατροπής (mapping algorithms) που υλοποιήθηκαν. Επίσης, αναλύουμε την εφαρμογή σε επίπεδο κλάσεων, εξετάζοντας και τις σημαντικότερες μεθόδους της καθεμιάς ξεχωριστά.

Στο κεφάλαιο 7, το οποίο είναι περισσότερο μια αυτοτελής ενότητα, παραθέτουμε τα συμπεράσματα της εργασίας.

Ακολουθούν η βιβλιογραφία και το Παράρτημα.

1. Μεταδεδομένα (Metadata)

1.1. Εισαγωγή

Η αύξηση του βαθμού ευκολίας παραγωγής και διανομής περιεχομένου, στις μέρες μας, αυξάνει καθημερινά σε σημαντικό βαθμό τις πηγές και τον όγκο της πληροφορίας που υπάρχει διαθέσιμη στο χρήστη.

Επίσης, η ευρεία χρήση του Ίντερνετ, παράλληλα με την ανάπτυξη ολοκληρωμένων διαδικτυακών υπηρεσιών που παράγουν, αποθηκεύουν ή διαχειρίζονται περιεχόμενο οδήγησε στην ανάγκη οργάνωσης αυτής της πληροφορίας. Η τεχνολογία των μεταδεδομένων δημιουργήθηκε για να οργανώσει αυτή την πληροφορία.

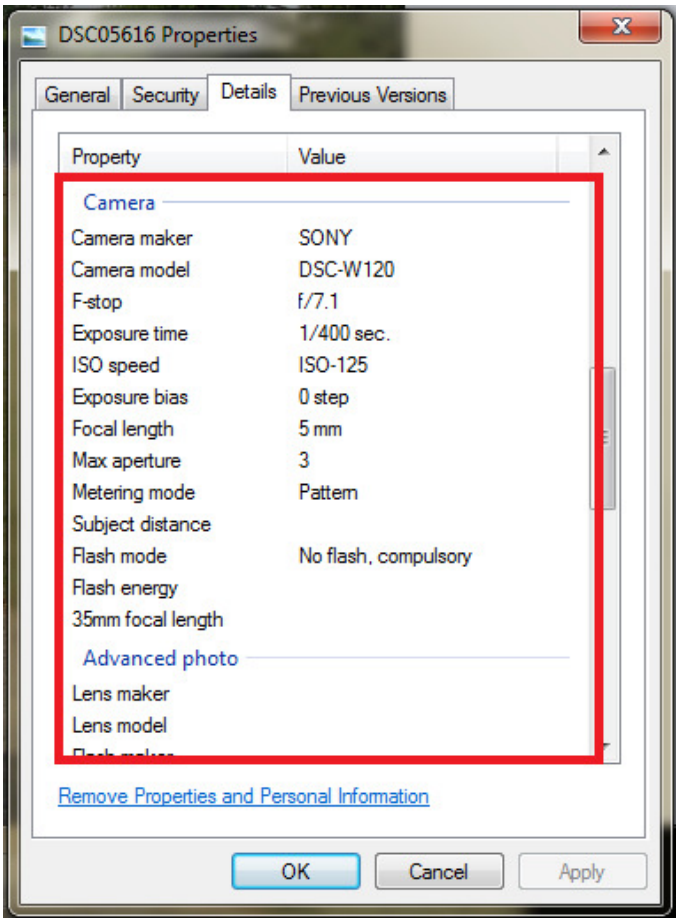
1.2. Τί είναι τα Μεταδεδομένα ;

Τα μεταδεδομένα είναι δεδομένα για τα δεδομένα (data about data). Περιγράφουν ηλεκτρονικά αποθηκευμένα δεδομένα ως προς την μορφή, τη δομή και τον τρόπο διαχείρισης τους. Η χρήση τους είναι ευρύτερα διαδεδομένη και δεν περιορίζεται μόνο στους σκοπούς αυτής της εργασίας. Για παράδειγμα σε κάποιες τεχνολογίες, τα περιγραφόμενα δεδομένα μπορεί να αφορούν:

- τα μέσα δημιουργίας
- τον σκοπό των δεδομένων
- την ημερομηνία και ώρα δημιουργίας
- τον δημιουργό ή συγγραφέα των δεδομένων
- τα πρότυπα (standards) που χρησιμοποιήθηκαν κ.α. [1]

Χαρακτηριστικό παράδειγμα χρήσης των μεταδεδομένων (βλ. Εικόνα 1), είναι η ενσωμάτωση στα αρχεία ψηφιακών φωτογραφιών των συνθηκών, δηλαδή των ρυθμίσεων (έκθεση στο φως, ταχύτητα διαφράγματος, ημερομηνία κτλ), με τις οποίες δημιουργήθηκε μια φωτογραφία.

Σε έναν πιο εξειδικευμένο ορισμό, τα μεταδεδομένα σύμφωνα με το NISO είναι δομημένες πληροφορίες που περιγράφουν, εξηγούν, εντοπίζουν και γενικότερα διευκολύνουν την ανάκτηση, τη χρήση και την διαχείριση μιας πηγής πληροφοριών [2].



Εικόνα 1.1 Τα μεταδεδομένα όπως φαίνονται στις ιδιότητες ενός JPEG αρχείου

1.3. Τύποι Μεταδεδομένων

Στη βιβλιογραφία εμφανίζονται πολλοί τύποι μεταδεδομένων, σχεδόν ένας για κάθε διαφορετικό τύπο περιεχομένου. Πολλοί από αυτούς θα μπορούσαν να θεωρηθούν υποκατηγορίες άλλων τύπων μεταδεδομένων. Εδώ θα

παρουσιάσουμε τους τύπους που έχουν άμεση σχέση με τα συστήματα διαχείρισης βάσεων δεδομένων.

Οι κυριότεροι τύποι μεταδεδομένων, λοιπόν είναι:

- **Περιγραφικά μεταδεδομένα (Descriptive metadata)**, που περιγράφουν μια πηγή πληροφοριών με σκοπό την εύρεση και ταυτοποίηση του περιεχομένου.
- **Δομικά μεταδεδομένα (Structural Metadata)**, που περιγράφουν τη δομή σύνθετων αντικειμένων και τον τρόπο με τον οποίο αυτά συνδέονται μεταξύ τους, όπως για παράδειγμα σε μια σχεσιακή βάση δεδομένων.
- **Διαχειριστικά μεταδεδομένα (Administrative metadata)**, που αφορούν στον τρόπο διαχείρισης μια πηγής πληροφοριών καθορίζοντας άδειες πρόσβασης του περιεχομένου, ημερομηνία και ώρα δημιουργίας κ.α.

Υποκατηγορίες των διαχειριστικών μεταδεδομένων, οι οποίες αρκετές φορές αναφέρονται στην βιβλιογραφία ως αυτοτελείς τύποι μεταδεδομένων, είναι

- τα **μεταδεδομένα Διαχείρισης Δικαιωμάτων (Rights Management metadata)**, τα οποία περιγράφουν το περιεχόμενο ως προς τα δικαιώματα πνευματικής ιδιοκτησίας,
- και τα **μεταδεδομένα Συντήρησης (Preservation metadata)** που περιέχουν πληροφορίες για την αρχειοθέτηση και τη συντήρηση – διαφύλαξη μιας πηγής πληροφοριών [2].

1.4. Ενσωματωμένα και μη-Ενσωματωμένα Μεταδεδομένα

Τα μεταδεδομένα μπορεί να είναι ενσωματωμένα (embedded) στο περιεχόμενο, όπως για παράδειγμα στα ψηφιακά αρχεία φωτογραφιών και βίντεο ή μη-ενσωματωμένα (non-embedded), όπως αποθηκεύονται συνήθως τα μεταδεδομένα αντικειμένων σε μια βάση δεδομένων.

Τα ενσωματωμένα μεταδεδομένα προσφέρουν το πλεονέκτημα ότι μεταφέρονται μαζί με το περιεχόμενο και επομένως η διαθεσιμότητα τους εξαρτάται από την διαθεσιμότητα του ίδιου του περιεχομένου. Έτσι καθίστανται πιο εύκολα στην διαχείριση τους. Συνήθως, η ενσωμάτωση τους στο περιεχόμενο γίνεται αυτόματα κατά τη στιγμή της δημιουργίας του. Παράλληλα, όμως, λόγω αυτής της ενσωμάτωσης δημιουργούν πρόβλημα πλεονάζουσας πληροφορίας, η οποία καταλαμβάνει χώρο και αυξάνει το κόστος συντήρησης.

Τα μη ενσωματωμένα δεδομένα, αντίθετα, έχουν τη δυνατότητα να αποθηκευτούν ξεχωριστά, για παράδειγμα σε μια βάση δεδομένων. Έτσι, γίνεται ευκολότερη η επεξεργασία τους, γεγονός που καθιστά τη λύση αυτή πιο ελκυστική σε θέματα αναζήτησης περιεχομένου. Επίσης, δεν δημιουργεί πρόβλημα πλεονάζουσας πληροφορίας, καθώς τα μεταδεδομένα όντας χωριστά από το περιεχόμενο που περιγράφουν μπορούν να παραλειφθούν ή και να διαγραφούν [3].

1.5. Μεταδεδομένα και Συστήματα Διαχείρισης Βάσεων Δεδομένων

Στα σχεσιακά και αντικειμενοσχεσιακά συστήματα διαχείρισης βάσεων δεδομένων, τα μεταδεδομένα αποκτούν ιδιαίτερη σημασία. Περιγράφουν τη δομή της βάσης ή ενός συγκεκριμένου τύπου δεδομένων όπως ενός αντικειμένου ή ενός πίνακα.

Η ανάκτηση των μεταδεδομένων μιας βάσης δεδομένων μας δίνει πληροφορίες μεταξύ άλλων, για :

- Τους πίνακες και τους υποπίνακες τους, που την αποτελούν, το μέγεθος τους, τον αριθμό εγγραφών τους, τον τύπο τους

- Τις στήλες (columns) κάθε πίνακα, το όνομα τους και τον τύπο των δεδομένων τους.
- Τα κύρια και ξένα κλειδιά κάθε πίνακα
- Το όνομα του σχήματος (schema)

Χρησιμοποιώντας και συνδυάζοντας κατάλληλα τα metadata είναι δυνατή η αναπαράσταση του σχήματος της βάσης. Τα μεταδεδομένα που μας δίνουν τις παραπάνω πληροφορίες αναφέρονται στην ορολογία των βάσεων δεδομένων ως ‘κατάλογος’ (catalog). Η SQL ορίζει έναν ενιαίο τρόπο πρόσβασης του καταλόγου, το INFORMATION_SCHEMA, το οποίο όμως δεν υποστηρίζεται από όλα τα συστήματα βάσεων δεδομένων [3].

1.6. Χρήση των Μεταδεδομένων

Αν και τα μεταδεδομένα ως τεχνολογία, όπως αναφέραμε στην εισαγωγή αυτού του κεφαλαίου, δημιουργήθηκαν με πρώτο και κύριο στόχο την ευκολότερη εύρεση περιεχομένου, νέοι τύποι μεταδεδομένων έχουν αναπτυχθεί για να εξυπηρετούν άλλους σκοπούς όπως [4]:

- Αξιολόγηση Περιεχομένου (content rating)
- Ασφάλεια
- Διαχειριστικός έλεγχος
- Διαχείριση πληροφοριών
- Θέματα πνευματικής ιδιοκτησίας
- Θέματα συντήρησης
- Προσωπικές πληροφορίες

Στα πλαίσια αυτής της εργασίας θα χρησιμοποιήσουμε τα μεταδεδομένα μιας βάσης για να δημιουργήσουμε μια αντίστοιχη XML μορφής, με βάση έναν συγκεκριμένο αλγόριθμο. Θα αναπαραστήσουμε το σχήμα της βάσης ως ένα XML Schema και θα αναπαραστήσουμε την ίδια τη βάση και τα δεδομένα της ως ένα XML αρχείο.

1.7. Περίληψη Κεφαλαίου

Συνοπτικά, τα μεταδεδομένα ορίζονται ως δεδομένα που περιγράφουν δεδομένα ή αλλιώς ως δομημένες πληροφορίες που περιγράφουν, εξηγούν, εντοπίζουν και γενικότερα διευκολύνουν την ανάκτηση, τη χρήση και την διαχείριση μιας πηγής πληροφοριών. Τα μεταδεδομένα μπορεί να είναι ενσωματωμένα στο περιεχόμενο ή μη-ενσωματωμένα. Στη δεύτερη περίπτωση αποθηκεύονται ξεχωριστά σε κάποια πηγή πληροφοριών όπως μια βάση δεδομένων.

Στα συστήματα διαχείρισης βάσεων δεδομένων τα μεταδεδομένα μπορούν να μας δώσουν πληροφορίες πολύ σημαντικές για την δομή της ίδιας της βάσης και του σχήματος της, ενός αντικειμένου ή ενός πίνακα. Έτσι, γίνεται δυνατή η αναπαράσταση της βάσης και του σχήματος της, με τον κατάλληλο συνδυασμό των πληροφοριών που μας δίνουν τα μεταδεδομένα.

Είναι επίσης δυνατή, η μετατροπή μιας βάσης σε μια άλλη μορφή, όπως για παράδειγμα σε XML, με χρήση των κατάλληλων εργαλείων και εφαρμογών. Ένα τέτοιο εργαλείο είναι το JDBC API , το οποίο μας επιτρέπει άμεση και εύκολη πρόσβαση στα μεταδεδομένα μιας βάσης δεδομένων και εξετάζεται στο επόμενο κεφάλαιο.

2. Το JDBC

Σε αυτό το κεφάλαιο, θα προσπαθήσουμε να δώσουμε μια κατατοπιστική περιγραφή του JDBC, εξετάζοντας τα βασικότερα στοιχεία του. Οι ονομασίες Java και JDBC είναι κατοχυρωμένα σήματα της εταιρείας Oracle Inc. στην Αμερική και σε άλλες χώρες.

2.1. Εισαγωγή

Είναι γεγονός ότι στις μέρες μας μεγάλος αριθμός εφαρμογών αναπτύσσονται σε Java. Η ανάπτυξη και εξέλιξη των ίδιων των γλωσσών προγραμματισμού φέρνει μαζί της και νέες τεχνολογίες που διευκολύνουν την ανάπτυξη και την ολοκλήρωση των εφαρμογών.

2.2. Τι είναι το JDBC ;

Το JDBC είναι μια διαπλατφορμική διεπιφάνεια (cross–platform interface) που συνδέει τις σχεσιακές βάσεις δεδομένων με την γλώσσα προγραμματισμού Java. Το JDBC έχει καθιερωθεί ως πρότυπο API[1] για την πρόσβαση και ανάκτηση δεδομένων από σχεσιακές βάσεις δεδομένων όπως οι MySQL, Oracle, IBM DB2 [5].

Σημείωση: *Σύμφωνα με την Sun Microsystems Inc. το JDBC δεν συνιστούσε ακρώνυμο για το Java DataBase Connectivity, όπως υποστηριζόταν από κάποιους προγραμματιστές και μηχανικούς λογισμικού. Ωστόσο, μετά την εξαγορά της από την Oracle Inc. το JDBC αναφέρεται ως "The Java Database Connectivity(JDBC)"*

Σήμερα που η ανάπτυξη ολοκληρωμένων εφαρμογών έχει άμεση σχέση με την διαχείριση και χρήση των μεταδεδομένων ενός πόρου πληροφοριών, χρησιμοποιώντας το JDBC API, μπορούμε να έχουμε άμεση και εύκολη πρόσβαση στα δεδομένα και τα μεταδεδομένα μιας σχεσιακής βάσης δεδομένων ή και περισσότερων, χρησιμοποιώντας την ίδια εφαρμογή.

API[1] : Application Programming Interface ή Διεπαφή Προγραμματισμού Εφαρμογών

Ως αποτέλεσμα οι εφαρμογές έχουν μεγαλύτερη αποδοτικότητα και μειωμένο κόστος ανάπτυξης, αφού δεν χρειάζεται να προγραμματιστούν διαφορετικές εφαρμογές για να έχουμε πρόσβαση σε διαφορετικές βάσεις δεδομένων. Για να επιτευχθεί αυτό, το JDBC χρησιμοποιεί ένα σύστημα διαχείρισης οδηγών (driver manager).

Παρακάτω γίνεται μια αναφορά στους πιο γνωστούς JDBC οδηγούς, με βάση τα συστήματα βάσεων δεδομένων που υποστηρίζουν καθώς και στον διαχειριστή οδηγών JDBC.

2.3. JDBC Οδηγοί

2.3.1. Τι είναι ένας JDBC οδηγός;

Ένας JDBC οδηγός είναι ένα λογισμικό που υλοποιεί την αλληλεπίδραση μιας εφαρμογής Java με μια βάση δεδομένων. Ο οδηγός μπορεί να είναι υλοποιημένος σε κώδικα Java ή σε άλλες γλώσσες προγραμματισμού. Για κάθε διαφορετικό Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ), υπάρχει ένα οδηγός τον οποίο παράγει η εταιρεία που εμπορεύεται το συγκεκριμένο ΣΔΒΔ.

Ο οδηγός είναι ουσιαστικά ένα λογισμικό το οποίο υλοποιεί όλες τις διεπαφές (interfaces) και κλάσεις (classes) του JDBC API που είναι απαραίτητες για τη σύνδεση και την αλληλεπίδραση με ένα ΣΔΒΔ μια συγκεκριμένης εταιρείας [6]. Μεταφράζει όλες τις κλήσεις που απευθύνονται στο ODBC ή στο JDBC σε εξειδικευμένες κλήσεις επί του αντίστοιχου ΣΔΒΔ. Οι οδηγοί κατατάσσονται σε τέσσερις κατηγορίες (τύπους) οι οποίες αναφέρονται παρακάτω.

2.3.2. Τύποι JDBC οδηγών

Οι οδηγοί JDBC χωρίζονται σε τέσσερις διαφορετικούς τύπους οι οποίοι αναφέρονται παρακάτω. Ο προγραμματιστής μπορεί να διαλέξει ποιον οδηγό θα χρησιμοποιήσει.

Τύπος I – Γέφυρα JDBC – ODBC

Μεταφράζει τις κλήσεις προς το JDBC API σε κλήσεις ODBC , οι οποίες αργότερα αποστέλλονται στον ODBC οδηγό [6]. Το ODBC βλέπει αυτό τον οδηγό, ως μια εφαρμογή και ο οδηγός υλοποιεί το JDBC API για κάθε βάση δεδομένων που διαθέτει έναν ODBC οδηγό. Παρέχεται με την εγκατάσταση της Java, θεωρείται όμως αργός οδηγός και η Sun στην τεκμηρίωση του οδηγού, προτείνει την χρήση του για πειραματικούς λόγους ή όπου μια εναλλακτική λύση δεν είναι διαθέσιμη. Σε κάποιες περιπτώσεις είναι απαραίτητη η εγκατάσταση επιπλέον κώδικα για συγκεκριμένα ΣΔΒΔ.

Τύπος II – Απευθείας μετάφραση σε API άμεσης συγγένειας

Ένας οδηγός αυτού τύπου μεταφράζει τις κλήσεις προς το JDBC API σε κλήσεις προς το API ενός συγκεκριμένου ΣΔΒΔ. Είναι συνήθως υλοποιημένος σε άλλη γλώσσα προγραμματισμού (C / C++) αλλά παρέχει ένα εξωτερικό στρώμα Java για τον προγραμματισμό και τη σύνδεση με Java εφαρμογές. Όπως και στον Τύπο I, είναι απαραίτητη η εγκατάσταση επιπλέον κώδικα στους υπολογιστές που τον χρησιμοποιούν.

Τύπος III – Γέφυρες Δικτύου

Είναι οδηγός υλοποιημένος εξ' ολοκλήρου σε Java (pure Java driver). Μεταφράζει τις κλήσεις προς το JDBC API σε κλήσεις προς ένα γενικό πρωτόκολλο δικτύου , οι οποίες αργότερα μεταφράζονται με τη χρήση ενός ενδιάμεσου συστατικού (middleware component, π.χ.Server) σε κλήσεις προς το πρωτόκολλο ενός συγκεκριμένου ΣΔΒΔ. Το ενδιάμεσο συστατικό μπορεί να χρησιμοποιήσει οποιονδήποτε οδηγό για να επικοινωνήσει με το ΣΔΒΔ και η επιλογή εξαρτάται από την εταιρεία που υλοποιεί και εμπορεύεται το ΣΔΒΔ. Ο συγκεκριμένος οδηγός, προσφέρει μεγαλύτερη ευελιξία καθώς έχει τη δυνατότητα να συνδέσει όλες τις Java εφαρμογές που υπάρχουν στο δίκτυο, με διαφορετικές βάσεις δεδομένων.

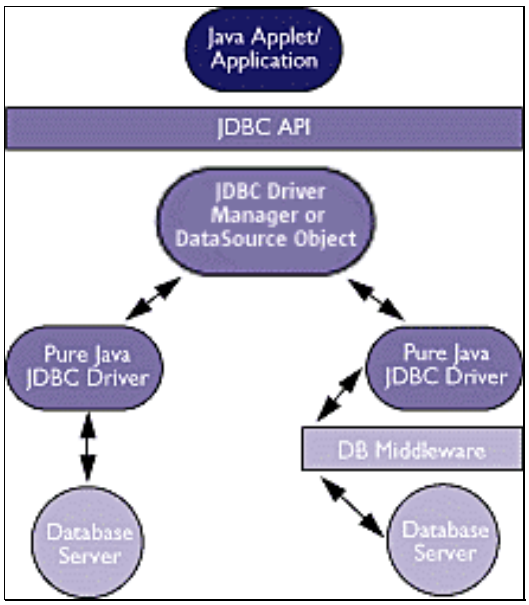
Τύπος IV – Απευθείας μετάφραση στο πρωτόκολλο του ΣΔΒΔ

Είναι οδηγός γραμμένος σε Java και σε αντίθεση με τους οδηγούς τύπου III μεταφράζει απευθείας τις κλήσεις προς το JDBC σε κλήσεις προς το συγκεκριμένο δικτυακό πρωτόκολλο που χρησιμοποιεί το ΣΔΒΔ. Έτσι μια Java εφαρμογή επικοινωνεί απευθείας με τον εξυπηρετητή της βάσης δεδομένων (Database Server) γεγονός που καθιστά τους οδηγούς αυτού του τύπου κατάλληλους για λύσης Intranet. Οδηγοί αυτού του τύπου αναπτύσσονται από όλες τις μεγάλες εταιρείες , όπως για παράδειγμα ο Red Brick της IBM.

2.4. Ο JDBC διαχειριστής οδηγών (driver manager)

Ο διαχειριστής οδηγών του JDBC είναι μια κλάση της Java. Σύμφωνα με την τεκμηρίωση που υπάρχει στην ιστοσελίδα της Oracle, αποτελεί την κύρια υπηρεσία διαχείρισης JDBC οδηγών, παρέχοντας όλες τις απαραίτητες μεθόδους για την διαχείριση τους καθώς και την υλοποίηση της αλληλεπίδρασης μεταξύ της εφαρμογής και της βάσης δεδομένων με τη χρήση οδηγών (drivers) JDBC.

2.4.1. Πώς λειτουργεί ο διαχειριστής οδηγών;



Εικόνα 2.1. Ο ρόλος του JDBC driver manager

Ο διαχειριστής οδηγών αρχικά φορτώνει τους οδηγούς για όλες τις διαφορετικές βάσεις δεδομένων στις οποίες θέλουμε η εφαρμογή μας να έχει πρόσβαση. Στην περίπτωση προσπάθειας δημιουργίας μιας σύνδεσης με μια βάση δεδομένων, ο διαχειριστής οδηγών αναζητά τον κατάλληλο οδηγό για τη συγκεκριμένη βάση προκειμένου να επιτευχθεί η σύνδεση.

Αναλυτικότερα, κατά την αρχικοποίηση της, η κλάση DriverManager προσπαθεί να φορτώσει τις κλάσεις των οδηγών που έχουν οριστεί στην ιδιότητα συστήματος “jdbc.drivers“.

Χρησιμοποιώντας την ιδιότητα “ jdbc.drivers” , δίνεται η δυνατότητα στο χρήστη να προσαρμόσει τον οδηγό που θέλει να χρησιμοποιήσει. Για παράδειγμα στο λειτουργικό σύστημα Linux ο χρήστης μπορεί να επεξεργαστεί το αρχείο ιδιοτήτων που βρίσκεται στο ~/.hotjava/properties και να ορίσει τον δικό του προσαρμοσμένο JDBC οδηγό ως εξής:

jdbc.drivers = my.custom.Driver

Η δεξιά πλευρά αυτής της εξίσωσης αποτελείται από το όνομα της κλάσης του οδηγού JDBC (driver Class Name).

Ο χρήστης μπορεί να ορίσει και περισσότερους από έναν οδηγούς χωρίζοντας τους με άνω-κάτω τελεία (**:**). Για παράδειγμα:

jdbc.drivers = my.custom.Driver**:**some.other.Driver

Είναι δυνατή επίσης η δήλωση – φόρτωση ενός JDBC οδηγού απευθείας από το χρήστη μέσα στον κώδικα. Η φόρτωση του οδηγού μπορεί να γίνει ως εξής:

Class.forName("org.postgresql.Driver");

Όταν καλείται η μέθοδος getConnection() της κλάσης DriverManager , η τελευταία προσπαθεί να εντοπίσει τον κατάλληλο οδηγό JDBC ανάμεσα σε αυτούς που δηλώθηκαν κατά την αρχικοποίηση της κλάσης και σε αυτούς που δηλώθηκαν από τον χρήστη μέσα στον κώδικα [7].

Σημείωση: *Από την έκδοση 2.0 του JDBC API υποστηρίζεται μια νέα κλάση, η DataSource, η οποία σύμφωνα με την τεκμηρίωση της Java, αποτελεί τον προτεινόμενο τρόπο σύνδεσης σε μια πηγή δεδομένων (data source), όταν αυτό είναι δυνατό.*

2.5. JDBC και Μεταδεδομένα

Όπως αναφέρθηκε στο κεφάλαιο 1, τα μεταδεδομένα είναι δεδομένα που περιγράφουν δεδομένα. Συγκεκριμένα, στις βάσεις δεδομένων, τα μεταδεδομένα περιγράφουν όλα τα απαραίτητα στοιχεία που πρέπει να έχουμε στη διάθεση μας για να διαχειριστούμε κατάλληλα τη βάση (βλέπε παράγραφο 1.4).

Ο ρόλος του JDBC είναι διαμεσολαβητικός ανάμεσα σε μια Java εφαρμογή και ένα σχεσιακό ή αντικειμενοσχεσιακό σύστημα διαχείρισης βάσεων δεδομένων. Ο προγραμματιστής χρησιμοποιεί το JDBC API και όλα τα εργαλεία που αυτό παρέχει για να εγκαταστήσει μια σύνδεση με τη βάση και να μπορεί να στέλνει αιτήματα SQL και να λαμβάνει αποτελέσματα.

2.5.1. Η σύνδεση (Connection)

Η διεπαφή Connection της Java χρησιμοποιείται για να δηλώσουμε ένα αντικείμενο αυτού του τύπου στο οποίο αποθηκεύουμε τη σύνδεση με τη βάση. Η σύνδεση με τη βάση γίνεται με τη βοήθεια της τάξης DriverManager χρησιμοποιώντας τη μέθοδο της “getConnection” η οποία λαμβάνει ως παραμέτρους το url, το όνομα χρήστη (username) και τον κωδικό (password) που χρειαζόμαστε για να έχουμε πρόσβαση στη βάση.

2.5.2. Το URL

Το url της βάσης αποτελείται από το πρωτόκολλο σύνδεσης με τη βάση (JDBC, ODBC κτλ.), το όνομα του οδηγού της βάσης , τη διεύθυνση της, τη θύρα από την οποία αποκτάμε πρόσβαση και το όνομα της βάσης, όπως φαίνεται στο παρακάτω παράδειγμα.

```
jdbc:postgresql://aetos.it.teithe.gr:5432/dblab6
```

Σημείωση: *Για τους ODBC οδηγούς το πρώτο κομμάτι του url είναι πάντα «jdbc:odbc».Για τους JDBC οδηγούς, ο προγραμματιστής θα πρέπει να συμβουλευτεί τον κατασκευαστή του οδηγού.*

2.5.3. Η διεπαφή Statement

Η Statement είναι μια διεπαφή (interface) της Java. Ένα αντικείμενο τέτοιου τύπου χρησιμοποιείται για να αποθηκευτεί μια δήλωση, η οποία είναι ουσιαστικά αντικείμενο του ίδιου τύπου, με την οποία μπορούμε να στείλουμε SQL ερωτήματα (queries) στη βάση δεδομένων και να επιστρέψουμε τα αποτελέσματα τους.

Το αντικείμενο τύπου Statement που αποθηκεύεται, δημιουργείται με τη βοήθεια της τάξης Connection, και πιο συγκεκριμένα του αντικείμενου τύπου Connection, χρησιμοποιώντας τη μέθοδο “createStatement” η οποία μπορεί να λάβει διάφορες παραμέτρους.

Τα αιτήματα προς τη βάση δεδομένων γίνονται με δύο τρόπους μέσω ενός αντικειμένου τύπου Statement.

Όσον αφορά τα SELECT ερωτήματα, αυτά πραγματοποιούνται με τη μέθοδο της τάξης Statement , “executeQuery()” η οποία λαμβάνει ως παράμετρο ένα String που περιέχει το επερώτημα προς τη βάση.

Αντίθετα όλα τα υπόλοιπα ερωτήματα, όπως για παράδειγμα INSERT, UPDATE, DELETE, πραγματοποιούνται με τη μέθοδο “executeUpdate()”, επίσης της τάξης Statement, η οποία στην πιο απλή μορφή της λαμβάνει ως παράμετρο επίσης ένα String που περιέχει το επερώτημα προς τη βάση δεδομένων.

Η διεπαφή PreparedStatement είναι μια υποδιεπαφή της Statement. Υπό αυτή την έννοια κληρονομεί όλη τη λειτουργικότητα της αλλά και την επεκτείνει περαιτέρω. Ένα αντικείμενο τύπου PreparedStatement χρησιμοποιείται για να αποθηκευτεί μια προμεταγλωττισμένη (precompiled) δήλωση. Αυτό το αντικείμενο μπορεί να χρησιμοποιηθεί μετά για να εκτελέσουμε αποδοτικότερα το ίδιο προμεταγλωττισμένο SQL ερώτημα πολλές φορές.

Λόγω της προμεταγλώττισης του και αντίθετα με ένα ερώτημα που αποθηκεύεται σε ένα αντικείμενο τύπου Statement, το ερώτημα αυτό μεταγλωττίζεται από την αρχή στο ΣΔΒΔ και έτσι είναι έτοιμο να εκτελεστεί απευθείας οποιαδήποτε στιγμή. Με αυτό τον τρόπο ελαττώνεται ο χρόνος εκτέλεσης.

Ένα άλλο πλεονέκτημα που έχει ένα PreparedStatement είναι η δυνατότητα για δυναμική ανάθεση παραμέτρων. Οι παράμετροι ορίζονται με το αγγλικό ερωτηματικό “ ? ”. Με αυτό τον τρόπο μπορούμε να εκτελούμε πολλές φορές ένα ερώτημα, αναθέτοντας στις παραμέτρους διαφορετικές τιμές. Για παράδειγμα ένα SQL SELECT ερώτημα, όπως το παρακάτω θα μεταγλωττιστεί μια φορά αλλά θα μπορεί να εκτελεστεί με διαφορετικές τιμές κάθε φορά, επιστρέφοντας διαφορετικά αποτελέσματα.

```
SELECT * FROM ATHLETE A WHERE A.weight>?
```

2.5.4. Το ResultSet

Τα αποτελέσματα ενός ερωτήματος (query) προς τη βάση, επιστρέφονται στην εφαρμογή με τη μορφή του ResultSet. Το ResultSet είναι μια διεπαφή (interface) της Java. Ένα αντικείμενο τύπου ResultSet αποθηκεύει ένα πίνακα

αποτελεσμάτων, αντίστοιχο του result set της βάσης δεδομένων, ο οποίος περιέχει όλες τις εγγραφές που ικανοποιούν τα κριτήρια του ερωτήματος που έγινε στη βάση.

Επίσης, διαθέτει έναν δείκτη, τον δρομέα (cursor), ο οποίος δείχνει κάθε φορά μια συγκεκριμένη γραμμή (row) του πίνακα. Η αρχική θέση του δρομέα είναι πριν από την πρώτη γραμμή. Με τη μέθοδο next() μπορούμε να μετακινήσουμε τον δρομέα στην επόμενη γραμμή. Η μέθοδος next() επιστρέφει τιμή “false” όταν φτάσει στο τέλος του πίνακα. Με τη μέθοδο beforeFirst(), μετακινούμε τον κέρσορα ακριβώς πριν από την πρώτη γραμμή του πίνακα, για να μπορούμε διατρέξουμε τις εγγραφές μια-μια ξεκινώντας από την πρώτη.

Σημείωση: *Ένα αντικείμενο τύπου ResultSet είναι εξορισμού μη ενημερώσιμο (non updatable) και διαθέτει έναν κέρσορα που κινείται μόνο μπροστά. Συνεπώς μπορούμε να διατρέξουμε τον πίνακα μόνο μια φορά, από την πρώτη ως την τελευταία γραμμή.*

Ανάλογα με τις απαιτήσεις της εφαρμογής μπορούμε να ορίσουμε τον κέρσορα ως ενημερώσιμο (updatable) και τον δρομέα να κινείται και προς τις δύο κατευθύνσεις (scrollable) όπως φαίνεται στο παρακάτω κομμάτι κώδικα. Εδώ, το TYPE_SCROLL_INSENSITIVE δηλώνει πως το ResultSet είναι scrollable, και δεν επηρεάζεται από αλλαγές που μπορεί να κάνουν άλλοι, όσο αυτό είναι ανοιχτό.

```
Statement stmt = con.createStatement(
                                        ResultSet.TYPE_SCROLL_INSENSITIVE,
                                        ResultSet.CONCUR_UPDATABLE);
```

2.6. Ανάκτηση Μεταδεδομένων

Η ανάκτηση των μεταδεδομένων μιας βάσης δεδομένων, στη συγκεκριμένη περίπτωση, γίνεται μέσω μιας Java εφαρμογής η οποία χρησιμοποιεί την

τεχνολογία JDBC και το JDBC API για να έχει πρόσβαση στην βάση και τα μεταδεδομένα της.

2.6.1. Τύποι και Ανάκτηση Μεταδεδομένων στην Τεχνολογία JDBC

Στην τεχνολογία JDBC υπάρχουν οι εξής τρεις (3) τύποι μεταδεδομένων:

- Τα ResultSetMetadata, τα οποία μας δίνουν πληροφορίες για τους τύπους και τις ιδιότητες των στηλών ενός αντικειμένου ResultSet.

- Τα DatabaseMetadata, τα οποία μας δίνουν πληροφορίες για τη δομή μιας βάσης δεδομένων

- Τα ParameterMetadata, τα οποία προσφέρουν πληροφορίες σχετικά με τις παραμέτρους ενός PreparedStatement

Παρακάτω θα παρουσιάσουμε αναλυτικότερα κάθε τύπο, εστιάζοντας την προσοχή μας στις μεθόδους που διαθέτουν για ανάκτηση των μεταδεδομένων και τη χρήση αυτών με παραδείγματα. Οι τύποι που αναφέρθηκαν παραπάνω είναι ουσιαστικά διεπαφές (interfaces) της Java.

ResultSetMetaData

Είναι μια διεπαφή της Java. Ένα αντικείμενο τύπου ResultSetMetaData χρησιμοποιείται για την ανάκτηση των μεταδεδομένων που αφορούν τις στήλες ενός αντικειμένου ResultSet. Υπό αυτή την έννοια, πριν απ' όλα θα πρέπει να έχουμε δημιουργήσει ένα αντικείμενο αυτού του τύπου, στο οποίο θα έχουμε αποθηκεύσει τα αποτελέσματα από την εκτέλεση ενός ερωτήματος (query) που έχουμε κάνει προς τη βάση δεδομένων χρησιμοποιώντας, όπως περιγράψαμε παραπάνω ένα αντικείμενο τύπου statement. Το παρακάτω παράδειγμα είναι αντιπροσωπευτικό.

```
String sqlQuery = "SELECT * FROM ATHLETE A";

ResultSet dataRS = statement.executeQuery(sqlQuery);

      dataRS.beforeFirst();

while(dataRS.next()){
ResultSetMetaData rsmd = dataRS.getMetaData();
}
```

Στο παραπάνω παράδειγμα, ορίζουμε ένα αντικείμενο τύπου String όπου αποθηκεύουμε το SQL ερώτημα. Χρησιμοποιώντας το αντικείμενο “statement” τύπου Statement και πιο συγκεκριμένα την μέθοδο “executeQuery()” εκτελούμε το ερώτημα στη βάση δεδομένων και αποθηκεύουμε το αποτέλεσμα του, στο αντικείμενο τύπου ResultSet, “dataRS”.

Μετακινούμε τον κέρσορα του dataRS ακριβώς πριν από την πρώτη εγγραφή του πίνακα με τη μέθοδο “beforeFirst()” (βλέπε παρ. 2.4.2.) και διατρέχουμε τον πίνακα χρησιμοποιώντας ένα βρόχο while και τη μέθοδο “next()” ως συνθήκη τέλους (παρ. 2.4.2.), γραμμή προς γραμμή. Για κάθε γραμμή παίρνουμε τα μεταδεδομένα τύπου ResultSetMetaData με τη μέθοδο “getMetaData()” της διεπαφής ResultSet και τα αποθηκεύουμε σε ένα ίδιου τύπου αντικείμενο το “rsmd”.

DataBaseMetaData

Η DataBaseMetaData αποτελεί μια ακόμα διεπαφή της Java. Χρησιμοποιούμε ένα αντικείμενο τύπου DataBaseMetaData για να ανακτήσουμε εκείνα τα μεταδεδομένα μιας βάσης δεδομένων που αφορούν τη δομή της βάσης ως σύνολο. Οι πληροφορίες που μας δίνουν αυτού του τύπου τα μεταδεδομένα, μπορεί να είναι το σχήμα της βάσης, το όνομα του ΣΔΒΔ, οι τύποι και τα ονόματα των πινάκων που την απαρτίζουν, τα κύρια και ξένα κλειδιά των πινάκων.

Για την ανάκτηση αυτών των μεταδεδομένων χρησιμοποιούμε την μέθοδο “getMetaData()” της διεπαφής (interface) Connection που αναφέρθηκε παραπάνω, και αποθηκεύουμε το αποτέλεσμα σε μια μεταβλητή τύπου DataBaseMetaData.

```
DatabaseMetaData dbmd = dbConnection.getMetaData();
```

Στο παραπάνω παράδειγμα το dbmd είναι η μεταβλητή τύπου DataBaseMetaData στην οποία αποθηκεύουμε τα μεταδεδομένα. Όπου dbConnection είναι μια μεταβλητή τύπου Connection. Η μέθοδος “getMetaData()” ανακτά ένα αντικείμενο τύπου DataBaseMetaData το οποίο περιέχει τα μεταδεδομένα της βάσης δεδομένων για την οποία το dbConnection αναπαριστά μια σύνδεση [8].

Στον πίνακα π2.1 αναφέρονται κάποιες από τις κύριες μεθόδους της διεπαφής DataBaseMetaData, τα αποτελέσματα που επιστρέφουν, καθώς και οι εξαιρέσεις (exceptions) που καλούνται σε περίπτωση σφάλματος.

Μέθοδος	**Επιστρέφει**	**Εξαιρέσεις(Exceptions)**
getSchemas()	Το όνομα του σχήματος της βάσης δεδομένων	SQLException
getDatabaseProductName()	Το όνομα του ΣΔΒΔ	SQLException
getTableTypes()	Τον τύπο των πινάκων της βάσης δεδομένων	SQLException
getTables()	Τα ονόματα των πινάκων της βάσης δεδομένων	SQLException
getPrimaryKeys()	Τα κύρια κλειδιά των πινάκων	SQLException
getImportedKeys()	Τα ξένα κλειδιά των πινάκων	SQLException
getColumns()	Τα ονόματα των στηλών	-

Πίνακας π2.1. Οι κύριες μέθοδοι της DataBaseMetaData, τα αποτελέσματα που επιστρέφουν και οι εξαιρέσεις.

ParameterMetaData

Η διεπαφή ParameterMetaData διαφέρει από τις δύο προηγούμενες στο γεγονός ότι οι πληροφορίες που μας δίνουν τα μεταδεδομένα που ανακτούμε με τη βοήθεια ενός αντικειμένου τέτοιου τύπου, αφορούν αποκλειστικά τις προμεταγλωττισμένες δηλώσεις (prepared statements). Οι πληροφορίες που παίρνουμε από τα μεταδεδομένα αυτά αφορούν τον αριθμό των παραμέτρων που έχει ένα prepared statement καθώς και τον τύπο τους.

Για να ανακτήσουμε τα μεταδεδομένα μιας προμεταγλωττισμένης δήλωσης, το πρώτο βήμα είναι να δηλώσουμε ένα αντικείμενο τύπου ParameterMetaData.

```
String sqlQuery = "SELECT * FROM ATHLETE A WHERE A.weight>?";

prStatement = dbConnection.prepareStatement(sqlQuery);

ParameterMetaData paramInfo = prStatement.getParameterMetaData();
```

Στο παραπάνω παράδειγμα, φαίνεται η δήλωση μιας String μεταβλητής, της “sqlQuery”. Αυτή η μεταβλητή περιέχει το ερώτημα προς τη βάση δεδομένων. Το ερώτημα περιέχει μια παράμετρο, η οποία δηλώνεται με αγγλικό ερωτηματικό (?). Στη δεύτερη γραμμή, χρησιμοποιώντας την μέθοδο “prepareStatement()” της διεπαφής Connection, δημιουργούμε ένα αντικείμενο τύπου PreparedStatement για να μπορούμε να στείλουμε το παραμετροποιημένο ερώτημα, “sqlQuery”, στη βάση δεδομένων [9]. Τέλος, στην τελευταία γραμμή, αποθηκεύουμε τα μεταδεδομένα σε μια μεταβλητή, την “paramInfo” η οποία είναι τύπου ParameterMetaData.

Με παρόμοιο τρόπο, χρησιμοποιώντας άλλες μεθόδους της διεπαφής ParameterMetaData μπορούμε να μάθουμε τον αριθμό και τον τύπο των παραμέτρων μιας προμεταγλωττισμένης δήλωσης (prepared statement). Τα παραδείγματα που ακολουθούν, τα οποία θεωρούνται συνέχεια του προηγούμενου παραδείγματος, υποδεικνύουν τον κώδικα που μας δίνει τις συγκεκριμένες πληροφορίες.

Για τον αριθμό των παραμέτρων:

```
int No_of_Param = paramInfo.getParameterCount();
```

Για τον τύπο των παραμέτρων:

```
String typeName = paramInfo.getParameterTypeName(i);

                              όπου i ο αριθμός της παραμέτρου, 0,1, κ.ο.κ.
```

RowSetMetaData

Μια υπο-διεπαφή (sub-interface) της διεπαφής ResultSetMetaData, η οποία δεν αναφέρεται συχνά είναι η RowSetMetaData. Ένα αντικείμενο αυτού του τύπου περιέχει πληροφορίες για τις στήλες ενός αντικειμένου RowSet. Η διεπαφή αυτή, κληρονομεί τη λειτουργικότητα της γονικής της κλάσης και την επεκτείνει, εμπλουτίζοντας την με μεθόδους που επιτρέπουν την ανάθεση τιμών σε ένα αντικείμενο τύπου RowSetMetaData.

Η περαιτέρω μελέτη αυτής της διεπαφής δεν κρίνεται απαραίτητη για τους σκοπούς αυτής της εργασίας. Για περισσότερες πληροφορίες, μπορείτε να ανατρέξετε στη βιβλιογραφία.

2.7. Τύποι δεδομένων JDBC

Στην προηγούμενη παράγραφο είδαμε τους τύπους των μεταδεδομένων στην τεχνολογία JDBC καθώς και τα βασικά γύρω από τους τρόπους ανάκτησης τους. Όπως αναφέραμε, με την ανάκτηση των μεταδεδομένων, παίρνουμε κάποιες

σημαντικές πληροφορίες για τη βάση δεδομένων, μεταξύ άλλων τύπους και ονόματα στηλών, πινάκων, σχημάτων.

Οι τύποι δεδομένων της SQL ερμηνεύονται διαφορετικά από την Java και το JDBC. Το JDBC διαθέτει τους εγγενείς τύπους δεδομένων της Java, (native Java data types). Όταν ανακτούμε μέσω του JDBC τους τύπους δεδομένων από τις στήλες ενός Συστήματος Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) το JDBC αντιστοιχίζει κάθε SQL τύπο δεδομένων σ' έναν εγγενή τύπο δεδομένων της Java.

Συνεπώς, αν θέλουμε να χρησιμοποιήσουμε τους SQL τύπους δεδομένων, τότε θα πρέπει να δημιουργήσουμε μια μέθοδο, η οποία θα κάνει αυτή την αντιστοίχηση.

2.8. Περίληψη Κεφαλαίου

Το JDBC είναι μια διαπλατφορμική διεπιφάνεια (cross–platform interface) που συνδέει τις σχεσιακές και αντικειμενοσχεσιακές βάσεις δεδομένων με την γλώσσα προγραμματισμού Java. Το JDBC αποτελεί ένα ολοκληρωμένο API για την πρόσβαση, διαχείριση και άντληση πληροφοριών από μια σχεσιακή ή αντικειμενοσχεσιακή βάση δεδομένων.

Προκειμένου να υλοποιήσει αυτή τη σύνδεση και την αλληλεπίδραση με τη βάση δεδομένων το JDBC χρησιμοποιεί ένα κομμάτι λογισμικού που λέγεται JDBC οδηγός. Ο οδηγός υλοποιεί όλες τις διεπαφές (interfaces) και κλάσεις (classes) του JDBC API που είναι απαραίτητες για τη σύνδεση και την αλληλεπίδραση με ένα ΣΔΒΔ μια συγκεκριμένης εταιρείας. Κάθε εταιρεία αναπτύσσει τον δικό της οδηγό για το δικό της ΣΔΒΔ.

Υπάρχουν τέσσερις τύποι JDBC οδηγών οι οποίοι διακρίνονται κυρίως από τους διαφορετικούς τρόπους υλοποίησης της αλληλεπίδρασης με τη βάση. Τους οδηγούς αυτούς διαχειρίζεται μια κλάση της Java η DriverManager, η οποία φορτώνει τους οδηγούς για όλα τα ΣΔΒΔ στα οποία θέλουμε να έχει πρόσβαση η εφαρμογή μας και αναζητά τον κατάλληλο οδηγό προκειμένου να επιτευχθεί μια σύνδεση.

Χρησιμοποιώντας διεπαφές της Java και του JDBC API μια εφαρμογή μπορεί να στέλνει SQL ερωτήματα και να αποκτά πρόσβαση στα δεδομένα και τα μεταδεδομένα της βάσης δεδομένων. Υπάρχουν τρείς κύριοι τύποι μεταδεδομένων στην τεχνολογία JDBC , **α) τα ResultSet Metadata**, **β) Database Metadata**, και **γ) Parameter Metadata** τα οποία μας δίνουν στοχευμένες πληροφορίες για τους τύπους, τη δομή μιας βάσης δεδομένων κ.α.

Σημαντική ιδιαιτερότητα του JDBC αποτελεί το πώς αναγνωρίζει και χειρίζεται τους τύπους δεδομένων των ΣΔΒΔ τόσο στα σχεσιακά όσο και στα αντικειμενοσχεσιακά συστήματα. Παρά το γεγονός ότι το πρότυπο της SQL περιλαμβάνει καθιερωμένους τύπους, οι εταιρείες κατασκευής ΣΔΒΔ τους προσαρμόζουν στα δικά τους συστήματα με αποτέλεσμα να παρατηρούνται μικροδιαφορές.

3. XML (eXtensible Markup Language)

Σε αυτό το κεφάλαιο θα μιλήσουμε για την XML, θα παρουσιάσουμε συνοπτικά την ιστορία και τους στόχους της καθώς και τα βασικά στοιχεία της γλώσσας σε επίπεδο δομής. Θα μιλήσουμε για το XML Schema και τα XML έγγραφα και τέλος θα αναφερθούμε στο ιεραρχικό – δενδρικό μοντέλο που διακρίνει τη δομή της XML.

3.1. Εισαγωγή

Η XML είναι μια γλώσσα σήμανσης όπως η HTML και αποτελεί διεθνές πρότυπο για την αναπαράσταση και ανταλλαγή δεδομένων στον Παγκόσμιο Ιστό. Ο όρος XML προέρχεται από τα αρχικά των λέξεων eXtensible Markup Language οι οποίες υποδηλώνουν την λειτουργική επεκτασιμότητα της σε σύγκριση με παλαιότερες γλώσσες σήμανσης όπως η HTML, SGML. Παρακάτω παρουσιάζονται αναλυτικά τα βασικότερα στοιχεία της XML που σχετίζονται με το θέμα αυτής της εργασίας.

3.2. Ιστορία

Η XML δημιουργήθηκε το 1996 από το XML Working Group του οργανισμού προτύπων W3C (World Wide Web Consortium). Είναι ουσιαστικά η εξέλιξη ενός υποσυνόλου μιας παλαιότερης γλώσσας σήμανσης, της SGML (Standard Generalized Markup Language, ISO 8879:1986).

3.3. Οι στόχοι

Το XML Working Group έθεσε εξαρχής τους στόχους της XML, λαμβάνοντας υπόψη τη ραγδαία ανάπτυξη του Παγκόσμιου Ιστού, την αυξανόμενη ζήτηση για ολοκληρωμένες εφαρμογές και την ανάγκη αλληλεπίδρασης των εφαρμογών μεταξύ τους.

Οι στόχοι αυτοί περιγράφουν μια γλώσσα με αυστηρά καθορισμένα χαρακτηριστικά ως προς τη δομή της αλλά μεγάλη ευελιξία και λειτουργικότητα. Μερικοί από τους στόχους που όρισε το XML Working Group περιγράφονται παρακάτω :

- Η XML πρέπει να είναι άμεσα χρήσιμη για το Διαδίκτυο
- Η XML πρέπει να υποστηρίζει μια μεγάλη γκάμα εφαρμογών
- Η δημιουργία XML εγγράφων πρέπει να είναι εύκολη.
- Ο αριθμός των προαιρετικών χαρακτηριστικών θα πρέπει να είναι ο ελάχιστος δυνατός, με ιδανική περίπτωση το μηδέν.

Για αναλυτικότερη παράθεση και μελέτη των στόχων της XML, όπως ορίστηκαν από το XML Working Group, ο αναγνώστης παραπέμπεται στη βιβλιογραφία [10].

3.4. Βασικά δομικά στοιχεία

Όπως αναφέρθηκε στην παράγραφο 3.3, τα XML έγγραφα θα πρέπει να δημιουργούνται εύκολα, συνεπώς η δομή τους θα πρέπει να είναι απλή. Στις απαιτήσεις της XML, συγκαταλέγεται η δυνατότητα της να είναι κατανοητή και να διαβάζεται εύκολα τόσο από τις εφαρμογές όσο και από τους ανθρώπους.

Η προσαρμογή σε αυτή την απαίτηση χαρακτηρίζει την δενδροειδή δομή ενός XML εγγράφου του οποίου η δομή αποτελείται από τα εξής βασικά στοιχεία:

- Τον πρόλογο – prolog
- Το στοιχείο ρίζα – root element
- Τις ετικέτες – Tags
- Τα χαρακτηριστικά – Attributes

Η εικόνα 3.1 απεικονίζει ένα παράδειγμα XML εγγράφου, στο οποίο διακρίνονται τα βασικά δομικά στοιχεία που περιγράφηκαν παραπάνω.

Πιο συγκεκριμένα στην 1^{η} γραμμή φαίνεται η διακήρυξη XML (XML Declaration) η οποία μαζί με τη διακήρυξη DTD (δεν εμφανίζεται στην εικόνα) αποτελούν τον XML πρόλογο. Στην 3^{η} γραμμή το στοιχείο “athlete” είναι το στοιχείο – ρίζα (root element) το οποίο στοιχείο αποτελείται από τις ετικέτες αρχής και τέλους <athlete> και </athlete> αντίστοιχα. Στην εικόνα 3.1 διακρίνονται επίσης τρία

χαρακτηριστικά. Αυτά είναι τα “version”, “encoding” και “standalone” τα οποία αποτελούν χαρακτηριστικά της XML διακήρυξης.

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

<athlete>
    <code>1</code>
    <name>Aziz</name>
    <surname>Zakari</surname>
    <gender>Male</gender>
    <date_of_birth>1976-02-09</date_of_birth>
    <weight>85</weight>
    <height>175</height>
    <country_of_origin>Ghana</country_of_origin>
    <country_of_participation>Ghana</country_of_participation>
</athlete>
```

Εικόνα 3.1. Τα βασικά στοιχεία της δενδροειδούς XML δομής

3.4.1. Ο πρόλογος (Prolog)

Σύμφωνα με το W3C κάθε XML έγγραφο που δημιουργείται θα πρέπει να περιλαμβάνει έναν πρόλογο ο οποίος αναφέρει την έκδοση της XML που χρησιμοποιεί. Ένα άλλο στοιχείο που πρέπει να περιλαμβάνεται στον πρόλογο είναι η κωδικοποίηση του εγγράφου XML. Αυτή δηλώνεται στον πρόλογο με τη βοήθεια του χαρακτηριστικού “encoding” και ορίζει τον πίνακα κωδικοποίησης (σετ χαρακτήρων) που χρησιμοποιείται στο έγγραφο.

Με αυτό τον τρόπο, οι εφαρμογές που ανταλλάσουν έγγραφα ή ο προγραμματιστής που τα διαβάζει, καταλαβαίνουν ότι πρέπει να χρησιμοποιήσουν τους συντακτικούς κανόνες της XML 1.0 έτσι ώστε να ικανοποιούν τις απαιτήσεις όπως αυτές περιγράφονται από το W3C στην τεκμηρίωση της XML 1.0 καθώς και την αντίστοιχη κωδικοποίηση προκειμένου να “μεταφράσουν” σωστά το έγγραφο [10].

Τέλος, ένα άλλο χαρακτηριστικό που χρησιμοποιείται στον πρόλογο είναι το “standalone” το οποίο παίρνει τιμές “yes” ή “no”. Το χαρακτηριστικό αυτό υποδεικνύει την σύνδεση – εξάρτηση του XML εγγράφου από ένα εξωτερικά ορισμένο XML Schema, δηλαδή ενός εγγράφου που καθορίζει την “γραμματική” που πρέπει να ακολουθεί το XML έγγραφο.

3.4.2. Το στοιχείο Ρίζα (Root Element)

Το στοιχείο-ρίζα είναι το εξωτερικό στοιχείο της δομής ενός XML εγγράφου. Σε κάθε XML έγγραφο μπορεί να υπάρχει μόνο ένα στοιχείο-ρίζα. Όπως δείχνει και η ονομασία του, δεν μπορεί να είναι κόμβος – παιδί κανενός άλλου στοιχείου. Όντας το εξωτερικό στοιχείο της δομής του εγγράφου είναι κόμβος-γονέας για όλους τους υπόλοιπους κόμβους του εγγράφου, που βρίσκονται εμφωλευμένοι σε αυτό. Στο παράδειγμα της εικόνας 3.1 το στοιχείο ρίζα είναι το “<athlete>”.

Η ανάγνωση ενός XML εγγράφου γίνεται από τα έξω προς τα μέσα, δηλαδή από τον κόμβο-γονέα στον κόμβο-παιδί. Συνεπώς, ο πρώτος κόμβος που διαβάζεται είναι το στοιχείο-ρίζα. Έτσι με τη βοήθεια του στοιχείου-ρίζα, μπορούμε να ανακτήσουμε τους κόμβους-παιδιά του (child Nodes) , οι οποίοι με τη σειρά τους μπορεί να έχουν άλλους κόμβους-παιδιά κ.ο.κ.

3.4.3. Οι ετικέτες (Tags)

Οι ετικέτες ή σημάνσεις στην XML είναι αυτές που συνθέτουν τα στοιχεία (elements) από τα οποία αποτελείται ένα κόμβος (node), και έχουν την εξής μορφή:

```
<όνομα_σήμανσης>Κείμενο</όνομα_σήμανσης>
```

Σε αντίθεση με την HTML, τα ονόματα των σημάνσεων στην XML καθορίζονται από τον συγγραφέα του εγγράφου ή την εφαρμογή που το δημιουργεί. Έτσι οι σημάνσεις αποκτούν ιδιαίτερη σημασία στα XML έγγραφα, καθώς συνηθίζεται να τους δίνονται τέτοια ονόματα που περιγράφουν το περιεχόμενο τους. Γι’ αυτό το λόγο η XML είναι γνωστή και ως αυτοπεριγραφόμενη γλώσσα (self-describing language).

Σύμφωνα με την τεκμηρίωση του W3C για την XML 1.0, κάθε σήμανση αρχής πρέπει να ακολουθείται και να “κλείνει” από μια σήμανση τέλους. Μια σήμανση τέλους διαφέρει από αυτήν της αρχής στο ότι περιλαμβάνει τον χαρακτήρα ‘ / ’ πριν το όνομα της σήμανσης για να υποδηλώσει το τέλος του συγκεκριμένου στοιχείου (element), όπως φαίνεται και στο προηγούμενο παράδειγμα.

Επίσης σημαντικό για τις σημάνσεις της XML είναι ότι οι σημάνσεις αρχής και τέλους θα πρέπει να έχουν ακριβώς το ίδιο όνομα και αυτό ισχύει γιατί η XML είναι case – sensitive γλώσσα. Αυτό σημαίνει ότι διαφορετικό είναι ένα στοιχείο με το όνομα “Athlete” από ένα άλλο με το όνομα “athlete” .

Στο παραπάνω παράδειγμα, είναι εύκολο να παρατηρήσει κανείς, ότι δεν υπάρχει κενός (λευκός) χώρος (white space) ούτε στο όνομα της σήμανσης, αλλά ούτε και στο περιεχόμενο κείμενο. Ο κενός χώρος στα ονόματα των σημάνσεων θα πρέπει να αποφεύγεται καθώς μπορεί να δημιουργήσει προβλήματα στην ανάγνωση του XML εγγράφου.

Τα κενά στοιχεία (empty elements) στην XML δηλώνονται τοποθετώντας την σήμανση τέλους ακριβώς μετά την σήμανση αρχής χωρίς κανένα χαρακτήρα (ούτε κενό) ανάμεσα τους:

```
<όνομα_σήμανσης></όνομα_σήμανσης>
```

Ένα κενό στοιχείο μπορεί επίσης να δηλωθεί έτσι:

```
<όνομα_σήμανσης/>
```

3.4.4. Τα χαρακτηριστικά (Attributes)

Όπως και στην HTML, έτσι και στην XML υπάρχουν τα χαρακτηριστικά (Attributes). Τα χαρακτηριστικά μας δίνουν περισσότερες πληροφορίες για τα στοιχεία τους [11]. Τοποθετούνται εντός της σήμανσης αρχής ενός στοιχείου, και έχουν την εξής μορφή:

```
<όνομα_σήμανσης χαρακτηριστικό=”τιμή”>Κείμενο</όνομα_σήμανσης>
```

Η τιμή των χαρακτηριστικών πρέπει πάντα να περικλείεται από μονά η διπλά εισαγωγικά. Αν η ίδια η τιμή περιλαμβάνει διπλά εισαγωγικά τότε μπορούμε να

χρησιμοποιήσουμε μονά. Κάθε στοιχείο μπορεί να έχει από κανένα έως πολλά χαρακτηριστικά. Ωστόσο κάθε χαρακτηριστικό ενός στοιχείου θα πρέπει να έχει ένα μοναδικό όνομα. Με λίγα λόγια δεν μπορεί ένα χαρακτηριστικό να εμφανίζεται παραπάνω από μία φορά στο ίδιο στοιχείο.

Όπως και στα στοιχεία, η XML μας δίνει τη δυνατότητα να δώσουμε στα χαρακτηριστικά το όνομα που επιθυμούμε, το οποίο είναι λογικό επόμενο καθώς αν τα στοιχεία μπορούν να έχουν οποιοδήποτε όνομα, πράγμα που σημαίνει ότι περιγράφουν διαφορετικά δεδομένα, τότε και τα χαρακτηριστικά που μας δίνουν επιπλέον πληροφορίες για αυτά δεν θα μπορούσαν να είναι προκαθορισμένα.

Παρά την ελευθερία που μας δίνει η XML όσον αφορά ένα απλό XML έγγραφο, όπως αναφέρθηκε παραπάνω, ορίζει και κάποιους συντακτικούς κανόνες που θα πρέπει να ικανοποιούνται για να θεωρείται ένα έγγραφο ως καλώς-ορισμένο (well-formed). Παρακάτω, θα εξηγήσουμε αναλυτικότερα τους κανόνες και τους περιορισμούς που ισχύουν.

3.5. Well Formed XML Documents

Για να είναι ένα XML έγγραφο καλώς-ορισμένο θα πρέπει να ικανοποιούνται, σε γενικές γραμμές, οι εξής συντακτικοί κανόνες:

- Κάθε XML έγγραφο να έχει (αν χρησιμοποιεί) την διακήρυξη (declaration) XML στην πρώτη γραμμή

- Κάθε XML έγγραφο θα πρέπει να έχει ένα και μοναδικό εξωτερικό στοιχείο-ρίζα (root-element)

- Κάθε στοιχείο που αρχίζει με μια σήμανση αρχής, θα πρέπει να τελειώνει με μια σήμανση τέλους ή αν το στοιχείο είναι άδειο, δηλαδή δεν εμφωλεύει μέσα του άλλα στοιχεία, να έχει τη μορφή <όνομα_σήμανσης/>.

Επίσης, όπως εξηγήθηκε αναλυτικότερα στην παράγραφο 3.4.4 οι τιμές των χαρακτηριστικών θα πρέπει να περικλείονται σε εισαγωγικά και τα ίδια τα χαρακτηριστικά να βρίσκονται μέσα στην σήμανση αρχής του στοιχείου.

Τα στοιχεία θα πρέπει να είναι σωστά εμφωλευμένα από έξω προς τα μέσα, πράγμα που σημαίνει ότι το πρώτο στοιχείο θα πρέπει να περιλαμβάνει (εμφωλεύει) μεταξύ της σήμανσης αρχής και τέλους τα υπόλοιπα v-1 στοιχεία, το δεύτερο στοιχείο θα πρέπει να περιλαμβάνει τα υπόλοιπα v-2 στοιχεία κ.ο.κ.

Στην ίδια λογική το στοιχείο ρίζα θα πρέπει να εμφωλεύει σωστά όλα τα στοιχεία του XML εγγράφου. Για περισσότερες πληροφορίες σχετικά με τα καλώς ορισμένα έγγραφα ο αναγνώστης παραπέμπεται στη βιβλιογραφία [12].

3.6. Έγκυρα XML έγγραφα και XML schema

Ένα XML έγγραφο για να είναι έγκυρο, θα πρέπει να είναι οπωσδήποτε και καλώς ορισμένο (well-formed). Το αντίθετο δεν ισχύει. Η εγκυρότητα ενός εγγράφου αποφασίζεται συγκρίνοντας το με ένα άλλο έγγραφο, το XML schema ή πιο σωστά το XSD (Xml Schema Definition) έγγραφο. Το XML schema ορίζει την «γραμματική» που πρέπει να ακολουθεί το XML έγγραφο. Σε περίπτωση που οι κανόνες που έχουν οριστεί στο XML schema ακολουθούνται από το XML έγγραφο, τότε λέμε ότι είναι ένα έγκυρο XML έγγραφο.

Ουσιαστικά η XML Schema είναι μια definition language για τα XML έγγραφα. Ενσωματώνει νέα χαρακτηριστικά [13] σε σχέση με παλαιότερες τεχνολογίες όπως τα DTD, όπως το ότι:

- Γράφεται σε XML
- Υποστηρίζει τύπους για τα στοιχεία και τα χαρακτηριστικά παρόμοιους με αυτούς των βάσεων δεδομένων
- Υποστηρίζει τύπους ενσωματωμένους αλλά και ορισμένους από τον χρήστη καθώς και σύνθετους τύπους που ορίζονται συνδυάζοντας απλούς
- Υποστηρίζει τη χρήση των Namespaces
- Είναι επεκτάσιμη για μελλοντικές προσθήκες

Υπάρχουν διάφοροι τρόποι ελέγχου της εγκυρότητας ενός XML εγγράφου. Ο τρόπος που περιγράφηκε παραπάνω αποτελεί έναν από αυτούς. Άλλοι είναι το DTD (Document Type Definition) που αναφέρθηκε σε προηγούμενη παράγραφο και ήταν ο πρώτος τρόπος επικύρωσης ενός XML εγγράφου, το RELAX-NG το οποίο είναι μια άλλη γλώσσα για τη δημιουργία XML σχημάτων (schema language) και επίσης αποτελεί διεθνές πρότυπο (ISO/IEC 19757-2) [14].

3.7. complexType και simpleType

Σε ένα XML Schema, συναντάμε τις περισσότερες φορές τα στοιχεία <complexType> και <simpleType>.

Το στοιχείο simpleType χρησιμοποιείται για να ορίσουμε τύπους που βασίζονται στους συνήθεις τύπους δεδομένων και να ορίσουμε περιορισμούς σε αυτούς. Για τον σκοπό αυτό εμφωλευμένα σ' ένα στοιχείο simpleType μπορούν να τοποθετηθούν στοιχεία όπως <restriction> για να περιορίσουμε ένα τύπο δεδομένων, <union> για να δημιουργήσουμε έναν νέο τύπο δεδομένων που θα βασίζεται σε δύο πρωταρχικούς – απλούς τύπους δεδομένων κτλ. Τα simpleType στοιχεία δεν επιτρέπεται να εμφωλεύουν άλλα στοιχεία, παρά μόνο χαρακτηριστικά.

Το στοιχείο complexType χρησιμοποιείται για να δηλώσουμε σύνθετους τύπους. Μπορούμε να έχουμε διαφορετικούς συνδυασμούς τύπων χρησιμοποιώντας διαφορετικά στοιχεία εμφωλευμένα σε ένα complexType στοιχείο.

Αν για παράδειγμα θέλουμε να αναπαραστήσουμε έναν τύπο για τον οποίον η σειρά εμφάνισης των στοιχείων του είναι αυστηρή, τότε χρησιμοποιούμε το στοιχείο sequence εμφωλευμένο στο complexType στοιχείο, και μέσα στο στοιχείο sequence εμφωλεύουμε αλλά στοιχεία κ.ο.κ. όπως στο παράδειγμα στην εικόνα 3.2.

```
<xs:element name="reserves" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="sid" type="xs:int" minOccurs="1"/>
      <xs:element name="bid" type="xs:int" minOccurs="1"/>
      <xs:element name="day" type="xs:date" minOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Εικόνα 3.2. complexType τύπος με αυστηρά καθορισμένη τη σειρά των στοιχείων του με τη χρήση του sequence

Στην εικόνα 3.2 ο τύπος είναι ενσωματωμένος (και εμφωλευμένος) στο στοιχείο reserves. Μια τέτοια δομή element – complexType – sequence μπορεί να χρησιμοποιηθεί για τον ορισμό πινάκων και UDT (User Defined Types) στο XML Schema.

3.8. minOccurs και maxOccurs

Τα χαρακτηριστικά minOccurs και maxOccurs χρησιμοποιούνται για να δηλώσουμε το πλήθος των εμφανίσεων ενός στοιχείου στο XML έγγραφο. Χρησιμοποιώντας το minOccurs μπορούμε να ορίσουμε τον ελάχιστο αριθμό εμφανίσεων ενός στοιχείου ενώ με το maxOccurs μπορούμε να ορίσουμε τον μέγιστο αριθμό εμφανίσεων ενός στοιχείου αντίστοιχα.

Αν χρησιμοποιήσουμε και τα δύο παράλληλα, τότε ορίζουμε ουσιαστικά ένα εύρος εμφανίσεων για το συγκεκριμένο στοιχείο. Το minOccurs μπορεί να πάρει την τιμή “0” το οποίο σημαίνει ότι το συγκεκριμένο στοιχείο μπορεί να παραλειφθεί. Αντίστοιχα το maxOccurs μπορεί να πάρει την τιμή “unbounded” το οποίο υποδεικνύει ότι το συγκεκριμένο στοιχείο μπορεί να εμφανιστεί άπειρες φορές στο XML έγγραφο.

3.9. Namespaces

Στην XML, όπως αναφέρθηκε σε προηγούμενες παραγράφους, τα ονόματα των στοιχείων και των χαρακτηριστικών τους δεν είναι αυστηρώς ορισμένα από την ίδια τη γλώσσα, όπως σε παλαιότερες, λιγότερο ευέλικτες γλώσσες σήμανσης

δεδομένων, σαν την HTML ή την SGML. Αυτή η ευελιξία που διαθέτει ο προγραμματιστής, να μπορεί δηλαδή να ορίσει κατά βούληση τα ονόματα των στοιχείων και των χαρακτηριστικών, εμπεριέχει τον κίνδυνο των διπλότυπων ονομάτων τα οποία δημιουργούν συγκρούσεις (collisions).

Οι πιθανότητες διπλοτυπίας αυξάνονται πολύ περισσότερο όταν δημιουργείται ένα XML έγγραφο συνδυάζοντας δύο ή περισσότερα διαφορετικά έγγραφα, όπως για παράδειγμα σε μια εφαρμογή που συλλέγει δεδομένα από διαφορετικές πηγές – εφαρμογές και τα συνδυάζει με έναν καθορισμένο τρόπο για να μπορεί να τα παρουσιάσει σαν ένα σύνολο.

Η λύση του προβλήματος διπλοτυπίας και συνεπώς της δημιουργίας συγκρούσεων μεταξύ των ονομάτων τους στην γλώσσα XML, βρίσκεται στο μηχανισμό των Namespaces. Μια ελληνική μετάφραση του όρου θα μπορούσε να είναι «ονόματα χώρου», υποδεικνύοντας την διακριτότητα των στοιχείων και των ονομάτων με βάση το «χώρο» που τους παρέχεται. Παρακάτω θα χρησιμοποιήσουμε τον όρο στην αγγλική γλώσσα, όπως είναι.

Ουσιαστικά, τα Namespaces είναι λεξιλόγια ονομάτων που υποδεικνύουν σε ποιο λεξιλόγιο ανήκει το κάθε ονοματισμένο στοιχείο ή χαρακτηριστικό που συναντάται σ' ένα XML έγγραφο. Στην πραγματικότητα, τα περισσότερα XML έγγραφα, περιέχουν στοιχεία ή και ολόκληρα τμήματα XML (XML fragments) από διαφορετικά XML έγγραφα τα οποία ακολουθούν προφανώς και διαφορετικά XML σχήματα.

Με τη χρήση των Namespaces αποφεύγονται οι συγκρούσεις ονομάτων μέσα στο έγγραφο και εξασφαλίζονται η εγκυρότητα του, ή αναγνωσιμότητα του από μια εφαρμογή καθώς και ότι το έγγραφο είναι καλώς ορισμένο.

Ένα Namespace ορίζεται στο αντίστοιχο XML έγγραφο ή σχήμα χρησιμοποιώντας το δεσμευμένο χαρακτηριστικό xmlns, ως εξής:

xmlns = "URI_Αναφορά"

Σε περίπτωση που χρησιμοποιούνται στοιχεία από πολλά διαφορετικά Namespaces τότε μπορούμε για κάθε διαφορετικό Namespace να ορίσουμε ένα αναγνωριστικό πρόθεμα (prefix), στη δήλωση κάθε σχήματος, για το οποίο θα μιλήσουμε παρακάτω. Για τη χρήση προθέματος θα πρέπει να τροποποιήσουμε την δήλωση του Namespace, η οποία θα πάρει την εξής μορφή:

xmlns**:πρόθεμα** = “URI_Αναφορά”

3.9.1. Namespaces και URI

Με δεδομένο ότι ακόμα και η ονομασία των Namespaces , βρίσκεται στην ευχέρεια του προγραμματιστή, η αποφυγή της σύγκρουσης ονομάτων τους είναι αναπόφευκτη. Επομένως, η ανάγκη ενός μηχανισμού που θα διακρίνει δύο ή και περισσότερα Namespaces μεταξύ τους είναι δεδομένη.

Ο μηχανισμός αυτός είναι ιδιαίτερα γνωστός και δεν έχει άμεση σχέση με την XML αυτή καθεαυτή. Η αποφυγή τέτοιου είδους συγκρούσεων εξασφαλίζεται από τον γνωστό μηχανισμό του Παγκόσμιου Ιστού , το URI(Unified Resource Identifier).

Η δικλείδα ασφαλείας που παρέχει το URI όσον αφορά την μοναδικότητα των διευθύνσεων στον παγκόσμιο Ιστό, είναι αυτή που χρησιμοποιεί ο μηχανισμός των Namespaces για να εξασφαλίσει τη λειτουργικότητα του.

Ωστόσο σύμφωνα με το W3C, η επιλογή των URI θέλει ιδιαίτερη προσοχή καθώς:

- Οι αναφορές URI που συνδέονται με ένα Namespace (binding) είναι case-sensitive, το οποίο σημαίνει ότι το Namespace που συνδέεται με την URI αναφορά “www.example.org/myNS” είναι διαφορετικό από το www.example.org/mynS
- Ορισμένοι χαρακτήρες αποφυγής (escape characters) μπορεί να αλλάξουν την URI αναφορά εάν αποαναφερθούν (dereferenced) όπως για παράδειγμα ο χαρακτήρας ‘%’, και καλό θα είναι να αποφεύγονται [15].

3.9.2. Το πρόθεμα (Namespace prefix)

Όπως ήδη αναφέρθηκε στις προηγούμενες παραγράφους, ένα XML έγγραφο μπορεί να δημιουργείται ως συνδυασμός δύο η περισσότερων XML εγγράφων, τμημάτων, στοιχείων ή χαρακτηριστικών τους.

Ο συνδυασμός αυτός μπορεί να επιφέρει συγκρούσεις ονομάτων καθώς στο ίδιο XML έγγραφο συμπεριλαμβάνονται στοιχεία από διαφορετικά έγγραφα με την ίδια ονομασία. Έτσι χρησιμοποιούμε τα XML Namespaces για να διακρίνουμε την προέλευση αυτών των στοιχείων ή και τους «γραμματικούς κανόνες» που αυτά ακολουθούν (schema).

Ωστόσο, μέσα στο ίδιο το XML έγγραφο, το Namespace που ανήκει κάθε στοιχείο δεν είναι διακριτό. Για το λόγο αυτό, χρησιμοποιούμε ένα δείκτη που προσδιορίζει ρητά την προέλευση και τη σύνδεση ενός στοιχείου με το αντίστοιχο Namespace του. Ο δείκτης αυτός είναι το πρόθεμα(prefix) το οποίο προηγείται του ονόματος σήμανσης του στοιχείου και γράφεται με την εξής μορφή:

```
<πρόθεμα:όνομα_σήμανσης></πρόθεμα:όνομα_σήμανσης>
```

Το πρόθεμα υποδεικνύει ότι το στοιχείο που ακολουθεί ανήκει σε εκείνο το Namespace που αναγνωρίζεται από το συγκεκριμένο πρόθεμα. Με αυτό τον τρόπο, η προέλευση του κάθε στοιχείου μέσα στο έγγραφο γίνεται πιο διακριτή για τον αναγνώστη αλλά και για τις εφαρμογές που θα το επεξεργαστούν χωρίς τα προβλήματα σύγκρουσης ονομάτων που αναλύθηκαν παραπάνω.

3.10. Το ιεραρχικό μοντέλο δεδομένων της XML

Το μοντέλο δεδομένων της XML, ονομάζεται ιεραρχικό ή δενδρικό λόγω της δομής των XML εγγράφων στην οποία τα στοιχεία (elements) με τον τρόπο που είναι δομημένα, εμφωλευμένα το ένα μέσα στο άλλο, εκφράζουν μεταξύ τους μια σχέση γονέα – παιδιού (parent-child), μια ιεραρχία.

Είναι ευνόητο ότι στην έννοια «μοντέλο» συγκαταλέγονται όλα τα βασικά δομικά στοιχεία της XML που αναφέρθηκαν παραπάνω. Τα στοιχεία αυτά είναι που συνθέτουν και σχηματίζουν την δενδρική – ιεραρχική δομή ενός XML εγγράφου.

3.11. Περίληψη Κεφαλαίου

Στο κεφάλαιο αυτό αναφερθήκαμε στην XML (eXtensible Markup Language) μια self-described γλώσσα σήμανσης, εξέλιξη ενός υποσυνόλου της παλαιότερης SGML. Η XML αποτελεί διεθνές πρότυπο για την ανταλλαγή δεδομένων στο διαδίκτυο και βρίσκει εφαρμογή κυρίως σε διαδικτυακές εφαρμογές και web services.

Η δομή κάθε XML έγγραφου αποτελείται από κάποια βασικά δομικά στοιχεία όπως ο πρόλογος, το στοιχείο-ρίζα, τα στοιχεία ,οι ετικέτες τους και τα χαρακτηριστικά τους. Ο σωστός συνδυασμός των παραπάνω δημιουργεί ένα καλώς ορισμένο (well formed) XML έγγραφο το οποίο χαρακτηρίζεται από «γραμματικούς κανόνες» που διατυπώνονται στο XML Schema. Αν το έγγραφο τηρεί αυτούς τους κανόνες τότε είναι ένα «έγκυρο XML έγγραφο».

Χρησιμοποιώντας, κάποια ειδικά στοιχεία και χαρακτηριστικά όπως τα simpleType, complexType, sequence, union, minOccurs, maxOccurs μπορούμε να ορίσουμε πιο εξειδικευμένους περιορισμούς στο XML Schema, εξασφαλίζοντας την αναπαράσταση των δεδομένων στο XML έγγραφο ακριβώς με την μορφή και τους περιορισμούς που θέλουμε.

Σε περίπτωση που στο XML Schema χρησιμοποιούμε στοιχεία από διαφορετικά σχήματα τότε είναι απαραίτητη η δήλωση μπροστά από το στοιχείο, ενός προθέματος (prefix) το οποίο ορίζει την προέλευση του συγκεκριμένου στοιχείου. Έτσι αποφεύγονται συγκρούσεις ονομάτων μέσα σ' ένα XML Schema.

Το μοντέλο δεδομένων της XML, ονομάζεται ιεραρχικό ή δενδρικό λόγω της δομής των XML εγγράφων. Η εμφωλευμένη αυτή δόμηση των στοιχείων εκφράζει σχέσεις γονέα-παιδιού, αναπαριστά δηλαδή, άμεσα συσχετίσεις μεταξύ των στοιχείων της.

4. Συστήματα Διαχείρισης Βάσεων Δεδομένων

Σε αυτό το κεφάλαιο θα μιλήσουμε για τα Συστήματα Διαχείρισης Βάσεων δεδομένων (ΣΔΒΔ). Σημειώνεται ότι στο συγκεκριμένο κεφάλαιο δεν θα αναλύσουμε σε βάθος το θέμα αλλά θα περιγράψουμε τα βασικά στοιχεία και θα συγκρίνουμε συνοπτικά δύο μοντέλα, το σχεσιακό και το αντικειμενοσχεσιακό. Για περισσότερες πληροφορίες, ο αναγνώστης παραπέμπεται στην μελέτη της βιβλιογραφίας.

4.1. Εισαγωγή

Μετά την ανάπτυξη των πρώτων υπολογιστικών συστημάτων και καθώς αυτά εξελίσσονταν και παράλληλα διευρύνονταν οι δυνατότητες, οι λειτουργίες και ο όγκος πληροφοριών που αυτά επεξεργάζονταν, γεννήθηκε η ανάγκη της αποθήκευσης αυτών των πληροφοριών με τρόπο τέτοιο, που θα εξασφάλιζε την δυνατότητα μελλοντικής αναφοράς και επανεπεξεργασίας τους.

4.2. Ιστορική αναδρομή

Μια από τις σημαντικότερες προσπάθειες της ικανοποίησης αυτής της ανάγκης έγινε στις αρχές της δεκαετίας του 1960 από τον Charles Bachman της General Electrics, ο οποίος κατασκεύασε το πρώτο Σύστημα Διαχείρισης Βάσεων Δεδομένων, το λεγόμενο IDS (Integrated Data Store). Ήταν ένα γενικής χρήσης ΣΔΒΔ το οποίο βασίζονταν στο δικτυωτό μοντέλο δεδομένων όπως αυτό ορίστηκε από το Συνέδριο Γλωσσών των Συστημάτων Δεδομένων (CODASYL).

Η ανάπτυξη του πρώτου DBMS, έδωσε προοπτική στον τομέα της έρευνας που αφορά τις δομές αποθήκευσης δεδομένων και έτσι πολλές εταιρείες άρχισαν να προσανατολίζονται προς την δημιουργία δικών τους ΣΔΒΔ. Έτσι στα τέλη της δεκαετίας του 1960 η IBM ανέπτυξε το σύστημα IMS (Information Management System) το οποίο ακολούθησε διαφορετικό μοντέλο αναπαράστασης της πληροφορίας από το IDS, το ιεραρχικό μοντέλο δεδομένων [16].

Την ίδια περίπου περίοδο η IBM σε συνεργασία με την American Airlines αναπτύσσει το σύστημα SABRE (Semi-Automatic Business Research Environment), ως εξέλιξη του συστήματος κρατήσεων Reservisor [17], το οποίο

υποστήριζε μεταξύ άλλων την καινοτομία της δυνατότητας πρόσβασης πολλαπλών χρηστών (τότε ταξιδιωτικών πρακτορείων) μέσω ενός δικτύου. Το έργο ξεκίνησε το 1953 και ολοκληρώθηκε με την κατασκευή του δικτύου το 1964 [18].

4.3. Η εμφάνιση του Σχεσιακού μοντέλου δεδομένων

Την αποσπασματική ανάπτυξη συστημάτων αποθήκευσης και διαχείρισης δεδομένων, με διάφορα μοντέλα δεδομένων, για συγκεκριμένους σκοπούς διαδέχτηκε η εμφάνιση ενός μοντέλου δεδομένων που θα καθόριζε σημαντικά τις εξελίξεις στο πεδίο των βάσεων δεδομένων μέχρι σήμερα.

Ο λόγος για το σχεσιακό μοντέλο δεδομένων το οποίο εισήχθη ως όρος για πρώτη φορά το 1970, από τον Edgar F. Codd της IBM, στην εργασία του με τίτλο «A Relational Model of Data for Large Shared Data Banks» [31]. Την εμφάνιση του σχεσιακού μοντέλου δεδομένων ακολούθησε μια έκρηξη στην ανάπτυξη σχεσιακών ΣΔΒΔ, με αποτέλεσμα το μοντέλο να υιοθετηθεί από πολλές εταιρείες του κλάδου και να ενισχυθεί η ακαδημαϊκή και ιδιωτική έρευνα για την περαιτέρω ανάπτυξη του [16].

Η σημασία του και τα οφέλη που γεννούσε για τις επιχειρήσεις ήταν τεράστια, με αποτέλεσμα όλο και περισσότερες επιχειρήσεις να χρησιμοποιούν σχεσιακά ΣΔΒΔ για την διαχείριση των δεδομένων τους.

Το σχεσιακό μοντέλο εδραιώθηκε στα μέσα της δεκαετίας του 1980 [16] ενώ παράλληλα αναπτύχθηκαν νέες, σχετικές με το σχεσιακό μοντέλο, τεχνολογίες όπως οι γλώσσες αιτημάτων (π.χ.SQL), οι οποίες επιτρέπουν την ανάκτηση, από τη βάση δεδομένων, πληροφοριών που πληρούν συγκεκριμένα κριτήρια, καθορισμένα από τον χρήστη.

Σήμερα έχουν αναπτυχθεί πολλά RDBMS (Relational DBMS), δηλαδή DBMS που υποστηρίζουν το σχεσιακό μοντέλο δεδομένων, από διάφορες εταιρείες του χώρου, τα οποία είναι εξοπλισμένα με διάφορες λειτουργίες και ολοκληρωμένα προγράμματα διαχείρισης των δεδομένων τους. Μερικά από τα πιο γνωστά είναι η DB2 της εταιρείας IBM, η Oracle, η PostgreSQL.

4.3.1. Βασική περιγραφή του σχεσιακού μοντέλου

Το σχεσιακό μοντέλο δεδομένων περιγράφει τα δεδομένα που αποθηκεύονται με την μορφή των πινάκων (tables). Κάθε πίνακας είναι μια «οντότητα» η οποία συνήθως έχει ένα η περισσότερα χαρακτηριστικά «γνωρίσματα». Οντότητα είναι ένα αντικείμενο του πραγματικού κόσμου το οποίο έχει διακριτή ύπαρξη σε σχέση με τα υπόλοιπα αντικείμενα [16].

Κάθε πίνακας, πρέπει να έχει τουλάχιστον ένα κύριο κλειδί, δηλαδή ένα χαρακτηριστικό γνώρισμα που καθιστά την οντότητα που περιγράφεται μοναδική. Οι οντότητες μπορούν να συσχετίζονται μεταξύ τους. Η διασύνδεση μεταξύ δύο ή περισσότερων οντοτήτων ονομάζεται συσχέτιση (relation) [16].

Σε έναν πίνακα μπορεί να συναντήσουμε και ένα ξένο κλειδί δηλαδή μια αναφορά σ' ένα κύριο κλειδί άλλου πίνακα. Ένας πίνακας συσχέτισης ή απλά μια συσχέτιση, μπορεί να έχει και ένα κύριο και ένα ξένο κλειδί, τα οποία είναι τα αντίστοιχα κύρια κλειδιά στους πίνακες που συσχετίζει.

Το σχεσιακό μοντέλο δεδομένων, όπως υιοθετήθηκε και εξελίχθηκε από τις εταιρείες του χώρου, έχει μια σειρά από πλεονεκτήματα έναντι των παλαιότερων μοντέλων δεδομένων. Μερικά από αυτά, όπως έχουν παρατηρηθεί κατά [19] είναι:

- η ευελιξία πρόσβασης
- η λογική και φυσική ανεξαρτησία των δεδομένων
- η ακεραιότητα των δεδομένων
- ο μειωμένος και ελεγχόμενος πλεονασμός των δεδομένων
- η αυξημένη παραγωγικότητα των προγραμματιστών

4.4. Το αντικειμενοσχεσιακό μοντέλο δεδομένων

Παράλληλα με την τεχνολογική έκρηξη που έφεραν οι αντικειμενοστραφείς γλώσσες προγραμματισμού, όπως είναι λογικό, αυξήθηκαν και οι ανάγκες για υποστήριξη των εφαρμογών που σχετίζονταν με τις βάσεις δεδομένων. Έτσι, οι εταιρείες παραγωγής ΣΔΒΔ στην προσπάθεια τους να συμβαδίσουν με τις

απαιτήσεις, ανέπτυξαν αντίστοιχα DBMS , τα οποία μπορούσαν να υποστηρίξουν μια ευρεία γκάμα εφαρμογών.

Έτσι γεννήθηκαν τα Συστήματα Διαχείρισης Βάσεων Δεδομένων Αντικειμένων. Σε πρώτη φάση αναπτύχθηκαν ΣΔΒΔ που υποστήριζαν πλήρως την αντικειμενοστρέφεια που χαρακτήριζε τις σύγχρονες εφαρμογές. Τα συγκεκριμένα ΣΔΒΔ ονομάζονται OODBMS , δηλαδή Object-Oriented DBMS.

4.4.1. Τί είναι το αντικειμενοσχεσιακό μοντέλο δεδομένων;

Το αντικειμενοσχεσιακό μοντέλο δεδομένων είναι ουσιαστικά η επέκταση του σχεσιακού μοντέλου, έτσι ώστε το τελευταίο να έχει αντικειμενοστρεφή χαρακτηριστικά και να μπορεί να υποστηρίζει μια ευρεία τάξη εφαρμογών.

Χαρακτηρίζεται ως «γέφυρα» μεταξύ των σχεσιακών και αντικειμενοσχεσιακών ΣΔΒΔ [16], καθώς έχει χαρακτηριστικά που συναντούνται και στα δύο μοντέλα και υποστηρίζει παρόμοιες λειτουργίες. Αποτελεί δηλαδή ένα είδος μεταβατικού μοντέλου από το ένα μοντέλο στο άλλο.

4.5. Διαφορές Σχεσιακού – Αντικειμενοσχεσιακού μοντέλου

Σε αντίθεση με το σχεσιακό μοντέλο, το αντικειμενοσχεσιακό μοντέλο δεδομένων δίνει τη δυνατότητα στον χρήστη να ορίσει αφηρημένους τύπους (ADT – Abstract Data Types) καθώς και δικούς του τύπους δεδομένων (UDT – User Defined Types) οι οποίοι βασίζονται στους ήδη ενσωματωμένους τύπους δεδομένων.

Επίσης, το αντικειμενοσχεσιακό μοντέλο υποστηρίζει μεταξύ άλλων πολλά από τα χαρακτηριστικά που συναντάμε στις αντικειμενοστρεφείς γλώσσες προγραμματισμού, όπως τα αντικείμενα (objects), τις κλάσεις (classes) και την κληρονομικότητα (inheritance). Ένα άλλο σημαντικό χαρακτηριστικό που υποστηρίζεται και έχει άμεση σχέση με τους ADT τύπους είναι η ενθυλάκωση (encapsulation). Ενθυλάκωση είναι η απόκρυψη των λεπτομερειών των τύπων ADT και των τρόπων λειτουργίας των μεθόδων τους από το εκάστοτε ΣΔΒΔ [16].

Μια ακόμη ουσιαστική διαφορά μεταξύ των δύο μοντέλων είναι το Object ID (OID). Το OID είναι ένας μοναδικός αναγνωριστικός αριθμός ενός αντικειμένου και δίνεται αυτόματα από το ΣΔΒΔ, σε κάθε αντικείμενο που καταχωρείται στη βάση. Σε αντίθεση με τα κύρια και τα ξένα κλειδιά τα οποία εξασφαλίζουν την μοναδικότητα στα πλαίσια ενός πίνακα, το OID είναι μοναδικό για όλο το ΣΔΒΔ και ισχύει για όλη τη διάρκεια ζωής του αντικειμένου. Με αυτόν τον τρόπο η αναφορική ακεραιότητα των δεδομένων είναι εξασφαλισμένη [20]. Είναι επίσης δυνατή η ανάκτηση μιας αναφοράς σ' ένα αντικείμενο της βάσης.

Αυτές οι καινοτομίες, επεκτείνουν σημαντικά τις δυνατότητες ενός DBMS καθιστώντας το ελκυστικότερη λύση για τις εταιρείες στον τομέα της αποθήκευσης και διαχείρισης πληροφοριών.

4.6. Περίληψη Κεφαλαίου

Με την εξέλιξη των πρώτων υπολογιστών γεννήθηκε η ανάγκη της αποθήκευσης της πληροφορίας με τρόπο που θα διευκόλυνε τη μελλοντική αναφορά αλλά και επεξεργασία της. Τα πρώτα ΣΔΒΔ αναπτύχθηκαν στις αρχές της δεκαετίας του 1960 από διάφορες εταιρείες για διαφορετικούς σκοπούς. Το 1970 εισήχθη για πρώτη φορά η έννοια του σχεσιακού μοντέλου δεδομένων, γεγονός που επιτάχυνε άρδην τις εξελίξεις για την κατασκευή πιο ολοκληρωμένων ΣΔΒΔ.

Το σχεσιακό μοντέλο συνέχισε να εξελίσσεται από πολλές εταιρείες του χώρου, η καθεμία από τις οποίες ανέπτυσσε τα δικά της σχεσιακά ΣΔΒΔ. Οι επιχειρήσεις, αντιλαμβανόμενες τα οφέλη που γεννούσε το νέο μοντέλο δεδομένων, άρχισαν να το υιοθετούν με αποτέλεσμα αυτό να εδραιωθεί στην αγορά στα μέσα της δεκαετίας του 1980.

Οι εξελίξεις στον τομέα των γλωσσών προγραμματισμού, γέννησαν νέες ανάγκες και έτσι οι εταιρείες κατασκευής και εμπορίας ΣΔΒΔ ανέπτυξαν ΣΔΒΔ που να συμβαδίζουν με τις προγραμματιστικές ανάγκες των εφαρμογών. Έτσι γεννήθηκαν τα αντικειμενοσχεσιακά και τα αντικειμενοστρεφή συστήματα (Object Oriented DBMS).

Τα πλεονεκτήματα των σχεσιακών ΣΔΒΔ , όπως την ευελιξία που προσφέρουν σε επίπεδο πρόσβασης, την ακεραιότητα των δεδομένων, την αυξημένη παραγωγικότητα της εργασίας των προγραμματιστών κ.α., διασφαλίζουν και επεκτείνουν τόσο τα αντικειμενοσχεσιακά όσο και τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων.

Τα αντικειμενοσχεσιακά ΣΔΒΔ υποστηρίζουν νέα χαρακτηριστικά όπως αντικείμενα (objects), κλάσεις (classes), κληρονομικότητα (inheritance), ενθυλάκωση (encapsulation).

Επίσης, παράλληλα με τον μηχανισμό των κύριων και ξένων κλειδιών υποστηρίζουν και το OID, δηλαδή ένα μοναδικό (για όλο το ΣΔΒΔ) αναγνωριστικό αριθμό για κάθε ένα από τα αντικείμενα που καταχωρούνται στη βάση δεδομένων.

Η σύγκριση των δύο συστημάτων που έγινε σε αυτό το κεφάλαιο, ήταν σκόπιμα συνοπτική. Η απόφαση της επιλογής του ενός ή του άλλου μοντέλου δεδομένων είναι στο χέρι του προγραμματιστή και εξαρτάται άμεσα από τις εκάστοτε ανάγκες. Στη γενική βιβλιογραφία μπορείτε να βρείτε, αναλυτικότερες και πιο στοχευμένες συγκρίσεις των δύο συστημάτων όπως για παράδειγμα η [21].

5. Αλγόριθμοι μετατροπής βάσεων δεδομένων (Mapping algorithms)

Στο προηγούμενο κεφάλαιο κάναμε μια συνοπτική περιγραφή και σύγκριση μεταξύ δύο διαφορετικών μοντέλων δεδομένων, του σχεσιακού και του αντικειμενοσχεσιακού. Έχοντας ως βασικό υπόβαθρο τα παραπάνω θα προσπαθήσουμε να εστιάσουμε στους αλγόριθμους μετατροπής βάσεων δεδομένων σε άλλες αντίστοιχες μορφές. Στα πλαίσια αυτής της εργασίας εξετάζεται η μετατροπή μια σχεσιακής και μιας αντικειμενοσχεσιακής βάσης δεδομένων σε μια αντίστοιχη αναπαράσταση σε μορφή XML.

5.1. Εισαγωγή

Από τις πρώτες ακόμα επιτυχημένες προσπάθειες κατασκευής συστημάτων αποθήκευσης και πιο συγκεκριμένα συστημάτων διαχείρισης βάσεων δεδομένων (βλέπε κεφάλαιο 4), είχε προκύψει ένα σημαντικό πρόβλημα. Αυτό συνοψίζεται στην εξής ερώτηση:

Πώς θα γίνει δυνατό, τα αποθηκευμένα δεδομένα μιας μορφής, να μετατραπούν σε δεδομένα άλλης μορφής προκειμένου να μπορούν να αποθηκευτούν και να διαχειριστούν από τα νέα συστήματα;

Η παραπάνω ερώτηση αναφέρεται στο πολύ γνωστό πρόβλημα του backward compatibility, δηλαδή της «συμβατότητας προς τα πίσω». Το πρόβλημα αυτό, οξύνονταν, καθώς κάθε εταιρεία ανέπτυσσε το δικό της ΣΔΒΔ το οποίο προφανώς είχε διαφορετικές προδιαγραφές από αυτά των άλλων εταιρειών ακόμα και αν αυτά χρησιμοποιούσαν το ίδιο μοντέλο δεδομένων.

Έτσι γεννήθηκε η ανάγκη της ύπαρξης διαδικασιών και λειτουργιών που επιτρέπουν την μετατροπή των δεδομένων σε μορφές πέραν αυτής που υποστηρίζει το εκάστοτε ΣΔΒΔ έτσι ώστε να μπορούν να χρησιμοποιηθούν από ΣΔΒΔ που ενσωματώνουν νεότερες τεχνολογίες. Οι διαδικασίες αυτές ονομάζονται αλγόριθμοι μετατροπής (mapping algorithms) τους οποίους θα αναλύσουμε παρακάτω.

5.2. Τί είναι ένας αλγόριθμος μετατροπής;

Όπως αναφέρθηκε στην εισαγωγή αυτού του κεφαλαίου, ένας αλγόριθμος μετατροπής δεδομένων είναι, στην ουσία, μια διαδικασία με την οποία τα δεδομένα που είναι αποθηκευμένα σε ένα σύστημα υπό μια μορφή, μετατρέπονται σε μια άλλη μορφή υπό κάποιους συγκεκριμένους περιορισμούς έτσι ώστε να μπορούν να αξιοποιηθούν από άλλα, συνήθως τεχνολογικά πιο εξελιγμένα, συστήματα.

Τα σύγχρονα ΣΔΒΔ υποστηρίζουν μια σειρά από τέτοιους αλγορίθμους. Οι αλγόριθμοι είναι ενσωματωμένοι στο ΣΔΒΔ και μας δίνουν τη δυνατότητα να εξάγουμε τα δεδομένα της βάσης σε διάφορες μορφές, όπως για παράδειγμα CSV(Comma Separated Values), PDF(Portable Document Format), XML (eXtensible Markup Language) κ.α.

Στα πλαίσια αυτής της εργασίας, βέβαια, δεν εστιάζουμε στους αλγόριθμους που μετατρέπουν τα δεδομένα μιας βάσης σε μορφή που διευκολύνει αποκλειστικά την ανάγνωση από έναν άνθρωπο, όπως είναι οι παραπάνω αλγόριθμοι με εξαίρεση την SQL και την XML.

Σημείωση: *Η XML έχει το πλεονέκτημα ότι λόγω της ίδιας της δομής ενός XML εγγράφου είναι ευανάγνωστη τόσο από τον υπολογιστή όσο και από έναν άνθρωπο.*

Εστιάζουμε στους αλγόριθμους εκείνους οι οποίοι πραγματεύονται την απευθείας μετατροπή μιας βάσης σε μια δομή δεδομένων με άλλη μορφή. Συνεπώς, θα εστιάσουμε στους αλγόριθμους μετατροπής σχεσιακών και αντικειμενοσχεσιακών βάσεων δεδομένων σε αντίστοιχες XML μορφής όπως είναι και το θέμα της εργασίας.

5.3. Αλγόριθμοι μετατροπής δεδομένων για το σχεσιακό μοντέλο δεδομένων

Οι αλγόριθμοι μετατροπής δεδομένων μιας σχεσιακής βάσης δεδομένων σε αντίστοιχη XML μορφής, σύμφωνα με την έρευνα που έγινε στα πλαίσια αυτής της εργασίας, βρίσκονται ακόμα κατά την πλειοψηφία τους σε πρώιμο, θεωρητικό στάδιο.

Στην διεθνή βιβλιογραφία, προτείνονται κάποιοι αλγόριθμοι σε θεωρητικό επίπεδο, χωρίς όμως στην πλειοψηφία τους να έχουν ενσωματωθεί και υλοποιηθεί σε εφαρμογές. Συνεπώς, δεν είναι δυνατή μια σύγκριση και αξιολόγηση απόδοσης τους (performance evaluation).

Κατά την μετατροπή των δεδομένων, είναι γενικά παραδεκτή και αποδεκτή, μια σχετική απώλεια της πληροφορίας από την αρχική σχεσιακή στην τελική XML μορφή. Οι αλγόριθμοι που αναφέρονται παρακάτω αφορούν και εστιάζουν στη διαφύλαξη διάφορων χαρακτηριστικών μιας σχεσιακής βάσης δεδομένων όπως την διατήρηση και ακριβή απόδοση των περιορισμών (constraints) και την αναφορική ακεραιότητα των δεδομένων.

Όπως είναι λογικό, οι αλγόριθμοι που προτείνονται αφορούν την μετατροπή του σχήματος μιας βάσης σε αντίστοιχο XML Schema, καθώς και στις δύο περιπτώσεις αυτά καθορίζουν τη δομή, τους περιορισμούς και την τελική μορφή των δεδομένων.

5.3.1. ConvRel

Ο αλγόριθμος ConvRel όπως προτείνεται από τους (AC Duta et al.) στο [13] είναι ένας αλγόριθμος ο οποίος ασχολείται με το πρόβλημα της αποδοτικής μετατροπής σχεσιακών σχημάτων σε εμφωλευμένα XML σχήματα.

Ο προτεινόμενος αλγόριθμος μετατροπής, εστιάζει στην διατήρηση των περιορισμών δομής (structural constraints), αναλύοντας τη σχέση κάθε συσχέτισης με τις υπόλοιπες.

Ταυτόχρονα προτείνονται διάφορες υποψήφιες XML δομές για κάθε διαφορετικού τύπου συσχέτιση ανάλογα με τους περιορισμούς δομής της. Από αυτούς επιλέγεται ο καλύτερος, με κριτήριο τη συμπαγή XML δομή και το μικρότερο μέγεθος των XML παραγόμενων δεδομένων.

Αναλυτικότερα για τον αλγόριθμο ConvRel και τα βήματα του κατά την μετατροπή, μπορείτε να βρείτε στην βιβλιογραφία [13].

5.3.2. Holistic Constraint-Preserving Transformation Algorithm

Στην εργασία των (R Zhou et al.) προτείνεται ένας αλγόριθμος ο οποίος, όπως περιγράφεται στο [22], μετατρέπει ένα σχεσιακό σχήμα στον αντίστοιχο XML Schema, εστιάζοντας στη διαφύλαξη των περιορισμών ακεραιότητας (integrity constraints), όπως τα κύρια και τα ξένα κλειδιά, τα null και not null στοιχεία καθώς και άλλα μοναδικά χαρακτηριστικά.

Ο αλγόριθμος αυτός ουσιαστικά αποτελεί μια εξέλιξη του αλγορίθμου που περιγράφεται στο [23]. Ο πρώτος αλγόριθμος που αναπτύχθηκε, υλοποιούσε την μετατροπή του σχεσιακού σχήματος σε XML schema σε δύο βήματα. Στο πρώτο βήμα, δημιουργούσε ένα ενδιάμεσο XML schema που αναπαριστούσε τις διαφορετικές σχέσεις, το οποίο χρησιμοποιούσε για να καταλήξει στο δεύτερο βήμα όπου δημιουργούνταν το τελικό XML Schema και παράλληλα ένας IND γράφος ή γράφος αναφορών από το σχεσιακό σχήμα, ο οποίος εξέφραζε τις συσχετίσεις αναφοράς μεταξύ των σχέσεων.

Στην νέα ολιστική προσέγγιση, που προτείνεται, ο αλγόριθμος όχι μόνο διατηρεί τους περιορισμούς ακεραιότητας, όπως άλλωστε έκανε και στην αρχική του μορφή, αλλά υλοποιεί την μετατροπή του σχεσιακού σχήματος απευθείας σε XML Schema χωρίς να χρειάζεται να δημιουργηθούν ενδιάμεσα σχήματα ή γράφοι. Το κλειδί σε αυτή την επιτυχία είναι η κατηγοριοποίηση που προτείνεται για τις σχέσεις η οποία περιγράφεται λεπτομερώς στην εν λόγω εργασία.

Τέλος, προτείνεται ένα σετ κανόνων για την μετατροπή ενός σχεσιακού σχήματος σε αντίστοιχο XML Schema, με έμφαση, όπως αναφέρθηκε και παραπάνω, στην

διατήρηση των περιορισμών ακεραιότητας (Integrity constraints). Στα [22], [23] αναλύονται διεξοδικά οι αλγόριθμοι και οι τρόποι λειτουργίας τους.

5.3.3. Άλλες προσεγγίσεις

Στο πεδίο των αλγορίθμων μετατροπής δεδομένων (Mapping algorithms) μιας σχεσιακής βάσης σε αντίστοιχη XML μορφής έχουν προταθεί και άλλες προσεγγίσεις οι οποίες διακρίνονται από διαφορετική λογική.

Αλγόριθμος με χρήση SQL-based γλώσσας

Για παράδειγμα, στο [24] προτείνονται διαφορετικοί τρόποι υλοποίησης ενός αλγόριθμου μετατροπής δεδομένων ο οποίος βασίζεται σε μια SQL-based γλώσσα που αναλύεται στην εν λόγω εργασία. Αναλύονται ξεχωριστά οι διαφορετικές προσεγγίσεις και γίνεται μια ειδικότερη σύγκριση τους (benchmarking) .

Reverse-Engineering αλγόριθμος

Στο [25] προτείνεται ένας reverse-engineering αλγόριθμος ο οποίος ασχολείται πιο πρακτικά με το ζήτημα της μετατροπής των δεδομένων μιας σχεσιακής βάσης. Πιο συγκεκριμένα, η εργασία ασχολείται με τις σχεσιακές βάσεις δεδομένων που σχεδιάστηκαν με την μεθοδολογία λογικού σχεδιασμού και δεν βασίζονται σε κάποιο παραδοσιακό εμπορικό DBMS. Συνεπώς, δεν υπάρχουν λειτουργίες που να παράγουν τα μεταδεδομένα της βάσης δίνοντας μας πληροφορίες για τη δομή της βάσης, τους περιορισμούς κτλ.

Σε γενικές γραμμές, ο αλγόριθμος αυτός, εξάγει το ER μοντέλο (Entity – Relationship model) από την βάση δεδομένων και ύστερα το χρησιμοποιεί για να παράγει το αντίστοιχο XML Schema.

SilkRoute

Μια ακόμη προσέγγιση, είναι το SilkRoute [26]. Το SilkRoute αποτελεί ένα ενδιάμεσο σύστημα για την δημιουργία XML δεδομένων από σχεσιακά συστήματα βάσεων δεδομένων. Η μετατροπή γίνεται σε δύο βήματα [25].

Στο πρώτο βήμα, ο αλγόριθμος χρησιμοποιεί μια γλώσσα ερωτημάτων (query language) την RXL (Relational to XML Transformation Language) για να

δημιουργήσει μια εικονική XML όψη τους (Virtual XML View). Στο δεύτερο βήμα δημιουργείται ένα ερώτημα προς την XML όψη (π.χ. XML-QL) το οποίο μεταβιβάζεται στον «query composer». Ο query composer, με βάση αυτό το ερώτημα δημιουργεί ένα RXL ερώτημα το οποίο με τη σειρά του μεταφράζεται σε ένα η περισσότερα SQL ερωτήματα.

Ο «XML generator» λαμβάνει τα αποτελέσματα των SQL ερωτημάτων τα οποία επεξεργάζεται για να δημιουργήσει το XML αρχείο, προσθέτοντας τα κατάλληλα δομικά στοιχεία(ετικέτες, χαρακτηριστικά, εμφωλευμένη δομή κτλ).

5.4. Αλγόριθμοι μετατροπής δεδομένων για το αντικειμενοσχεσιακό μοντέλο δεδομένων

Από την έρευνα που έγινε στα πλαίσια αυτής της εργασίας, δεν παρατηρήθηκε ιδιαίτερη ανάπτυξη, ακόμα και σε θεωρητικό επίπεδο στον τομέα των αλγορίθμων μετατροπής δεδομένων μιας αντικειμενοσχεσιακής βάσης σε αντίστοιχη XML μορφής.

Αυτή η τάση ,που παρατηρείται στα πλαίσια της έρευνας μας, ίσως θα μπορούσε να δικαιολογηθεί από το γεγονός ότι ακόμα και στο σχεσιακό μοντέλο δεν έχει καθιερωθεί, έστω και ανεπίσημα, ένας αλγόριθμος μετατροπής δεδομένων σε XML μορφή και οι περισσότεροι έχουν μείνει μέχρι στιγμής, σε θεωρητικό στάδιο. Παρακάτω θα αναφερθούμε συνοπτικά τον αλγόριθμο XPERANTO [27].

5.4.1. XPERANTO

Το XPERANTO ειδικεύεται στην εξαγωγή των οντοτήτων, συσχετίσεων και αντικειμένων ορισμένων από τον χρήστη, δηλαδή βρίσκει εφαρμογή κυρίως στο αντικειμενοσχεσιακό μοντέλο δεδομένων [25]. Στόχος του, να εξυπηρετεί ως ενδιάμεσο συστατικό (middleware component) τους χρήστες που θέλουν να έχουν πρόσβαση σε μια XML αναπαράσταση της αντικειμενοσχεσιακής βάσης, χωρίς να χρειάζεται να ασχολούνται με το αντικειμενοσχεσιακό μοντέλο αυτό καθεαυτό [27].

Παρέχει μια XML όψη για τα δεδομένα της βάσης και διαθέτει ένα μηχανισμό XML ερωτημάτων ο οποίος μετατρέπει τα XML ερωτήματα σε αντίστοιχα SQL

ερωτήματα προς τη βάση δεδομένων και μεταφράζει τα αποτελέσματα αυτών των ερωτημάτων σε XML μορφή [25].

Το XPERANTO υποστηρίζει την μετατροπή σχεσιακών δομών δεδομένων σε XML μορφή, αλλά και αντικειμενοσχεσιακών, όπως πίνακες (Typed tables), στήλες (Typed columns) και αντικείμενα (Typed objects), με τύπους ορισμένους από το χρήστη, αναφορών (references) και oids, κληρονομικότητα (inheritance) και συλλογές (collections) [27].

5.5. Περίληψη Κεφαλαίου

Αλγόριθμος μετατροπής δεδομένων, είναι μια διαδικασία κατά την οποία τα δεδομένα που είναι αποθηκευμένα σε ένα σύστημα υπό μια μορφή, μετατρέπονται σε μια άλλη μορφή υπό κάποιους συγκεκριμένους περιορισμούς έτσι ώστε να μπορούν να αξιοποιηθούν από άλλα, συνήθως τεχνολογικά πιο εξελιγμένα, συστήματα.

Σε αυτό το κεφάλαιο, παρουσιάσαμε τους αλγόριθμους που προέκυψαν από την έρευνα που έγινε στα πλαίσια αυτής της εργασίας. Τους χωρίσαμε σε δύο κατηγορίες με βάση την αρχική δομή στην οποία βρίσκονται αποθηκευμένα τα δεδομένα, σε αλγόριθμους μετατροπής δεδομένων για το σχεσιακό και το αντικειμενοσχεσιακό μοντέλο αντίστοιχα.

Όσον αφορά το σχεσιακό μοντέλο, από την έρευνα πρόεκυψαν οι αλγόριθμοι ConvRel [13] και Holistic Constraint-Preserving Transformation Algorithm [22]. Ο δεύτερος αποτελεί εξέλιξη μιας προηγούμενης δουλειάς η οποία μπορεί να βρεθεί στο [23]. Επίσης, αναφέρθηκαν και άλλες προσεγγίσεις που εντάσσονται στην κατηγορία αυτή, όπως η μετατροπή δεδομένων που ανακτήθηκαν χρησιμοποιώντας μια SQL – based γλώσσα [24], ένας reverse – engineering αλγόριθμος για την εξαγωγή δεδομένων από βάσεις παλαιότερης τεχνολογίας [25], και το SilkRoute ένα ενδιάμεσο σύστημα για την δημιουργία XML δεδομένων από σχεσιακά συστήματα βάσεων δεδομένων [26].

Στην κατηγορία των αλγορίθμων μετατροπής δεδομένων για τα αντικειμενοσχεσιακά συστήματα παρουσιάσαμε συνοπτικά το XPERANTO [27] το οποίο είναι ένα ενδιάμεσο συστατικό που υποστηρίζει την μετατροπή σχεσιακών δομών δεδομένων αλλά και αντικειμενοσχεσιακών στα αντίστοιχα τους XML έγγραφα και σχήματα. Για το λόγο αυτό, θα μπορούσε να θεωρηθεί ότι ανήκει και στις δύο κατηγορίες.

6. Υλοποίηση εφαρμογής μετατροπής σχεσιακών και αντικειμενοσχεσιακών βάσεων δεδομένων σε αντίστοιχες XML μορφής

Σε αυτό το κεφάλαιο θα παρουσιάσουμε την εφαρμογή που αναπτύξαμε στα πλαίσια αυτής της εργασίας. Η εφαρμογή, μετατρέπει μια σχεσιακή ή αντικειμενοσχεσιακή βάση δεδομένων σε αντίστοιχη δομή δεδομένων XML μορφής.

6.1. Εισαγωγή

Στο προηγούμενο κεφάλαιο αναφερθήκαμε στους αλγόριθμους μετατροπής δεδομένων που έχουν προταθεί στη διεθνή βιβλιογραφία και αναφέραμε τους σημαντικότερους από αυτούς, με βάση την έρευνα μας, για κάθε κατηγορία. Σε αυτό το κεφάλαιο, παρουσιάζουμε την εφαρμογή που αναπτύξαμε για την υλοποίηση ενός αντίστοιχου αλγορίθμου που θα μπορεί να ανακτά τα δεδομένα από μια σχεσιακή ή μια αντικειμενοσχεσιακή βάση και να τα μετατρέπει στα αντίστοιχα XML έγγραφα και XML schemata.

Όπως ήδη αναφέραμε, οι αλγόριθμοι που περιγράφηκαν στο προηγούμενο κεφάλαιο, στην πλειοψηφία τους, βρίσκονται ακόμα σε θεωρητικό επίπεδο, είναι προτάσεις, και δεν έχουν υλοποιηθεί σε κάποιο εργαλείο, με εξαίρεση ίσως τους SilkRoute [13] και reverse-engineering αλγόριθμο στο [25] οι οποίοι έχουν υλοποιηθεί σε εργαλεία για ερευνητικούς σκοπούς.

Παρακάτω θα παρουσιάσουμε τις τεχνολογίες που χρησιμοποιήσαμε για να υλοποιήσουμε την εφαρμογή μας. Επίσης, θα παρουσιάσουμε τις κλάσεις που αναπτύχθηκαν για την εφαρμογή και τη διάταξη τους, τους αλγορίθμους που ακολουθήσαμε για το σχεσιακό και αντικειμενοσχεσιακό μοντέλο. Τέλος, θα μιλήσουμε για τον κώδικα της εφαρμογής παραθέτοντας παραδείγματα κώδικα και θα αναφέρουμε τις βασικότερες μεθόδους κάθε κλάσης.

6.2. Σκοπός της εφαρμογής

Ο σκοπός της εφαρμογής, είναι η μετατροπή σχεσιακών και αντικειμενοσχεσιακών βάσεων δεδομένων σε αντίστοιχες δομές δεδομένων XML μορφής. Δεν χρησιμοποιείται ο όρος «βάση δεδομένων» για την XML καθώς, όπως πολύ εύστοχα αναφέρεται στο [28], η XML δεν αποτελεί βάση δεδομένων παρά μόνο με την αυστηρότερη έννοια του όρου, δηλαδή ότι είναι μια «συλλογή δεδομένων».

Σε αντίθεση με τους αλγόριθμους που παραθέσαμε στο Κεφάλαιο 5, η μετατροπή που προσπαθούμε να επιτύχουμε εδώ εστιάζει στην αναπαράσταση των δεδομένων της βάσης, με αντίστοιχη έγκυρη XML δομή. Δεν επιδιώκουμε να κάνουμε reverse engineering της βάσης δεδομένων, από την αρχική σχεσιακή ή αντικειμενοσχεσιακής της μορφή και να την αναδημιουργήσουμε πλήρως υπό XML μορφή.

Συνεπώς, παραβλέπουμε σκοπίμως, και για λόγους απλότητας στην υλοποίηση, την πλήρη αναπαράσταση των συσχετίσεων μεταξύ των οντοτήτων και των γενικότερων περιορισμών ακεραιότητας (κύρια, ξένα κλειδιά) αν και αυτά εκφράζονται στο XML Schema με έναν τρόπο που θα αναλύσουμε παρακάτω.

6.3. Τεχνολογίες που χρησιμοποιήθηκαν

Για την ανάπτυξη της εφαρμογής μας χρησιμοποιήθηκαν διαφορετικές τεχνολογίες οι οποίες θεωρήθηκαν οι καταλληλότερες για να υποστηρίξουν τον στόχο αυτής της εφαρμογής. Οι τεχνολογίες που χρησιμοποιήθηκαν είναι οι εξής:

- RDBMS (PostgreSQL) & ORDBMS (DB2)
- Java & JDBC (NetBeans)
- Java Swing
- XML 1.0

Η εφαρμογή μας μπορεί να τρέξει για κάθε RDBMS, ωστόσο έχει υλοποιηθεί σκόπιμα περιορισμός για την PostgreSQL, στα πλαίσια των επιλογών που υπάρχουν στο GUI, το οποίο δημιουργήθηκε με χρήση της Java Swing.

Το ORDBMS που χρησιμοποιήθηκε στις δοκιμές και υποστηρίζεται από την εφαρμογή είναι η IBM DB2. Συγκεκριμένα, οι δοκιμές έγιναν σε μια τοπική βάση δεδομένων που δημιουργήθηκε στην IBM DB2 9.7.

Η εφαρμογή είναι προγραμματισμένη σε κώδικα Java χρησιμοποιώντας παράλληλα και εκτεταμένα το JDBC API. Το JDBC API περιγράφεται αναλυτικότερα στο Κεφάλαιο 2. Η ανάπτυξη της εφαρμογής έγινε στο περιβάλλον προγραμματισμού NetBeans 6.8.

Η εφαρμογή αφού ανακτήσει τα δεδομένα και τα μεταδεδομένα της βάσης δεδομένων, τα μετατρέπει στα αντίστοιχα της βάσης, XML έγγραφα και schemata, με βάση τις προδιαγραφές της XML 1.0 όπως ορίζονται στο [29].

6.4. Υλοποιημένοι αλγόριθμοι

Όπως αναφέρθηκε προηγουμένως, στόχος της εφαρμογής είναι να αναπαραστήσει τα δεδομένα της βάσης, σε αντίστοιχη XML δομή. Λόγω του ότι η μετατροπή γίνεται ακριβώς ως αυτό το επίπεδο, οι αλγόριθμοι που ακολουθούνται για τα δύο μοντέλα είναι κατά το μεγαλύτερο μέρος τους παρόμοιοι. Παρακάτω θα παραθέσουμε αναλυτικά τα βήματα που ακολουθεί ο κάθε αλγόριθμος και θα ακολουθήσει μια συνοπτική σύγκριση.

6.4.1. Υλοποιημένοι αλγόριθμοι για το σχεσιακό μοντέλο

Ο αλγόριθμος που περιγράφεται παρακάτω αφορά την μετατροπή του σχεσιακού σχήματος σε αντίστοιχο XML Schema. Τα βήματα που ακολουθούνται, αποφασίστηκαν μετά από την έρευνα που έγινε στα πλαίσια αυτής της εργασίας, η οποία εστίασε κυρίως στην εύρεση των συντακτικών περιορισμών και των περιορισμών δομής που διακρίνουν το XML Schema.

Συνεπώς, στόχος του αλγορίθμου που υλοποιήθηκε είναι η παραγωγή ενός καθαρού, συμπαγούς και καλώς ορισμένου (well – formed) XML σχήματος.

Τα βήματα του αλγορίθμου μετατροπής σχεσιακού σχήματος σε XML είναι τα εξής:

Βήμα 1ο : Δημιούργησε την XML διακήρυξη

Βήμα 2ο : Δημιούργησε ένα εξωτερικό στοιχείο 'schema' και δήλωσε ένα XML namespace.

Βήμα 3ο : Δημιούργησε ένα στοιχείο-ρίζα 'root', μέσα στο στοιχείο 'schema'

Βήμα 4ο : Δημιούργησε ένα στοιχείο 'complexType', μέσα στο στοιχείο 'root'

Βήμα 5ο : Δημιούργησε ένα στοιχείο 'sequence', μέσα στο στοιχείο 'complexType'

Βήμα 6ο : LOOP : Για κάθε πίνακα:

- δημιούργησε ένα στοιχείο (element) με χαρακτηριστικά για 'name' το όνομα του πίνακα, και για το δεσμευμένο χαρακτηριστικό 'maxOccurs' την τιμή 'unbounded'.
- Δημιούργησε ένα στοιχείο 'complexType' μέσα στο στοιχείο 'element'
- Δημιούργησε ένα στοιχείο 'sequence' μέσα στο στοιχείο 'complexType'
- **LOOP : Για κάθε στήλη του πίνακα :**
  - **Αν η στήλη είναι κύριο ή ξένο κλειδί :**
    - Δημιούργησε ένα στοιχείο (element) με χαρακτηριστικά, για 'name' το όνομα της στήλης, 'type' τον αντίστοιχο XML τύπο της στήλης και για το δεσμευμένο χαρακτηριστικό 'minOccurs' την τιμή '1'
  - **Αλλιώς:**
    - Δημιούργησε ένα στοιχείο (element) με χαρακτηριστικά, για 'name' το όνομα της στήλης, 'type' τον αντίστοιχο XML τύπο της στήλης και για το δεσμευμένο χαρακτηριστικό 'minOccurs' την τιμή '0'
- Κλείσε το στοιχείο 'sequence' για τον συγκεκριμένο πίνακα
- Κλείσε το στοιχείο 'complexType' για τον συγκεκριμένο πίνακα
- Κλείσε το στοιχείο 'element' για τον συγκεκριμένο πίνακα

Βήμα 7ο : Κλείσε το στοιχείο 'sequence' για το 'root' στοιχείο

Βήμα 8ο : Κλείσε το στοιχείο 'complexType' για το 'root' στοιχείο

Βήμα 9ο : Κλείσε το στοιχείο 'root'

Βήμα 10ο : Κλείσε το στοιχείο 'schema'

Ο παραπάνω αλγόριθμος δημιουργεί ένα καλώς ορισμένο (well formed) XSD έγγραφο, το XML Schema, όπως το παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="root">
        <xs:complexType>
              <xs:sequence>
              <!--TABLE boats-->
                <xs:element name="boats" maxOccurs="unbounded">
                      <xs:complexType>
                                  <xs:sequence>
                                  <xs:element name="bid" type="xs:int" minOccurs="1"/>
                                  <xs:element name="bname" type="xs:string" minOccurs="0"/>
                                  <xs:element name="color" type="xs:string" minOccurs="0"/>
                                  </xs:sequence>
                      </xs:complexType>
                </xs:element>
                      <!--TABLE reserves-->
                <xs:element name="reserves" maxOccurs="unbounded">
                                  <xs:complexType>
                                        <xs:sequence>
                                        <xs:element name="sid" type="xs:int" minOccurs="1"/>
                                        <xs:element name="bid" type="xs:int" minOccurs="1"/>
                                        <xs:element name="day" type="xs:date" minOccurs="1"/>
                                        </xs:sequence>
                                  </xs:complexType>
                </xs:element>
                <!--TABLE sailors-->
                <xs:element name="sailors" maxOccurs="unbounded">
                                  <xs:complexType>
                                        <xs:sequence>
                                        <xs:element name="sid" type="xs:int" minOccurs="1"/>
                                        <xs:element name="sname" type="xs:string" minOccurs="0"/>
                                        <xs:element name="rating" type="xs:int" minOccurs="0"/>
                                        <xs:element name="age" type="xs:float" minOccurs="0"/>
                                        </xs:sequence>
                                  </xs:complexType>
                </xs:element>
              </xs:sequence>
        </xs:complexType>
  </xs:element>
</xs:schema>
```

Για περισσότερες πληροφορίες για τα στοιχεία «minOccurs», «maxOccurs», «element», «complexType» και «sequence» μπορείτε να ανατρέξετε στο Κεφάλαιο 3, όπου τα έχουμε ήδη αναλύσει.

Στον αλγόριθμο για την μετατροπή του σχεσιακού σχήματος σε XML Schema, αναφέρουμε ότι η τιμή του χαρακτηριστικού “type” ενός στοιχείου element θα είναι «ο αντίστοιχος XML τύπος της στήλης». Αυτό συμβαίνει γιατί το ΣΔΒΔ μας επιστρέφει στο JDBC τον SQL τύπο της αντίστοιχης στήλης. Όμως δεν υποστηρίζονται όλοι οι τύποι της SQL απευθείας από το XML Schema. Για το λόγο αυτό είναι απαραίτητη η μετατροπή τους. Στην εφαρμογή προβλέπεται και υλοποιείται αυτή η μετατροπή. Η αντιστοίχηση των τύπων έγινε με βάση το [30].

Αντίστοιχος είναι και ο αλγόριθμος που ακολουθείται για την παραγωγή του XML εγγράφου. Παρουσιάζονται, βέβαια διαφορές αφού κάποια στοιχεία που δεν χρειαζόμαστε στο XML Schema, όπως ο αριθμός εγγραφών ανά πίνακα, τα χρειαζόμαστε για να δημιουργήσουμε το XML έγγραφο και αντίστροφα.

Τα βήματα του αλγορίθμου μετατροπής των δεδομένων σε XML έγγραφο είναι τα εξής:

Βήμα 1ο : Δημιούργησε την XML διακήρυξη

Βήμα 2ο : Δημιούργησε ένα στοιχείο-ρίζα ‘root’, και δήλωσε το ίδιο XML namespace που δηλώθηκε στο XML Schema, καθώς και την τοποθεσία του αρχείου του σχήματος.

Βήμα 3ο : LOOP : Για κάθε πίνακα:

- **LOOP : Για κάθε γραμμή (εγγραφή) του πίνακα :**
  - δημιούργησε ένα στοιχείο <όνομα_πίνακα>
    - **LOOP : Για κάθε στήλη του πίνακα :**
    - **ΑΝ η τιμή της στήλης δεν είναι null ή κενή**
      - Δημιούργησε ένα στοιχείο <όνομα_στήλης>’τιμή’</όνομα στήλης> , όπου ‘τιμή’ η τιμή της στήλης (δεδομένα)
  - Κλείσε το στοιχείο <όνομα_πίνακα>

Βήμα 4ο : Κλείσε το στοιχείο ‘root’

Ο αλγόριθμος μετατροπής των δεδομένων σε XML έγγραφο δημιουργεί ένα καλώς ορισμένο και έγκυρο (valid) XML έγγραφο όπως το παρακάτω.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Relational_schema.xsd">
<!-- Στην παραπάνω XML δήλωση, με το χαρακτηριστικό “noNamespaceSchemaLocation” δηλώνουμε ότι αυτό το έγγραφο δεν χρησιμοποιεί Namespace και ακολουθεί ένα τοπικό στιγμιότυπο σχήματος (xsi) με το όνομα “Relational_schema.xsd”», το οποίο βρίσκεται στον ίδιο φάκελο με το έγγραφο  -->
<!--boats TABLE-->
<boats>
        <bid>101</bid>
        <bname>Interlake</bname>
        <color>Blue</color>
</boats>
<boats>
        <bid>102</bid>
        <bname>Interlake</bname>
        <color>Red</color>
</boats>
<boats>
<!--reserves TABLE-->
<reserves>
        <sid>22</sid>
        <bid>101</bid>
        <day>1998-10-10</day>
</reserves>
<reserves>
        <sid>22</sid>
        <bid>102</bid>
        <day>1998-10-10</day>
</reserves>
<reserves>
        <sid>31</sid>
        <bid>102</bid>
        <day>1998-11-10</day>
</reserves>
<!--sailors TABLE-->
<sailors>
        <sid>22</sid>
        <sname>Dustin</sname>
        <rating>7</rating>
        <age>45</age>
</sailors>
<sailors>
        <sid>31</sid>
        <sname>Lubber</sname>
        <rating>8</rating>
        <age>55.5</age>
</sailors>
```

Σημειώνεται ότι το παραπάνω XML έγγραφο αποτελεί παράδειγμα από το οποίο έχει αποκοπεί πληροφορία για λόγους εξοικονόμησης χώρου και εύκολης ανάγνωσης.

6.4.2. Υλοποιημένοι αλγόριθμοι για το αντικειμενοσχεσιακό μοντέλο

Ο αλγόριθμος που υλοποιήθηκε για την μετατροπή των δεδομένων του αντικειμενοσχεσιακού μοντέλου σε αντίστοιχη XML δομή, διακρίνεται, επίσης, από τη φιλοσοφία των προηγούμενων αλγορίθμων που αφορούσαν το σχεσιακό μοντέλο.

Πέρα από την παρατήρηση που κάναμε παραπάνω, ότι δηλαδή μας ενδιαφέρει να αναπαραστήσουμε τα δεδομένα της βάσης με αντίστοιχη έγκυρη XML δομή, στο σημείο αυτό κάνουμε και μια παραδοχή για λόγους διευκόλυνσης στην ανάπτυξη της εφαρμογής.

Παραδοχή: Σε κάθε πίνακα της αντικειμενοσχεσιακής βάσης δεδομένων, συγκεκριμένα της DB2, στην πρώτη στήλη υπάρχει το αντίστοιχο πεδίο του πεδίου oid , το οποίο ορίζεται πάντα από τον χρήστη με οποιοδήποτε όνομα. Δεν είναι το μοναδικό αναγνωριστικό που ανατίθεται από το DBMS για όλο το σύστημα για κάθε αντικείμενο, αλλά είναι η παράμετρος με την οποία αυτό δημιουργείται.

Επίσης ο αλγόριθμος αυτός δεν εστιάζει και δεν αναπαριστά τις συσχετίσεις και περιορισμούς ακεραιότητας όπως τα κύρια και τα ξένα κλειδιά, στα παραγόμενα XML και XSD αρχεία, παρά το γεγονός ότι αυτά υποστηρίζονται και από τα αντικειμενοσχεσιακά συστήματα. Κάτι τέτοιο απαιτεί μια εκτεταμένη έρευνα και δεν είναι δυνατό να παρουσιαστεί στα πλαίσια μιας πτυχιακής εργασίας. Τέλος, θεωρούμε ότι δεν υπάρχουν απλοί πίνακες στη βάση, αλλά μόνο πίνακες αντικειμένων, με τύπους ορισμένους από τον χρήστη (UDT – User Distinct Types).

Λαμβάνοντας υπ’ όψη τις παραπάνω παρατηρήσεις και παραδοχές, θα παρουσιάσουμε παρακάτω τον αλγόριθμο μετατροπής των αντικειμενοσχεσιακών σχημάτων σε αντίστοιχο XML Schema, και αμέσως μετά τον αλγόριθμο για την μετατροπή των δεδομένων σε XML έγγραφο.

Τα βήματα του αλγορίθμου μετατροπής αντικειμενοσχεσιακού σχήματος σε XML είναι τα εξής:

Βήμα 1ο : Δημιούργησε την XML διακήρυξη

Βήμα 2ο : Δημιούργησε ένα εξωτερικό στοιχείο 'schema' και δήλωσε ένα XML namespace.

Βήμα 3ο : Δημιούργησε ένα στοιχείο-ρίζα 'root', μέσα στο στοιχείο 'schema'

Βήμα 4ο : Δημιούργησε ένα στοιχείο 'complexType', μέσα στο στοιχείο 'root'

Βήμα 5ο : Δημιούργησε ένα στοιχείο 'sequence', μέσα στο στοιχείο 'complexType'

Βήμα 6ο : LOOP : Για κάθε πίνακα (σετ διαρκών αντικειμένων):

- δημιούργησε ένα στοιχείο (element) με χαρακτηριστικά για 'name' το όνομα του πίνακα, και για το δεσμευμένο χαρακτηριστικό 'maxOccurs' την τιμή 'unbounded'.
- Δημιούργησε ένα στοιχείο 'complexType' μέσα στο στοιχείο 'element'
- Δημιούργησε ένα στοιχείο 'sequence' μέσα στο στοιχείο 'complexType'
- **LOOP : Για κάθε στήλη του πίνακα <u>εκτός από την πρώτη</u>:**
    - Δημιούργησε ένα στοιχείο (element) με χαρακτηριστικά, για 'name' το όνομα της στήλης, 'type' τον αντίστοιχο XML τύπο της στήλης και για το δεσμευμένο χαρακτηριστικό 'minOccurs' την τιμή '0'

- Κλείσε το στοιχείο 'sequence' για τον συγκεκριμένο πίνακα
- Δημιούργησε ένα χαρακτηριστικό (attribute) με χαρακτηριστικά «όνομα της 1ης στήλης='τιμή της 1ης στήλης'», το 'type' και τιμή τον τύπο της 1ης στήλης και το δεσμευμένο χαρακτηριστικό 'use' και τιμή 'required'
- Κλείσε το στοιχείο 'complexType' για τον συγκεκριμένο πίνακα
- Κλείσε το στοιχείο 'element' για τον συγκεκριμένο πίνακα

Βήμα 7ο : Κλείσε το στοιχείο 'sequence' για το 'root' στοιχείο

Βήμα 8ο : Κλείσε το στοιχείο 'complexType' για το 'root' στοιχείο

Βήμα 9ο : Κλείσε το στοιχείο 'root'

Βήμα 10ο : Κλείσε το στοιχείο 'schema'

Όπως φαίνεται, κατά την μελέτη και σύγκριση των αλγορίθμων που παράγουν τα XML schemata από το σχεσιακό και αντικειμενοσχεσιακό σχήμα αντίστοιχα, οι δύο αλγόριθμοι έχουν ελάχιστες διαφορές. Η μια είναι η παράλειψη της πρώτης στήλης ως στοιχείο και δήλωση της ως χαρακτηριστικό, με βάση και την παραδοχή που έγινε νωρίτερα.

Η δεύτερη αφορά στο χαρακτηριστικό use=”required” το οποίο υποδεικνύει ότι το χαρακτηριστικό γονέας του θα πρέπει οπωσδήποτε να περιλαμβάνεται σε κάθε element που αναπαριστά πίνακα, στο XML Schema.

Ο παραπάνω αλγόριθμος δημιουργεί ένα καλώς ορισμένο (well – formed) XSD έγγραφο, το XML Schema, όπως το παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="root">
	<xs:complexType>
		<xs:sequence>
<!--TYPED TABLE DROMOLOGIO is of type DROMOLOGIO_T -->

	<xs:element name="dromologio" maxOccurs="unbounded">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="afetiria" type="xs:string" minOccurs="0" />
				<xs:element name="proorismos" type="xs:string" minOccurs="0" />
				<xs:element name="hmera_dromologiou" type="xs:string" minOccurs="0"
/>
				<xs:element name="ora_dromologiou" type="xs:string" minOccurs="0" />
				<xs:element name="leoforeio" type="xs:string" minOccurs="0" />
				<xs:element name="timi_eisitiriou" type="xs:int" minOccurs="0" />
			</xs:sequence>
			<xs:attribute name="OID" type="xs:string" use="required"/>
		</xs:complexType>
	</xs:element>
<!--TYPED TABLE LEOFOREIO is of type LEOFOREIO_T -->

	<xs:element name="leoforeio" maxOccurs="unbounded">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="arithmos" type="xs:string" minOccurs="0" />
				<xs:element name="edra" type="xs:string" minOccurs="0" />
				<xs:element name="hmer_kataskeuhs" type="xs:date" minOccurs="0" />
				<xs:element name="pliroma" type="xs:string" minOccurs="0" />
				<xs:element name="xiliometra" type="xs:int" minOccurs="0" />
			</xs:sequence>
			<xs:attribute name="OID" type="xs:string" use="required"/>
		</xs:complexType>
	</xs:element>
		</xs:sequence>
	</xs:complexType>
</xs:element>
</xs:schema>
```

Στο παραπάνω XML Schema έχει σκόπιμα παραληφθεί ένα μέρος της πληροφορίας.

Στο σχήμα συμπεριλαμβάνονται και πληροφορίες για τα ονόματα όλων των ορισμένων από τον χρήστη τύπων καθώς και τα ονόματα των συναρτήσεων που βρέθηκαν στο σχήμα, όπως φαίνεται παρακάτω. Οι συναρτήσεις με τα ονόματα «>» (μεγαλύτερο από), «<=» (μικρότερο ή ίσο) κτλ. είναι συναρτήσεις που δημιουργήθηκαν αυτόματα από την DB2.

!-- ****** A list of all the User Types ****** -->	<!-- ****** A list of all the functions ****** -->
<!-- ARITHMOS_KUKLOFORIAS -->	<!-- < -->
<!-- DROMOLOGIO_T -->	<!-- <= -->
<!-- LEOFOREIO_T -->	<!-- <> -->
<!-- ODIGOS_T -->	<!-- = -->
<!-- PLIROMA_T -->	<!-- > -->
<!-- POLI_T -->	<!-- >= -->
<!-- SYNODOS_T -->	<!-- AFETIRIA -->

Ο αλγόριθμος μετατροπής των δεδομένων για το αντικειμενοσχεσιακό μοντέλο σε XML έγγραφο είναι ο εξής:

Βήμα 1ο : Δημιούργησε την XML διακήρυξη

Βήμα 2ο : Δημιούργησε ένα στοιχείο-ρίζα 'root', και δήλωσε το ίδιο XML namespace που δηλώθηκε στο XML Schema, καθώς και την τοποθεσία του αρχείου του σχήματος.

Βήμα 3ο : LOOP : Για κάθε πίνακα:

- **LOOP : Για κάθε γραμμή (εγγραφή) του πίνακα :**
  - δημιούργησε ένα στοιχείο <όνομα_πίνακα>, με χαρακτηριστικό το όνομα της $1^{ης}$ στήλης και για τιμή, την τιμή της $1^{ης}$ στήλης
    - **LOOP : Για κάθε στήλη του πίνακα :**
    - **ΑΝ η τιμή της στήλης δεν είναι null ή κενή**
      - Δημιούργησε ένα στοιχείο <όνομα_στήλης>'τιμή'</όνομα στήλης> , όπου 'τιμή' η τιμή της στήλης (δεδομένα)
  - Κλείσε το στοιχείο <όνομα_πίνακα>

Βήμα 4ο : Κλείσε το στοιχείο 'root'

Η διαφορά του αλγόριθμου μετατροπής δεδομένων στο σχεσιακό και το αντικειμενοσχεσιακό μοντέλο, έγκειται στο ότι στο δεύτερο χρησιμοποιούμε την πρώτη στήλη κάθε πίνακα ως το μοναδικό αναγνωριστικό κάθε αντικειμένου που ορίζεται από τον χρήστη (βλέπε παραδοχή) και χρησιμοποιείται στο XML έγγραφο ως χαρακτηριστικό για κάθε αντικείμενο – εγγραφή του πίνακα.

Ο παραπάνω αλγόριθμος δημιουργεί ένα καλώς ορισμένο και έγκυρο XML έγγραφο όπως το παρακάτω, το οποίο είναι απλά ένα δείγμα του τελικού XML εγγράφου.

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

<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Object-Relational_schema.xsd">

<!-- Στην παραπάνω XML δήλωση, με το χαρακτηριστικό “noNamespaceSchemaLocation” δηλώνουμε ότι αυτό το έγγραφο δεν χρησιμοποιεί Namespace και ακολουθεί ένα τοπικό στιγμιότυπο σχήματος (xsi) με το όνομα “Object-Relational_schema.xsd”», το οποίο βρίσκεται στον ίδιο φάκελο με το έγγραφο  -->


<!--DROMOLOGIO TYPED TABLE-->
<dromologio OID="1">
        <afetiria>10</afetiria>
        <proorismos>20</proorismos>
        <hmera_dromologiou>Deutera</hmera_dromologiou>
        <ora_dromologiou>10:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="2">
        <afetiria>10</afetiria>
        <proorismos>20</proorismos>
        <hmera_dromologiou>Triti</hmera_dromologiou>
        <ora_dromologiou>10:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

</dromologio>

<!--LEOFOREIO TYPED TABLE-->
<leoforeio OID="10">
        <arithmos>NHE1010</arithmos>
        <edra>10</edra>
        <hmer_kataskeuhs>2007-01-05</hmer_kataskeuhs>
        <pliroma>1</pliroma>
        <xiliometra>10000</xiliometra>
</leoforeio>

<!--ODIGOS TYPED TABLE-->
……
<!--PLIROMA TYPED TABLE-->
……
<!--POLI TYPED TABLE-->
……
<!--SYNODOS TYPED TABLE-->
……
<!--YPALLILOS TYPED TABLE-->
……

</root>
```

6.4.3. Χειρισμός null τιμών

Οι αλγόριθμοι που περιγράφηκαν παραπάνω χειρίζονται το θέμα των null τιμών ενιαία. Η λογική είναι η εξής. Οι αλγόριθμοι που παράγουν το XSD αρχείο, δηλαδή το XML Schema για κάθε στοιχείο που δεν είναι κύριο ή ξένο κλειδί προσθέτουν στο στοιχείο το χαρακτηριστικό “minOccurs=0” το οποίο υποδεικνύει ότι το συγκεκριμένο στοιχείο μπορεί να παραλειφθεί στο XML έγγραφο.

Η λογική αυτή εξειδικεύεται λίγο διαφορετικά στον αλγόριθμο που αφορά την μετατροπή του αντικειμενοσχεσιακού σχήματος. Με βάση την παραδοχή που κάνουμε, δεν αναζητάμε τα κύρια και ξένα κλειδιά των πινάκων οπότε δεν μπορεί να γίνει αντίστοιχος καθολικός έλεγχος.

Ωστόσο, παραδεχόμαστε ότι η πρώτη στήλη κάθε πίνακα περιέχει ένα μοναδικό αναγνωριστικό (στα παραδείγματα μας το OID). Συνεπώς, θεωρούμε τις υπόλοιπες στήλες ως στοιχεία τα οποία μπορούν να παραληφθούν και για το λόγο αυτό προσθέτουμε και εδώ το χαρακτηριστικό “minOccurs=0”.

Όσον αφορά τα XML έγγραφα, και στους δύο αλγορίθμους, πριν εξάγουμε ένα στοιχείο του πίνακα ως μέρος της XML δομής, ελέγχουμε αν η τιμή της στήλης που αντιστοιχεί σε αυτό έχει την τιμή null ή είναι κενή αλφαριθμητικά (“ ”). Επομένως, όταν ισχύει η παραπάνω συνθήκη επιλέγουμε να παραλείψουμε το στοιχείο και να μην το προσθέσουμε στο XML έγγραφο ενώ αν δεν ισχύει το συμπεριλαμβάνουμε κανονικά.

Αυτός ο τρόπος χειρισμού των null τιμών εξυπηρετεί κυρίως ένα σκοπό. Την αποφυγή πλεονάζουσας πληροφορίας (redundancy) η οποία θα μπορούσε να δημιουργήσει προβλήματα όπως δυσκολία στην ανάγνωση του XML εγγράφου και προφανώς αύξηση του μεγέθους του αρχείου στο δίσκο.

Η τελευταία συνέπεια είναι ιδιαίτερα σημαντική αν αναλογιστούμε το γεγονός ότι η XML χρησιμοποιείται ευρέως από εφαρμογές και web services για την ανταλλαγή δεδομένων μέσω του διαδικτύου. Σε μεγάλες βάσεις δεδομένων, όπου το μέγεθος του παραγόμενου XML εγγράφου θα είναι αντίστοιχο, ο τρόπος χειρισμού των null

τιμών που περιγράψαμε παραπάνω θα μπορούσε να αποβεί σωτήριος ακόμα και για την ίδια τη λειτουργικότητα των εφαρμογών.

6.5. Η εφαρμογή : κλάσεις, μέθοδοι και κώδικας

Όπως αναφέρθηκε και προηγουμένως η εφαρμογή προγραμματίστηκε στη γλώσσα προγραμματισμού Java χρησιμοποιώντας εκτεταμένα το JDBC API και η υλοποίηση έγινε στο προγραμματιστικό περιβάλλον NetBeans 6.8.

Θεωρούμε σκόπιμο, να παρουσιάσουμε παρακάτω περιγραφικά, τον τρόπο λειτουργίας της εφαρμογής παραθέτοντας εικόνες και πίνακες όπου αυτό κρίνεται απαραίτητο. Επίσης, θα παρουσιάσουμε τις κλάσεις που συνθέτουν την εφαρμογή καθώς και τις βασικότερες μεθόδους τους. Τέλος, θα αναφέρουμε τα προβλήματα που συναντήσαμε στα πλαίσια της ανάπτυξης της εφαρμογής.

Σε αυτό το σημείο, πρέπει να τονίσουμε ότι δεν θεωρούμε επωφελή την παράθεση και περιγραφή του κώδικα της εφαρμογής γραμμή προς γραμμή για τον αναγνώστη, αντιθέτως, μάλλον αποπροσανατολιστική. Συνεπώς, δεν θα παραθέσουμε κομμάτια του κώδικα της εφαρμογής, αλλά μια οργανωμένη παρουσίαση των βασικών στοιχείων της (κλάσεις, διάταξη κλάσεων, μέθοδοι κτλ).

6.5.1. Κλάσεις και διάταξη

Η εφαρμογή μας αποτελείται από διάφορες κλάσεις κάθε μια από τις οποίες εκπληρώνει διαφορετικό, προφανώς, στόχο.

Οι κλάσεις που συνιστούν την εφαρμογή μας είναι πέντε (5) στο σύνολο τους και είναι οι εξής:

- MapGUI (MapGUI.java)
- RelSchema (RelSchema.java)
- RelMap (RelMap.java)
- ORSchema (ORSchema.java)
- ORMap (ORMap.java)

Στο παράρτημα Ι δίνεται αναλυτικά όλος ο κώδικας της εφαρμογής, ανά κλάση καθώς και τα παραγόμενα xml και xsd αρχεία.

Η διάταξη των κλάσεων θα μπορούσε να παρασταθεί όπως στην εικόνα 6.1. Προσοχή, η αναπαράσταση των κλάσεων με αυτή τη μορφή δεν δηλώνει κληρονομικότητα της κλάσης MapGUI με τις υπόλοιπες. Υποδηλώνει την ροή των δεδομένων, όπως φαίνεται και από τα αντίστοιχα βελάκια.

Εικόνα 6.1. Η διάταξη των κλάσεων της εφαρμογής

Κλάση MapGUI

Σε αυτή την κλάση έχει υλοποιηθεί όλο το Graphical User Interface της εφαρμογής. Λειτουργεί, δηλαδή, και ως κλάση εισόδου δεδομένων. Ο χρήστης εισάγει τα δεδομένα , όπως:

- το όνομα του server στον οποίο φιλοξενείται η βάση (host)
- το όνομα χρήστη (username) και τον κωδικό πρόσβασης (password)
- την θύρα στην οποία θα γίνει η σύνδεση με τη βάση (port)
- το όνομα της βάσης
- την τοποθεσία στην οποία θέλει να εξάγει τα παραγόμενα xml και xsd αρχεία (Export Location)
- το όνομα του παραγόμενου XML εγγράφου

Τέλος, ο χρήστης μπορεί να επιλέξει ποιο DBMS θέλει να μετατρέψει. Η επιλογή γίνεται με ένα ComboBox στο οποίο ο χρήστης μπορεί να επιλέξει μεταξύ της PostgreSQL και της IBM DB2.

Ανάλογα με την επιλογή τα δεδομένα αυτά θα περαστούν στις αντίστοιχες κλάσεις. Για την PostgreSQL, οι κλάσεις που υλοποιούν την μετατροπή (mapping) είναι οι

RelSchema (για το XML Schema) και η RelMap (για το XML έγγραφο). Αντίστοιχα, για την IBM DB2 , είναι οι ORSchema και ORMap.

Επίσης στην κλάση αυτή, έχουν υλοποιηθεί κάποιοι βασικοί αυτοματισμοί, όπως η κατασκευή του connection String με βάση τα στοιχεία που δίνονται από τον χρήστη, στις ενότητες «Database Properties» και «Connection & Login Details».

Ακόμα, ο χρήστης δεν χρειάζεται να δηλώσει όνομα για το XML Schema καθώς αυτό δημιουργείται αυτόματα από το όνομα αρχείου που δίνει ο χρήστης για το XML έγγραφο, προσθέτοντας την κατάληξη «_schema.xsd» στις κλάσεις που δημιουργείται το schema. Τέλος, μολονότι ο χρήστης μπορεί να επιλέξει να συμπληρώσει χειροκίνητα την θύρα για τη σύνδεση με τη βάση, για τα δύο DBMS που υποστηρίζονται ο αριθμός της θύρας ανατίθεται αυτόματα.

Κλάση RelSchema

Σε αυτή την κλάση γίνεται η μετατροπή του σχεσιακού σχήματος στο αντίστοιχο XML Schema με βάση τον αλγόριθμο που αναλύθηκε στην ενότητα 6.4.1. Το παραγόμενο αποτέλεσμα είναι ένα καλώς ορισμένο xsd αρχείο.

Κλάση RelMap

Σε αυτή την κλάση γίνεται η μετατροπή των δεδομένων της σχεσιακής βάσης στο αντίστοιχο XML έγγραφο. Ο αλγόριθμος της μετατροπής αναλύθηκε στην ενότητα 6.4.1. Το παραγόμενο αποτέλεσμα είναι ένα καλώς ορισμένο και έγκυρο XML έγγραφο.

Κλάση ORSchema

Σε αυτή την κλάση γίνεται η μετατροπή του σχεσιακού σχήματος στο αντίστοιχο XML Schema με βάση τον αλγόριθμο που αναλύθηκε στην ενότητα 6.4.2. Το παραγόμενο αποτέλεσμα είναι ένα καλώς ορισμένο xsd αρχείο.

Κλάση RelMap

Σε αυτή την κλάση γίνεται η μετατροπή των δεδομένων της σχεσιακής βάσης στο αντίστοιχο XML έγγραφο. Ο αλγόριθμος της μετατροπής αναλύθηκε στην ενότητα

6.4.2. Το παραγόμενο αποτέλεσμα είναι ένα καλώς ορισμένο και έγκυρο XML έγγραφο.

6.5.2. Περιγραφή βασικών μεθόδων και λειτουργίας

Στο σημείο αυτό θα παραθέσουμε τις βασικότερες μεθόδους κάθε κλάσης που έχουν άμεση σχέση με τη λειτουργία της εφαρμογής, καθώς και άλλες που θεωρούμε ότι έχουν βαρύνουσα σημασία.

Βασικές μέθοδοι της MapGUI		
Μέθοδος	**Παράμετροι**	**Επιστρέφει**
run()	-	-
jButton1ActionPerformed()	java.awt.event.ActionEvent evt	-
jButton2ActionPerformed()	java.awt.event.ActionEvent evt	-
clearPassword()	-	-
checkDBMS()	-	-
purgeFields()	-	-
nullifyValues()	-	-

Πίνακας π6.1: Βασικές μέθοδοι της MapGUI

Οι βασικές μέθοδοι της κλάσης MapGUI που φαίνονται στον πίνακα π6.1. , έχουν άμεση σχέση με την σωστή λειτουργία της κλάσης. Πιο συγκεκριμένα, η μέθοδος “run()” η οποία βρίσκεται μέσα στην main μέθοδο της κλάσης ευθύνεται για τη δημιουργία του γραφικού περιβάλλοντος της εφαρμογής, του GUI.

Η μέθοδος “jButton1ActionPerformed()” λαμβάνει τα δεδομένα εισόδου που δίνει ο χρήστης μέσω του GUI και εκτελώντας διάφορους ελέγχους, δημιουργεί τα κατάλληλα αντικείμενα κλάσεων και καλεί τις κατάλληλες μεθόδους προκειμένου να πραγματοποιηθεί η μετατροπή. Επίσης, σε αυτή τη μέθοδο δημιουργείται το connectionString και οι συνδέσεις με τη βάση.

Η μέθοδος “jButton2ActionPerformed()” δημιουργεί ένα νέο διάλογο για την εύρεση της τοποθεσίας εξαγωγής των παραγόμενων, από τη μετατροπή, αρχείων.

Η μέθοδος clearPassword() καθαρίζει τις μεταβλητές και τις δομές που έχει αποθηκευτεί προηγούμενα ο κωδικός καθώς σύμφωνα με την τεκμηρίωση της Java η παραμονή του στις μεταβλητές αποτελεί ρίσκο ασφαλείας.

Η ckeckDBMS() αναλαμβάνει να ελέγξει ποιο DBMS έχει επιλεγεί από τον χρήστη.

Οι μέθοδοι purgeFields() και nullifyValues() χρεώνονται να «καθαρίσουν» τα πεδία του GUI και τις μεταβλητές στις οποίες αποθηκεύονται οι τιμές τους, αντίστοιχα.

Κοινές μέθοδοι για όλες τις υπόλοιπες κλάσεις		
Μέθοδος	**Παράμετροι**	**Επιστρέφει**
initConnection()	String	Connection
closeConnection()	-	-
trimWhite()	String	String

Πίνακας π6.2: Κοινές μέθοδοι για τις κλάσεις RelSchema, RelMap, ORSchema και ORMap

Οι μέθοδοι που αναφέρονται παραπάνω (βλ. πίνακα 6.2) είναι κοινές για όλες τις υπόλοιπες κλάσεις.

Η μέθοδος initConnection() είναι αυτή που δημιουργεί τη σύνδεση με τη βάση δεδομένων και την δημιουργία του statement, ενώ η closeConnection() είναι αυτή που την τερματίζει. Η closeConnection() φροντίζει, επίσης, το κλείσιμο του statement αλλά και του PrintStream το οποίο χρησιμοποιείται για την έκδοση των παραγόμενων αρχείων.

Η trimWhite() φροντίζει για την απαλοιφή κενών χαρακτήρων γύρω από το String που δέχεται ως όρισμα. Οι κενοί χαρακτήρες μπορεί να δημιουργήσουν πρόβλημα στην ανάγνωση, ακόμα και στην εγκυρότητα του XML σχήματος και εγγράφου, επομένως η υλοποίηση μια τέτοιας μεθόδου θεωρείται πολύ σημαντική.

Βασικές μέθοδοι της RelSchema		
Μέθοδος	**Παράμετροι**	**Επιστρέφει**
createRelXMLSchema()	Connection, String, String	-
getPKeys()	DatabaseMetaData , String, String	-
isPKey()	String	Boolean
getFKeys()	DatabaseMetaData , String, String	-
isFKey()	String	Boolean
getXMLType()	int	String

Πίνακας π6.3: Βασικές μέθοδοι της RelSchema

Στην κλάση RelSchema, είναι αρκετές οι μέθοδοι στις οποίες εστιάζουμε την προσοχή μας, όπως φαίνεται στον πίνακα π6.3. Η createRelXMLSchema() είναι η εναρκτήρια μέθοδος που καλείται στην κλάση MapGUI και με τη σειρά της καλεί όλες τις απαραίτητες μεθόδους προκειμένου να πραγματοποιήσει την μετατροπή του σχεσιακού σχήματος σε XML Schema.

Οι μέθοδοι getPKeys, getFKeys ανακτούν τα κύρια και τα ξένα κλειδιά από τους πίνακες της βάσης, ενώ οι isPKey, isFKey ελέγχουν αν μια στήλη είναι κύριο ή ξένο κλειδί αντίστοιχα.

Η getXMLType είναι πολύ χρήσιμη μέθοδος, καθώς μετατρέπει τους τύπους του DBMS, από την μορφή τύπων που υποστηρίζει το JDBC σε τύπους που να υποστηρίζονται από την XML και συγκεκριμένα από το XML Schema. Για περισσότερα ανατρέξτε στην ενότητα 2.6.

Βασικές μέθοδοι της RelMap		
Μέθοδος	**Παράμετροι**	**Επιστρέφει**
StartMappingRDBMStoXML()	Connection, String, String	-
getDBMetadata()	Connection	DatabaseMetaData
getTables()	DatabaseMetaData	-
getRecords()	String, Statement	ResultSet
getColumnsForEachTable()	DatabaseMetaData ,ResultSet ,String, String, PrintStream	-

Πίνακας π6.4: Βασικές μέθοδοι της RelMap

Η μέθοδος StartMappingRDBMStoXML, είναι η εναρκτήρια μέθοδος της κλάσης RelMap. Καλείται και αυτή στην κλάση MapGUI και με τη σειρά της καλεί όλες εκείνες τις μεθόδους που θα πραγματοποιήσουν την ανάκτηση και την μετατροπή των δεδομένων από τη σχεσιακή βάση στο αντίστοιχο XML έγγραφο.

Κάποιες από αυτές, είναι οι getDBMetadata η οποία ανακτά τα μεταδεδομένα της βάσης και τα αποθηκεύει σε ένα αντικείμενο DatabaseMetaData, η getTables η οποία ανακτά τα ονόματα και τους τύπους των πινάκων της βάσης και τα αποθηκεύει σε αντικείμενα Vector, η getRecords() η οποία εκτελεί ένα ερώτημα (query) προς τη βάση λαμβάνοντας για κάθε πίνακα τα δεδομένα του και τα αποθηκεύει σε ένα αντικείμενο ResultSet και τέλος η getColumnsForEachTable η οποία πέρα από το ότι ανακτά τα ονόματα και τους τύπους στηλών για κάθε πίνακα αποτελεί τον πυρήνα της μετατροπής, αφού μέσα στο σώμα της μεθόδου γίνεται η ετικετοποίηση, το λέγομενο XML Tagging.

Οι περισσότερες από τις μεθόδους της προηγούμενης παραγράφου, υλοποιούνται με τον ίδιο ή παρόμοιο τρόπο και στις άλλες κλάσεις, ανάλογα με τις ανάγκες. Δεν θεωρείται σκόπιμη η αναφορά τους σε κάθε πίνακα για λόγους διακριτότητας των σημαντικών μεθόδων.

Βασικές μέθοδοι της ORSchema		
Μέθοδος	**Παράμετροι**	**Επιστρέφει**
createORSchema()	Connection, String, String	-
getXMLType()	int	String
getUserDefinedTypes()	-	-

Πίνακας π6.5: Βασικές μέθοδοι της ORSchema

Στην κλάση ORSchema, ξεχωρίζουμε τις μεθόδους που φαίνονται στον πίνακα 6.5.

Η createORSchema είναι η εναρκτήρια μέθοδος αυτής της κλάσης, η οποία όπως και οι προηγούμενες καλείται στην κλάση MapGUI και με τη σειρά της καλεί όλες τις απαραίτητες μεθόδους για την μετατροπή του αντικειμενοσχεσιακού σχήματος σε αντίστοιχο XML Schema.

Η μέθοδος getXMLType, έχει την ίδια λειτουργία που έχει και στην κλάση RelSchema. Η καινούρια μέθοδος εδώ είναι η getUserDefinedTypes η οποία ανακτά απευθείας με ερώτημα στην αντικειμενοσχεσιακή βάση τους τύπους που έχουν οριστεί από τον χρήστη.

Βασικές μέθοδοι της ORMap		
Μέθοδος	**Παράμετροι**	**Επιστρέφει**
StartMappingORDBMStoXML()	Connection, String, String	-
getUserDefinedTypes()	DatabaseMetaData	-

Πίνακας π6.6: Βασικές μέθοδοι της ORMap

Στην κλάση ORMap την εναρκτήρια μέθοδο αποτελεί η " StartMappingORDBMStoXML" η οποία καλείται στην κλάση MapGUI και με τη σειρά της καλεί τις μεθόδους που χρειάζεται για την ανάκτηση των δεδομένων και μεταδεδομένων της αντικειμενοσχεσιακής βάσης στον αντίστοιχο XML έγγραφο. Η getUserDefinedTypes ήδη αναφέρθηκε στην προηγούμενη παράγραφο.

Παραπάνω παραλείπονται σκόπιμα όλες οι μέθοδοι που «εκτυπώνουν» τα παραγόμενα αρχεία. Οι μέθοδοι αυτές, μπορούν να βρεθούν στον κώδικα της εφαρμογής στο παράρτημα Ι.

6.6. Δυσκολίες και προβλήματα κατά την ανάπτυξη

Τα προβλήματα που αντιμετωπίσαμε, κατά την ανάπτυξη της εφαρμογής ήταν κυρίως τεχνικά και εστιάζονται στο αντικειμενοσχεσιακό μοντέλο.

Πιο συγκεκριμένα, η DB2 δεν συνεργαζόταν σωστά με την αρχική εφαρμογή που είχαμε αναπτύξει, η οποία είχε αναπτυχθεί με την μορφή εμφωλευμένων while βρόγχων. Κάθε ένας από αυτούς, διέτρεχε ένα ResultSet το οποίο περιείχε δεδομένα ή μεταδεδομένα μέχρι να φτάσει στην τελευταία γραμμή (μια γραμμή μετά το τέλος).

Ωστόσο, το πρόβλημα εστιαζόταν στην χρήση των ResultSet καθώς η DB2 απαγόρευε την μετακίνηση του κέρσορα σ' ένα ResultSet ακόμα και αν αυτό είχε οριστεί ως scrollable. Το πρόβλημα εκφραζόταν ως κλείσιμο του ResultSet πριν ολοκληρωθούν οι απαραίτητες λειτουργίες, και δεν μπορέσαμε να το λύσουμε κάνοντας commit όπου χρειαζόμασταν.

Ο επαναπρογραμματισμός της εφαρμογής, με άλλη λογική, αυτή της δημιουργίας ResultSet και του επαναληπτικού βρόγχου τους καθώς και των απαραίτητων μεταβλητών σε επίπεδο μεθόδων, έφερε το επιθυμητό αποτέλεσμα, ίσως με ενδεχόμενη αύξηση της πολυπλοκότητας της εφαρμογής.

Ωστόσο, αργότερα, συναντήσαμε προβλήματα με την ανάκτηση των UDT τύπων από την IBM DB2 βάση. Για παράδειγμα, η μέθοδος "getUDTs" της κλάσης DatabaseMetaData δεν επέστρεφε αποτελέσματα, ακόμα και χρησιμοποιώντας όλες τις δυνατές κατηγορίες τύπων (DISTINCT κτλ.) Το πρόβλημα, όμως, εστιάζεται στο JDBC και όχι στην DB2. Από την έρευνα που έγινε στο διαδίκτυο στην προσπάθεια να ξεπεραστεί αυτό το πρόβλημα, παρατηρήθηκε μια σύγχυση για το αν η κλάση DatabaseMetaData λειτουργεί σωστά.

Η λύση ήταν να ανακτήσουμε τους τύπους απευθείας με ερώτημα στη βάση, στους πίνακες συστήματος που δημιουργεί η IBM DB2. Το ίδιο έγινε και για την ανάκτηση των συναρτήσεων (functions).

6.7. Περίληψη κεφαλαίου

Στο παραπάνω κεφάλαιο, παρουσιάσαμε τις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη αυτής της εφαρμογής και για την επίτευξη του στόχου της.

Επίσης, παρουσιάσαμε αναλυτικά -βήμα προς βήμα- τους αλγόριθμους που υλοποιήθηκαν στην εφαρμογή μας και με βάση τους οποίους γίνεται η μετατροπή των βάσεων σε αντίστοιχες XML μορφής και παραθέσαμε δείγματα από τα παραγόμενα XML έγγραφα και σχήματα.

Παρουσιάσαμε τις κλάσεις που συνθέτουν την εφαρμογή καθώς και τον τρόπο λειτουργίας τους και τη διάταξη τους. Παραθέσαμε σε πίνακες και εξηγήσαμε για κάθε κλάση τις σημαντικότερες υλοποιημένες μεθόδους και αναφερθήκαμε στα προβλήματα που συναντήσαμε κατά την ανάπτυξη της εφαρμογής.

7. Συμπεράσματα

Τα συμπεράσματα που εξάγονται τόσο από την έρευνα που έγινε στα πλαίσια αυτής της πτυχιακής εργασίας, όσο και κατά την ανάπτυξη της εφαρμογής, εκφράζονται άμεσα ή έμμεσα σε όλη την εργασία.

Συνοψίζοντας, συμπεραίνουμε ότι όσο το διαδίκτυο αναπτύσσεται, και οι εφαρμογές πληθαίνουν, τόσο εντονότερη γίνεται η ανάγκη για ανταλλαγή πληροφορίας μεταξύ τους. Η XML είναι μια πολλά υποσχόμενη τεχνολογία και χρησιμοποιείται κατά κόρον από τις διαδικτυακές εφαρμογές και τα web services στον παγκόσμιο Ιστό για την ανταλλαγή δεδομένων.

Ωστόσο επειδή η πληροφορία είναι συνήθως αποθηκευμένη σε σχεσιακά αλλά και αντικειμενοσχεσιακά συστήματα, τόσο αυξάνεται η ανάγκη για την εύρεση εκείνων των αλγορίθμων που θα καταφέρουν να αναπαριστούν τα δεδομένα αυτών των συστημάτων σε XML δομή, με την ελάχιστη δυνατή απώλεια πληροφορίας.

Έχοντας υπ’ όψη την έρευνα που κάναμε, θεωρούμε ότι ο τομέας των αλγορίθμων μετατροπής δεδομένων από σχεσιακά και αντικειμενοσχεσιακά συστήματα σε XML, βρίσκεται ακόμα σε πρώιμο στάδιο και υπάρχουν μεγάλες δυνατότητες ανάπτυξης του.

Θεωρούμε, ότι σε αυτό τον τομέα, υπάρχει πρόσφορο έδαφος για ακαδημαϊκή έρευνα και εξέλιξη της γνώσης και θέλουμε να πιστεύουμε ότι η παρούσα πτυχιακή εργασία, βοηθάει προς αυτή την κατεύθυνση.

Βιβλιογραφία

[1] Database Metadata, Wikipedia

Τελευταία προσπέλαση: 2/8/2010

[2] NISO (2004), Understanding Metadata, NISO press, USA

[3] Relational Database Metadata, Wikipedia

Τελευταία προσπέλαση: 3/8/2010

[4] Taylor C. (1999), An Introduction to Metadata, University of Queensland, Queensland, Australia

[5] Parsian M. (2006), JDBC Metadata, MySQL, and Oracle Recipes: What Is JDBC Programming?, Apress

[6] IBM, What is a JDBC Driver

Τελευταία προσπέλαση: 25/8/2010

[7] Java Documentation, DriverManager Class

Τελευταία προσπέλαση: 28/8/2010

[8] Java Documentation, DatabaseMetaData

Τελευταία προσπέλαση: 3/9/2010

[9] Java Documentation, Connection Interface

Τελευταία προσπέλαση: 4/9/2010

[10] W3C (2006), XML 1.0 Recommendation : Origin and Goals

Τελευταία προσπέλαση: 9/9/2010

[11] W3Schools , XML Attributes

Τελευταία προσπέλαση: 9/9/2010

[12] W3C (2006) XML 1.0 Recommendation: Well-formed XML Documents

Τελευταία προσπέλαση: 10/9/2010

[13] Duta A.C., Barker K., Alhajj R. (2004), Proceedings of the 2004 ACM symposium on Applied computing : ConvRel: relationship conversion to XML nested structures, Nicosia, Cyprus, pp. 698 - 702

[14] Relax-NG : Introduction

Τελευταία προσπέλαση: 12/9/2010

[15] W3C Recommendation (2009), Namespaces in XML 1.0 (Third Edition)

Τελευταία προσπέλαση: 14/9/2010

[16] Ramakrishnan R. , Gehrke J. (1999), Database Management Systems v. 2, vol.1, McGraw-Hill, Inc. New York, NY, USA

[17] Copeland D.G. , Mason R.O. , McKenney J.L. (1995), IEEE Annals of the History of Computing, vol.17, Issue:3, Sabre: the development of information-based competence and execution of information-based competition, Univ. of Western Ontario, London, Ont, pp. 30-57

[18] HPG, Historic facts

Τελευταία προσπέλαση: 28/9/2010

[19] Dey D. and Sarkar S. , (1996), A probabilistic relational model and algebra. ACM Trans. Database Syst. 21, 3 (Sep. 1996), 339-369

[20] Paton N. , Gray P. (1988), Advances in Object-Oriented Database Systems : Identification of database objects by key ,Springer Berlin / Heidelberg

[21] Sabau G (2007), Comparison of RDBMS, OODBMS and ORDBMS, Inforec Association, Bucharest, Romania

[22] Rui Zhou , Chengfei Liu , Jianxin Li, Holistic constraint-preserving transformation from relational schema into XML schema, Proceedings of the 13th international conference on Database systems for advanced applications, March 19-21, 2008, New Delhi, India

[23] Liu, C., Vincent, M.W., Liu, J. (2006), Constraint preserving transformation from relational schema to XML schema, Springer Netherlands

[24] Shanmugasundaram J.,Shekita E., Barr R. ,Carey M., Lindsay B.,Pirahesh H. , Reinwald B. (2001), The VLDB Journal (2001) 10: Efficiently publishing relational data as XML documents, Springer Berlin / Heidelberg p.133-154

[25] Wang C. , Lo A. , Alhajj R. , Barker K. (2004), Reverse engineering based approach for transferring legacy relational databases into xml, Journal of Information and Organizational Sciences, Faculty of Organization and Informatics Varaždin, Croatia

[26] Fernandez M., Kadiyska Y. Suciu D. Morishima A. and Tan W.C.(2002), Bulletin of the Technical Committee (IEEE) In Data engineering : SilkRoute: A framework for publishing relational data in XML, ACM, p.12-19

[27] Carey M. ,Florescu D. ,Ives Z. ,Lu Y. ,Shanmugasundaram J., Shekita E., Subramanian S.(2000),XPERANTO: Publishing object-relational data as XML, WebDB (Informal Proceedings), p.105-110

[28] R Bourret (2005) , XML and Databases, Springer

[29] W3C (2006) XML 1.0 Recommendation

Τελευταία προσπέλαση: 15/10/2010

[30] IBM, Mapping of SQL and JDBC data types to XML data types

Τελευταία προσπέλαση: 16/10/2010

[31] Codd E.F. (1970), A Relational Model of Data for Large Shared Data Banks, ACM New York, USA

Παράρτημα

Τα παραγόμενα από τους αλγόριθμους XML σχήματα και έγγραφα

Το XML Schema που παράγεται από την σχεσιακή βάση δεδομένων.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="root">
            <xs:complexType>
                <xs:sequence>
<!--TABLE boats-->
                    <xs:element name="boats" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                        <xs:element name="bid" type="xs:int" minOccurs="1"/>
                        <xs:element name="bname" type="xs:string" minOccurs="0"/>
                        <xs:element name="color" type="xs:string" minOccurs="0"/>
                        </xs:sequence>
                    </xs:complexType>
                    </xs:element>
        <!--TABLE reserves-->
        <!-- bid in table reserves is primary key of table boats -->
        <!-- sid in table reserves is primary key of table sailors -->
                    <xs:element name="reserves" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                        <xs:element name="sid" type="xs:int" minOccurs="1"/>
                        <xs:element name="bid" type="xs:int" minOccurs="1"/>
                        <xs:element name="day" type="xs:date" minOccurs="1"/>
                        </xs:sequence>
                    </xs:complexType>
                    </xs:element>
        <!--TABLE sailors-->
                    <xs:element name="sailors" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                        <xs:element name="sid" type="xs:int" minOccurs="1"/>
                        <xs:element name="sname" type="xs:string" minOccurs="0"/>
                        <xs:element name="rating" type="xs:int" minOccurs="0"/>
                        <xs:element name="age" type="xs:float" minOccurs="0"/>
                        </xs:sequence>
                    </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
</xs:schema>
```

Το valid XML έγγραφο που παράγεται από την σχεσιακή βάση δεδομένων.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="R_schema.xsd">
<!--boats TABLE-->

<boats>
        <bid>101</bid>
        <bname>Interlake</bname>
        <color>Blue</color>
</boats>

<boats>
        <bid>102</bid>
        <bname>Interlake</bname>
        <color>Red</color>
</boats>

<boats>
        <bid>103</bid>
        <bname>Clipper</bname>
        <color>Green</color>
</boats>

<boats>
        <bid>104</bid>
        <bname>Marine</bname>
        <color>Red</color>
</boats>

<!--reserves TABLE-->

<reserves>
        <sid>22</sid>
        <bid>101</bid>
        <day>1998-10-10</day>
</reserves>

<reserves>
        <sid>22</sid>
        <bid>102</bid>
        <day>1998-10-10</day>
</reserves>

<reserves>
        <sid>22</sid>
        <bid>103</bid>
        <day>1998-10-08</day>
</reserves>

<reserves>
        <sid>22</sid>
        <bid>104</bid>
        <day>1998-10-07</day>
</reserves>

<reserves>
        <sid>31</sid>
        <bid>102</bid>
```

```
        <day>1998-11-10</day>
</reserves>

<reserves>
        <sid>31</sid>
        <bid>103</bid>
        <day>1998-11-06</day>
</reserves>

<reserves>
        <sid>31</sid>
        <bid>104</bid>
        <day>1998-11-12</day>
</reserves>

<reserves>
        <sid>64</sid>
        <bid>101</bid>
        <day>1998-09-05</day>
</reserves>

<reserves>
        <sid>64</sid>
        <bid>102</bid>
        <day>1998-09-08</day>
</reserves>

<reserves>
        <sid>74</sid>
        <bid>103</bid>
        <day>1998-09-08</day>
</reserves>

<!--sailors TABLE-->

<sailors>
        <sid>22</sid>
        <sname>Dustin</sname>
        <rating>7</rating>
        <age>45</age>
</sailors>

<sailors>
        <sid>29</sid>
        <sname>Brutus</sname>
        <rating>1</rating>
        <age>33</age>
</sailors>

<sailors>
        <sid>31</sid>
        <sname>Lubber</sname>
        <rating>8</rating>
        <age>55.5</age>
</sailors>

<sailors>
        <sid>32</sid>
        <sname>Andy</sname>
        <rating>8</rating>
```

```
        <age>25.5</age>
</sailors>

<sailors>
        <sid>58</sid>
        <sname>Rusty</sname>
        <rating>10</rating>
        <age>35</age>
</sailors>

<sailors>
        <sid>64</sid>
        <sname>Horatio</sname>
        <rating>7</rating>
        <age>35</age>
</sailors>

<sailors>
        <sid>71</sid>
        <sname>Zorba</sname>
        <rating>10</rating>
        <age>16</age>
</sailors>

<sailors>
        <sid>74</sid>
        <sname>Horatio</sname>
        <rating>9</rating>
        <age>40</age>
</sailors>

<sailors>
        <sid>85</sid>
        <sname>Art</sname>
        <rating>3</rating>
        <age>25.5</age>
</sailors>

<sailors>
        <sid>95</sid>
        <sname>Bob</sname>
        <rating>3</rating>
        <age>63.5</age>
</sailors>

</root>
```

Το XML Schema που παράγεται από την αντικειμενοσχεσιακή βάση δεδομένων.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="root">
                  <xs:complexType>
                        <xs:sequence>
<!--TYPED TABLE DROMOLOGIO is of type DROMOLOGIO_T -->

                <xs:element name="dromologio" maxOccurs="unbounded">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="afetiria" type="xs:string" minOccurs="0" />
                            <xs:element name="proorismos" type="xs:string" minOccurs="0" />
                            <xs:element name="hmera_dromologiou" type="xs:string" minOccurs="0" />
                            <xs:element name="ora_dromologiou" type="xs:string" minOccurs="0" />
                            <xs:element name="leoforeio" type="xs:string" minOccurs="0" />
                            <xs:element name="timi_eisitiriou" type="xs:int" minOccurs="0" />
                          </xs:sequence>
                          <xs:attribute name="OID" type="xs:string" use="required"/>
                        </xs:complexType>
                </xs:element>
<!--TYPED TABLE LEOFOREIO is of type LEOFOREIO_T -->

                <xs:element name="leoforeio" maxOccurs="unbounded">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="arithmos" type="xs:string" minOccurs="0" />
                            <xs:element name="edra" type="xs:string" minOccurs="0" />
                            <xs:element name="hmer_kataskeuhs" type="xs:date" minOccurs="0" />
                            <xs:element name="pliroma" type="xs:string" minOccurs="0" />
                            <xs:element name="xiliometra" type="xs:int" minOccurs="0" />
                          </xs:sequence>
                          <xs:attribute name="OID" type="xs:string" use="required"/>
                        </xs:complexType>
                </xs:element>
<!--TYPED TABLE ODIGOS is of type ODIGOS_T -->

                <xs:element name="odigos" maxOccurs="unbounded">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="onoma" type="xs:string" minOccurs="0" />
                            <xs:element name="eponumo" type="xs:string" minOccurs="0" />
                            <xs:element name="fulo" type="xs:string" minOccurs="0" />
                            <xs:element name="hlikia" type="xs:int" minOccurs="0" />
                            <xs:element name="dieuthinsi" type="xs:string" minOccurs="0" />
                            <xs:element name="hmer_proslipsis" type="xs:date" minOccurs="0" />
                            <xs:element name="typos" type="xs:string" minOccurs="0" />
                            <xs:element name="arithmos" type="xs:int" minOccurs="0" />
                          </xs:sequence>
                          <xs:attribute name="OID" type="xs:string" use="required"/>
                        </xs:complexType>
                </xs:element>
<!--TYPED TABLE PLIROMA is of type PLIROMA_T -->

                <xs:element name="pliroma" maxOccurs="unbounded">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="odigos" type="xs:string" minOccurs="0" />
                            <xs:element name="synodos" type="xs:string" minOccurs="0" />
```

```
                        </xs:sequence>
                        <xs:attribute name="OID" type="xs:string" use="required"/>
                    </xs:complexType>
            </xs:element>
<!--TYPED TABLE POLI is of type POLI_T -->

            <xs:element name="poli" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="onoma" type="xs:string" minOccurs="0" />
                        <xs:element name="xora" type="xs:string" minOccurs="0" />
                      </xs:sequence>
                      <xs:attribute name="OID" type="xs:string" use="required"/>
                    </xs:complexType>
            </xs:element>
<!--TYPED TABLE SYNODOS is of type SYNODOS_T -->

            <xs:element name="synodos" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="onoma" type="xs:string" minOccurs="0" />
                        <xs:element name="eponumo" type="xs:string" minOccurs="0" />
                        <xs:element name="fulo" type="xs:string" minOccurs="0" />
                        <xs:element name="hlikia" type="xs:int" minOccurs="0" />
                        <xs:element name="dieuthinsi" type="xs:string" minOccurs="0" />
                        <xs:element name="hmer_proslipsis" type="xs:date" minOccurs="0" />
                        <xs:element name="vathmos" type="xs:string" minOccurs="0" />
                      </xs:sequence>
                      <xs:attribute name="OID" type="xs:string" use="required"/>
                    </xs:complexType>
            </xs:element>
<!--TYPED TABLE YPALLILOS is of type YPALLILOS_T -->

            <xs:element name="ypallilos" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="onoma" type="xs:string" minOccurs="0" />
                        <xs:element name="eponumo" type="xs:string" minOccurs="0" />
                        <xs:element name="fulo" type="xs:string" minOccurs="0" />
                        <xs:element name="hlikia" type="xs:int" minOccurs="0" />
                        <xs:element name="dieuthinsi" type="xs:string" minOccurs="0" />
                        <xs:element name="hmer_proslipsis" type="xs:date" minOccurs="0" />
                      </xs:sequence>
                      <xs:attribute name="OID" type="xs:string" use="required"/>
                    </xs:complexType>
            </xs:element>

<!-- ****** A list of all the User Types ****** -->

<!--ARITHMOS_KUKLOFORIAS -->
<!--DROMOLOGIO_T -->
<!--LEOFOREIO_T -->
<!--ODIGOS_T -->
<!--PLIROMA_T -->
<!--POLI_T -->
<!--SYNODOS_T -->
<!--TYPOS_DIPLOMATOS -->
<!--VATHMOS_SUNODOU -->
<!--YPALLILOS_T -->
```

```
<!-- ****** A list of all the functions ****** -->

<!--< -->
<!--<= -->
<!--<> -->
<!--= -->
<!--> -->
<!-->= -->
<!--AFETIRIA -->
<!--ARITHMOS -->
<!--ARITHMOS_KUKLOFORIAS -->
<!--DIEUTHINSI -->
<!--DROMOLOGIO_T -->
<!--EDRA -->
<!--EPONUMO -->
<!--FULO -->
<!--HLIKIA -->
<!--HMERA_DROMOLOGIOU -->
<!--HMER_KATASKEUHS -->
<!--HMER_PROSLIPSIS -->
<!--INTEGER -->
<!--LEOFOREIO -->
<!--LEOFOREIO_T -->
<!--ODIGOS -->
<!--ODIGOS_T -->
<!--ONOMA -->
<!--ORA_DROMOLOGIOU -->
<!--PLIROMA -->
<!--PLIROMA_T -->
<!--POLI_T -->
<!--PROORISMOS -->
<!--SYNODOS -->
<!--SYNODOS_T -->
<!--TIMI_EISITIRIOU -->
<!--TYPOS -->
<!--TYPOS_DIPLOMATOS -->
<!--VARCHAR -->
<!--VATHMOS -->
<!--VATHMOS_SUNODOU -->
<!--XILIOMETRA -->
<!--XORA -->
<!--YPALLILOS_T -->
                                </xs:sequence>
                        </xs:complexType>
</xs:element>
</xs:schema>
```

Το valid XML έγγραφο που παράγεται από την αντικειμενοσχεσιακή βάση δεδομένων.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Object-Relational_schema.xsd">

<!--DROMOLOGIO TYPED TABLE-->

<dromologio OID="1">
        <afetiria>10</afetiria>
        <proorismos>20</proorismos>
        <hmera_dromologiou>Deutera</hmera_dromologiou>
        <ora_dromologiou>10:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="2">
        <afetiria>10</afetiria>
        <proorismos>20</proorismos>
        <hmera_dromologiou>Triti</hmera_dromologiou>
        <ora_dromologiou>10:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="3">
        <afetiria>10</afetiria>
        <proorismos>20</proorismos>
        <hmera_dromologiou>Tetarti</hmera_dromologiou>
        <ora_dromologiou>10:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="4">
        <afetiria>10</afetiria>
        <proorismos>20</proorismos>
        <hmera_dromologiou>Pemti</hmera_dromologiou>
        <ora_dromologiou>10:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="5">
        <afetiria>10</afetiria>
        <proorismos>20</proorismos>
        <hmera_dromologiou>Paraskeui</hmera_dromologiou>
        <ora_dromologiou>10:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="6">
        <afetiria>10</afetiria>
        <proorismos>20</proorismos>
        <hmera_dromologiou>Sabbato</hmera_dromologiou>
        <ora_dromologiou>10:00:00</ora_dromologiou>
```

```
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="7">
        <afetiria>10</afetiria>
        <proorismos>20</proorismos>
        <hmera_dromologiou>Kuriaki</hmera_dromologiou>
        <ora_dromologiou>10:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="11">
        <afetiria>20</afetiria>
        <proorismos>10</proorismos>
        <hmera_dromologiou>Deutera</hmera_dromologiou>
        <ora_dromologiou>20:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="12">
        <afetiria>20</afetiria>
        <proorismos>10</proorismos>
        <hmera_dromologiou>Triti</hmera_dromologiou>
        <ora_dromologiou>20:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="13">
        <afetiria>20</afetiria>
        <proorismos>10</proorismos>
        <hmera_dromologiou>Tetarti</hmera_dromologiou>
        <ora_dromologiou>20:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="14">
        <afetiria>20</afetiria>
        <proorismos>10</proorismos>
        <hmera_dromologiou>Pemti</hmera_dromologiou>
        <ora_dromologiou>20:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="15">
        <afetiria>20</afetiria>
        <proorismos>10</proorismos>
        <hmera_dromologiou>Paraskeui</hmera_dromologiou>
        <ora_dromologiou>20:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="16">
        <afetiria>20</afetiria>
```

```
        <proorismos>10</proorismos>
        <hmera_dromologiou>Sabbato</hmera_dromologiou>
        <ora_dromologiou>20:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="17">
        <afetiria>20</afetiria>
        <proorismos>10</proorismos>
        <hmera_dromologiou>Kuriaki</hmera_dromologiou>
        <ora_dromologiou>20:00:00</ora_dromologiou>
        <leoforeio>10</leoforeio>
        <timi_eisitiriou>50</timi_eisitiriou>
</dromologio>

<dromologio OID="21">
        <afetiria>30</afetiria>
        <proorismos>40</proorismos>
        <hmera_dromologiou>Kuriaki</hmera_dromologiou>
        <ora_dromologiou>15:00:00</ora_dromologiou>
        <leoforeio>20</leoforeio>
        <timi_eisitiriou>30</timi_eisitiriou>
</dromologio>

<dromologio OID="22">
        <afetiria>40</afetiria>
        <proorismos>30</proorismos>
        <hmera_dromologiou>Deutera</hmera_dromologiou>
        <ora_dromologiou>08:00:00</ora_dromologiou>
        <leoforeio>20</leoforeio>
        <timi_eisitiriou>30</timi_eisitiriou>
</dromologio>

<dromologio OID="31">
        <afetiria>10</afetiria>
        <proorismos>40</proorismos>
        <hmera_dromologiou>Triti</hmera_dromologiou>
        <ora_dromologiou>08:00:00</ora_dromologiou>
        <leoforeio>30</leoforeio>
        <timi_eisitiriou>10</timi_eisitiriou>
</dromologio>

<dromologio OID="32">
        <afetiria>40</afetiria>
        <proorismos>10</proorismos>
        <hmera_dromologiou>Triti</hmera_dromologiou>
        <ora_dromologiou>12:00:00</ora_dromologiou>
        <leoforeio>30</leoforeio>
        <timi_eisitiriou>10</timi_eisitiriou>
</dromologio>

<dromologio OID="33">
        <afetiria>10</afetiria>
        <proorismos>40</proorismos>
        <hmera_dromologiou>Tetarti</hmera_dromologiou>
        <ora_dromologiou>08:00:00</ora_dromologiou>
        <leoforeio>30</leoforeio>
        <timi_eisitiriou>10</timi_eisitiriou>
</dromologio>
```

```
<dromologio OID="34">
	<afetiria>40</afetiria>
	<proorismos>10</proorismos>
	<hmera_dromologiou>Tetarti</hmera_dromologiou>
	<ora_dromologiou>12:00:00</ora_dromologiou>
	<leoforeio>30</leoforeio>
	<timi_eisitiriou>10</timi_eisitiriou>
</dromologio>

<!--LEOFOREIO TYPED TABLE-->

<leoforeio OID="10">
	<arithmos>NHE1010</arithmos>
	<edra>10</edra>
	<hmer_kataskeuhs>2007-01-05</hmer_kataskeuhs>
	<pliroma>1</pliroma>
	<xiliometra>10000</xiliometra>
</leoforeio>

<leoforeio OID="20">
	<arithmos>NEE9990</arithmos>
	<edra>10</edra>
	<hmer_kataskeuhs>2006-01-06</hmer_kataskeuhs>
	<pliroma>2</pliroma>
	<xiliometra>50000</xiliometra>
</leoforeio>

<leoforeio OID="30">
	<arithmos>YOI7867</arithmos>
	<edra>20</edra>
	<hmer_kataskeuhs>2005-01-02</hmer_kataskeuhs>
	<pliroma>3</pliroma>
	<xiliometra>100000</xiliometra>
</leoforeio>

<leoforeio OID="40">
	<arithmos>PAY4532</arithmos>
	<edra>30</edra>
	<hmer_kataskeuhs>2004-01-09</hmer_kataskeuhs>
	<pliroma>4</pliroma>
	<xiliometra>150000</xiliometra>
</leoforeio>

<!--ODIGOS TYPED TABLE-->

<odigos OID="23">
	<onoma>Anna</onoma>
	<eponumo>Papasotiriou</eponumo>
	<fulo>F</fulo>
	<hlikia>32</hlikia>
	<dieuthinsi>Ermou 17</dieuthinsi>
	<hmer_proslipsis>2003-01-11</hmer_proslipsis>
	<typos>Erasitexniko</typos>
	<arithmos>4443233</arithmos>
</odigos>

<!--PLIROMA TYPED TABLE-->

<pliroma OID="1">
```

```
        <odigos>21</odigos>
        <synodos>14</synodos>
</pliroma>

<pliroma OID="2">
        <odigos>22</odigos>
        <synodos>11</synodos>
</pliroma>

<pliroma OID="3">
        <odigos>21</odigos>
        <synodos>12</synodos>
</pliroma>

<pliroma OID="4">
        <odigos>23</odigos>
        <synodos>13</synodos>
</pliroma>

<!--POLI TYPED TABLE-->

<poli OID="10">
        <onoma>Thessaloniki</onoma>
        <xora>Ellada</xora>
</poli>

<poli OID="20">
        <onoma>Athina</onoma>
        <xora>Ellada</xora>
</poli>

<poli OID="30">
        <onoma>Florina</onoma>
        <xora>Ellada</xora>
</poli>

<poli OID="40">
        <onoma>Larisa</onoma>
        <xora>Ellada</xora>
</poli>

<poli OID="50">
        <onoma>Volos</onoma>
        <xora>Ellada</xora>
</poli>

<poli OID="60">
        <onoma>Drama</onoma>
        <xora>Ellada</xora>
</poli>

<poli OID="110">
        <onoma>Sofia</onoma>
        <xora>Boulgaria</xora>
</poli>

<!--SYNODOS TYPED TABLE-->

<synodos OID="12">
        <onoma>Giorgios</onoma>
```

```
        <eponumo>Georgiadis</eponumo>
        <fulo>M</fulo>
        <hlikia>53</hlikia>
        <dieuthinsi>Serron 34</dieuthinsi>
        <hmer_proslipsis>1976-11-04</hmer_proslipsis>
        <vathmos>A</vathmos>
</synodos>

<synodos OID="13">
        <onoma>Sotiria</onoma>
        <eponumo>Bellou</eponumo>
        <fulo>F</fulo>
        <hlikia>38</hlikia>
        <dieuthinsi>Gravias 5</dieuthinsi>
        <hmer_proslipsis>1999-02-06</hmer_proslipsis>
        <vathmos>B</vathmos>
</synodos>

<synodos OID="14">
        <onoma>Sofia</onoma>
        <eponumo>Iatrou</eponumo>
        <fulo>F</fulo>
        <hlikia>41</hlikia>
        <dieuthinsi>Peramou 8</dieuthinsi>
        <hmer_proslipsis>1997-02-06</hmer_proslipsis>
        <vathmos>B</vathmos>
</synodos>

<!--YPALLILOS TYPED TABLE-->

<ypallilos OID="2">
        <onoma>Giorgios</onoma>
        <eponumo>Karampelas</eponumo>
        <fulo>M</fulo>
        <hlikia>40</hlikia>
        <dieuthinsi>Megalou Alexandrou 105</dieuthinsi>
        <hmer_proslipsis>2000-10-10</hmer_proslipsis>
</ypallilos>

<ypallilos OID="12">
        <onoma>Giorgios</onoma>
        <eponumo>Georgiadis</eponumo>
        <fulo>M</fulo>
        <hlikia>53</hlikia>
        <dieuthinsi>Serron 34</dieuthinsi>
        <hmer_proslipsis>1976-11-04</hmer_proslipsis>
</ypallilos>

<ypallilos OID="13">
        <onoma>Sotiria</onoma>
        <eponumo>Bellou</eponumo>
        <fulo>F</fulo>
        <hlikia>38</hlikia>
        <dieuthinsi>Gravias 5</dieuthinsi>
        <hmer_proslipsis>1999-02-06</hmer_proslipsis>
</ypallilos>

<ypallilos OID="14">
        <onoma>Sofia</onoma>
        <eponumo>Iatrou</eponumo>
```

```
        <fulo>F</fulo>
        <hlikia>41</hlikia>
        <dieuthinsi>Peramou 8</dieuthinsi>
        <hmer_proslipsis>1997-02-06</hmer_proslipsis>
</ypallilos>

<ypallilos OID="23">
        <onoma>Anna</onoma>
        <eponumo>Papasotiriou</eponumo>
        <fulo>F</fulo>
        <hlikia>32</hlikia>
        <dieuthinsi>Ermou 17</dieuthinsi>
        <hmer_proslipsis>2003-01-11</hmer_proslipsis>
</ypallilos>

</root>
```

Ο κώδικας της εφαρμογής (Java)

Ο κώδικας Java για την κλάση **MapGUI** είναι :

```
import java.io.*;
import java.sql.*;
/**
 * @author Michail Mpellas
 * email: mmpell@it.teithe.gr
 * class: MapGUI
 * @version 1.0
 */
public class MapGUI extends javax.swing.JFrame {

    /** Creates new form MapGUI */
    String usrname = "";
      char[] pswd = null;
      String url = "";
      String port = "";
      String dbName = "";
      String driverClassName = "";
      String dbms = "";
      String loc = "";
      String fname = "";
      Connection dbcon = null;
      String password ="";
      String conString="";

    public MapGUI() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jOptionPane1 = new javax.swing.JOptionPane();
        jFileChooser1 = new javax.swing.JFileChooser();
        jButton1 = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        jLabel5 = new javax.swing.JLabel();
        jComboBox1 = new javax.swing.JComboBox();
        jPanel2 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jTextField3 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jTextField1 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jPasswordField1 = new javax.swing.JPasswordField();
        jLabel8 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
```

```
        jPanel3 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jTextField5 = new javax.swing.JTextField();
        jButton2 = new javax.swing.JButton();
        jLabel7 = new javax.swing.JLabel();
        jTextField6 = new javax.swing.JTextField();

        jFileChooser1.setFileSelectionMode(javax.swing.JFileChooser.DIRECTORIES_ONLY);
        jFileChooser1.setForeground(java.awt.Color.white);
        jFileChooser1.setDragEnabled(true);

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(204, 204, 255));
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setForeground(java.awt.Color.lightGray);
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowOpened(java.awt.event.WindowEvent evt) {
                formWindowOpened(evt);
            }
        });

        jButton1.setText("Start Mapping");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Database Properties"));
        jPanel1.setName("dpPanel"); // NOI18N

        jLabel5.setText("Database Type");

        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[]
{ "PostGreSQL", "DB2" }));
        jComboBox1.setName("dbSelector"); // NOI18N
        jComboBox1.addItemListener(new java.awt.event.ItemListener() {
            public void itemStateChanged(java.awt.event.ItemEvent evt) {
                jComboBox1ItemStateChanged(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel5,      javax.swing.GroupLayout.PREFERRED_SIZE,      300,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(5, 5, 5))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jComboBox1, 0, 291, Short.MAX_VALUE)
                        .addGap(284, 284, 284)))
                .addContainerGap())
        );
```

```
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addComponent(jLabel5)
                .addGap(6, 6, 6)
                .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(60, Short.MAX_VALUE))
        );

        jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Connection & Login
Details"));
        jPanel2.setName("cldPanel"); // NOI18N

        jLabel4.setText("Host:");

        jLabel6.setText("Port:");

        jTextField3.setText("192.168.16.99");
        jTextField3.setName("hostField"); // NOI18N

        jTextField4.setName("portField"); // NOI18N

        jTextField1.setText("DB400");
        jTextField1.setName("usrnameField"); // NOI18N

        jLabel3.setText("Password:");

        jLabel2.setText("Username:");

        jPasswordField1.setText("66G68");
        jPasswordField1.setName("passwdField"); // NOI18N

        jLabel8.setText("Database Name:");

        jTextField2.setText("DB2LAB");
        jTextField2.setName("dbnameField"); // NOI18N

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                            .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE, 269,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 220,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 125,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel2)))
                    .addGroup(jPanel2Layout.createSequentialGroup()
```

```
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                    .addComponent(jLabel6)
                    .addComponent(jTextField4,  javax.swing.GroupLayout.PREFERRED_SIZE,  84,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(35, 35, 35)
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                    .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE, 150,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel8))
                .addGap(18, 18, 18)
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)
                    .addComponent(jPasswordField1)
                    .addComponent(jLabel3,      javax.swing.GroupLayout.DEFAULT_SIZE,      125,
Short.MAX_VALUE))))
            .addContainerGap(163, Short.MAX_VALUE))
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRA
ILING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addComponent(jLabel4)
                        .addGap(18, 18, 18)
                        .addComponent(jTextField3,            javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addComponent(jLabel2)
                        .addGap(18, 18, 18)
                        .addComponent(jTextField1,            javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRA
ILING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addComponent(jLabel3)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jPasswordField1,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                            .addComponent(jLabel6)
                            .addComponent(jLabel8))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                            .addComponent(jTextField4,        javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField2,        javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addContainerGap(14, Short.MAX_VALUE))
        );

        jPanel3.setBorder(javax.swing.BorderFactory.createTitledBorder("Export Details"));
        jPanel3.setName("edPanel"); // NOI18N
```

```
        jLabel1.setText("Export Location:");

        jTextField5.setText("C:\\Users\\MiXaLis\\Desktop");
        jTextField5.setName("exportField"); // NOI18N

        jButton2.setText("Browse..");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jLabel7.setText("Export Filename (no invalid characters allowed):");

        jTextField6.setName("filenameField"); // NOI18N

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
                    .addGroup(jPanel3Layout.createSequentialGroup()
                        .addComponent(jTextField5,   javax.swing.GroupLayout.PREFERRED_SIZE,   266,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(jButton2,     javax.swing.GroupLayout.PREFERRED_SIZE,     107,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(jLabel1)
                    .addComponent(jLabel7)
                    .addComponent(jTextField6,     javax.swing.GroupLayout.PREFERRED_SIZE,     147,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(184, Short.MAX_VALUE))
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                    .addComponent(jTextField5,                javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton2))
                .addGap(18, 18, 18)
                .addComponent(jLabel7)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jTextField6,                javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(57, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
```

```
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
               .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                  .addGap(20, 20, 20)
                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                     .addComponent(jPanel2,                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                     .addComponent(jPanel1,                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                     .addComponent(jPanel3,                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
               .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                  .addContainerGap(518, Short.MAX_VALUE)
                  .addComponent(jButton1)))
            .addContainerGap())
      );
      layout.setVerticalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(layout.createSequentialGroup()
            .addGap(21, 21, 21)
            .addComponent(jPanel1,                   javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jPanel2,                   javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(jPanel3,                   javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,          21,
Short.MAX_VALUE)
            .addComponent(jButton1)
            .addGap(40, 40, 40))
      );

      pack();
   }// </editor-fold>


   //The button that performs the map "Start Mapping"
   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

      //get all the values from their input fields
      dbms = jComboBox1.getSelectedItem().toString();
      usrname = this.jTextField1.getText();
      pswd = this.jPasswordField1.getPassword();
      url = this.jTextField3.getText();
      port = this.jTextField4.getText();
      dbName = this.jTextField2.getText();
      loc = jTextField5.getText();
      fname = jTextField6.getText();

      //create the connection string
      conString += url + ":" + port + "/" + dbName;

      //create the password
      for(int i=0; i<pswd.length; i++){
         password+= pswd[i];
      }
```

```
      //check for the selected DBMS
      if(dbms.equals("PostGreSQL")){
         driverClassName = "org.postgresql.Driver";
      } else if(dbms.equals("DB2")){
         driverClassName = "com.ibm.db2.jcc.DB2Driver";
      }
//
//       System.out.println(usrname);
//       System.out.println(password);
//       System.out.println(conString);
//
//
      try{
         //if the dbms is PostGreSQL
         if(dbms.equals("PostGreSQL")){

         //create the instances of their respective classes
         RelMap rl = new RelMap(conString,usrname,password);
         RelSchema rs = new RelSchema(conString,usrname,password);

         //create the necessary connections and perform the mapping
         dbcon = rl.initConnection(driverClassName);
         rl.StartMappingRDBMStoXML(dbcon, loc, fname);
         rl.closeConnection();

         dbcon = rs.initConnection(driverClassName);
         rs.createRelXMLSchema(dbcon, loc, fname);
         rs.closeConnection();

      } else if(dbms.equals("DB2")){   //if the dbms is DB2

         //create the instances of their respective classes
         ORMap orm = new ORMap(conString,usrname,password);
         ORSchema ors = new ORSchema(conString,usrname,password);

         //create the necessary connections and perform the mapping
         dbcon = orm.initConnection(driverClassName);
         orm.StartMappingORDBMStoXML(dbcon, loc, fname);
         orm.closeConnection();

         dbcon = ors.initConnection(driverClassName);
         ors.createORSchema(dbcon, loc, fname);
         ors.closeConnection();
      }
      clearPassword();  //clear the password for safety reasons
      nullifyValues(); //clear all the variable values
      checkDBMS(); //check for the selected DBMS
      }catch(Exception e){
         jOptionPane1.showMessageDialog(this,e.toString(),"Error
",jOptionPane1.ERROR_MESSAGE);
         nullifyValues();
      }
   }

   //The button that implements the Folder Browsing "Browse.."
   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
      int returnVal = this.jFileChooser1.showDialog(this, "Select");// TODO add your handling code
here:
      if (returnVal == jFileChooser1.APPROVE_OPTION) {
```

```
        File file = jFileChooser1.getSelectedFile();
        this.jTextField5.setText(file.getPath());
    }
}

//when the windows is opened check the selected
private void formWindowOpened(java.awt.event.WindowEvent evt) {
  checkDBMS();
}

//if the user changes the selected dbms
private void jComboBox1ItemStateChanged(java.awt.event.ItemEvent evt) {
    purgeFields(); //clear the field values
    checkDBMS(); //check what he had selected
}

/**
* @param args the command line arguments
*/
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new MapGUI().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JComboBox jComboBox1;
private javax.swing.JFileChooser jFileChooser1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JOptionPane jOptionPane1;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPasswordField jPasswordField1;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JTextField jTextField4;
private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
// End of variables declaration


//get the selected DBMS
private void checkDBMS(){

    dbms = jComboBox1.getSelectedItem().toString();

    if(dbms.equals("PostGreSQL")){
```

```
            jTextField4.setText("5432");
            conString = "jdbc:postgresql://";
        } else if(dbms.equals("DB2")){
            jTextField4.setText("50001");
            conString = "jdbc:db2://";
        }

    }

    //clear field values
    private void purgeFields(){
        jTextField1.setText("");
        jTextField2.setText("");
        jTextField3.setText("");
        jPasswordField1.setText("");
        jTextField6.setText("");
    }

    //clear the password for safety reasons
    private void clearPassword(){
        if(password!=null || pswd!=null){
        password = "";
        for(int i=0; i<pswd.length; i++)
            pswd[i] = 0;
        }
    }

    //clear the variable values
    private void nullifyValues(){
        url = "";
        port = "";
        dbName = "";
        driverClassName = "";
        //-dbms = "";
        loc = "";
        fname = "";
        password ="";
        conString="";
    }

}
```

Ο κώδικας Java για την κλάση **RelSchema** είναι:

```
import java.sql.*;
import java.util.Vector;
import java.io.*;
/**
 * @author Michail Mpellas
 * email: mmpell@it.teithe.gr
 * class: RelSchema
 * @version 1.0
 */
public class RelSchema {

   private String     url ="";
   static String     driverClassName ="";
   private String     username ="";
   private String     passwd ="";
   private Connection dbConnection    = null;
   private Statement  statement       = null;
   private ResultSet rs               = null;
   private PrintStream schema         = null;
   private Vector<String> tableName = new Vector<String>();
   private Vector<String> tableType = new Vector<String>();
   private Vector<String> tablesPK = new Vector<String>();
   private Vector<String> tablesFK = new Vector<String>();
   private Vector<String> tablesFKRef = new Vector<String>();
   private Vector<String> columnName = new Vector<String>();
   private Vector<String> columnType = new Vector<String>();



   public RelSchema(String _url,String _username,String _password){
      this.url = _url;
      this.username = _username;
      this.passwd = _password;
   }

   //Performs all the necessary tasks to perform the mapping of an RDBMS schema to XML Schema
   public void createRelXMLSchema(Connection dbConnection,String location, String fileName) throws Exception{
```

```
        //get the connection, database metadata and the tables of the database
        Connection con = dbConnection;
        DatabaseMetaData dmd = getDBMetadata(con);
        getTables(dmd);

        //create new print Stream to write to a file
        try{
            schema = new PrintStream(new FileOutputStream(location + "/" + fileName +
"_schema.xsd"));
        }catch(IOException ioe){System.out.println(ioe.toString());}
        printSchemaDeclaration(schema);

          //for every table print an xml comment with its table name and table type, get its data and
columns and print them out
          for(int i=0; i<tableName.size(); i++){
            // System.out.println(tableName.elementAt(i));
             schema.println("<!--" + tableType.elementAt(i) +" " + tableName.elementAt(i) + "-->");
             ResultSet data = getRecords(tableName.elementAt(i),statement);
             getPKeys(dmd,username,tableName.elementAt(i));

             getColumnsForEachTable(dmd,data,username,tableName.elementAt(i),schema);
          }//end for
//        for(int j=0; j<tablesFKRef.size(); j++){
//           System.out.println("tablesFKREF " + j + " " + tablesFKRef.elementAt(j));
//        }

        printSchemaRootEnd(schema);
//closeConnection();

    }//end main

    //Initiate connection
    public Connection initConnection(String driverClassName) throws ClassNotFoundException,
SQLException{
    Class.forName (driverClassName);
        dbConnection = DriverManager.getConnection (url, username, passwd);
        statement    = dbConnection.createStatement();
        System.out.println("Connection Initiated...\n Drivers Loaded: " + driverClassName);
        return dbConnection;
    }
```

```
   //close the connection
   public void closeConnection() throws SQLException{
      System.out.println("  ==>Connection closed on User's Request\n");
      schema.close();
      statement.close();
      dbConnection.close();
   }

   //get the tables of the database
   private void getTables(DatabaseMetaData dbmd) throws SQLException{
      String [] tableTypes = {"TABLE"};

        //Getting the db tables
         rs = dbmd.getTables("%", username, "%", tableTypes);
        while (rs.next()) {
           try{
                tableName.add(rs.getString("TABLE_NAME"));
                tableType.add(rs.getString("TABLE_TYPE"));

             }catch(com.ibm.db2.jcc.am.SqlException pse){System.out.println(pse.toString());}
        }
   }//end gateTables

   //get the primary keys of a table
   private void getPKeys(DatabaseMetaData dbmd,String username,String tableName) throws
SQLException{
      tablesPK.clear();
      ResultSet pkrs = dbmd.getPrimaryKeys("%", username , tableName);
      while(pkrs.next()){
        tablesPK.add(pkrs.getString("COLUMN_NAME"));
        //System.out.println("PK: " + pkrs.getString("COLUMN_NAME"));
       //    System.out.println("*****************************\nTABLE : "  +tableName  +"\nPK:"  +
trimWhite(pkrs.getString("COLUMN_NAME"))+"\n*************************\n");
      }
      //System.out.println("PK SIZE :" + tablesPK.size());
   }

   //check if a column name corresponds to a primary key
   private boolean isPKey(String ColumnName){
```

```
        boolean isPK = false;
        for(int i=0; i<tablesPK.size(); i++){
            if(tablesPK.elementAt(i).compareToIgnoreCase(ColumnName)==0)
                isPK = true;
        }
        return isPK;
    }//end isPKey

    //get the foreign/imported keys of a table
    private void getFKeys(DatabaseMetaData dbmd,String username,String tableName) throws
SQLException{
        tablesFK.clear();
        ResultSet fkrs = dbmd.getImportedKeys("%", username , tableName);
        while(fkrs.next()){
            tablesFK.add(fkrs.getString("FKCOLUMN_NAME"));
            tablesFKRef.add(fkrs.getString("PKTABLE_NAME"));
            System.out.println(fkrs.getString("PKCOLUMN_NAME"));
            schema.println("<!-- " + fkrs.getString("FKCOLUMN_NAME") + " in table "+ tableName + " is
primary key of table " + fkrs.getString("PKTABLE_NAME") + " -->");
            //System.out.println("*****************************\nTABLE : " +tableName +"\nFFFFK:" +
trimWhite(fkrs.getString("FKCOLUMN_NAME"))+"\n*************************\n");
        }
        System.out.println("FK SIZE :" + tablesFK.size());
    }//end getFKeys

    //check if a column name corresponds to a foreign/imported key
    private boolean isFKey(String ColumnName){
        boolean isFK = false;
        for(int i=0; i<tablesFK.size(); i++){
            if(tablesFK.elementAt(i).compareToIgnoreCase(ColumnName)==0)
                isFK = true;
        }
        //System.out.println("RES IS**fk " + isFK + " IN " + ColumnName);//
        return isFK;
    }//end isFKey

    //get the table records in a resultSet
    private ResultSet getRecords(String tableName, Statement statement) throws SQLException{
        String sqlQuery = "";
        sqlQuery = "SELECT * FROM " + tableName;
```

```
        ResultSet dataRS = statement.executeQuery(sqlQuery);
        return dataRS;
    }

    //Getting the database MetaData
    private DatabaseMetaData getDBMetadata(Connection dbConnection) throws SQLException{
        DatabaseMetaData dbmd = dbConnection.getMetaData();
        return dbmd;
    }

    //Getting the table metadata in a resultset MetaData
    private ResultSetMetaData getRecordsMetaData(ResultSet dataRS) throws SQLException{
        ResultSetMetaData rsmd = dataRS.getMetaData();
        return rsmd;
    }

    //get the columns, the data, and print them all out accordingly
    private void getColumnsForEachTable(DatabaseMetaData dbmd,ResultSet dataRS, String
username, String tableName, PrintStream schema) throws SQLException{
            ResultSetMetaData rsmd = null;
            //clear vectors before filling them out again
            columnName.clear();
            tablesFK.clear();
            tablesFKRef.clear();
            rsmd = getRecordsMetaData(dataRS);
            ResultSet colrs = dbmd.getColumns("%", username, tableName, "%"); //get table columns
            getFKeys(dbmd,username,tableName); //get table foreign/imported keys
               while (colrs.next()) {
                  columnName.add(colrs.getString("COLUMN_NAME")); //add column names to a
vector object
                  }

               printSchemaTableStart(tableName,schema);
               columnType.clear();

                  //for every column
                  for(int j=1; j<=rsmd.getColumnCount(); j++){
                     columnType.add(getXMLType(rsmd.getColumnType(j))); //get its corresponding
XML type
```

```
                    //primary/foreign key/ element check and print accordingly
                    if(isPKey(columnName.elementAt(j-1)) && !isFKey(columnName.elementAt(j-1))){
                       printSchemaElement(columnName.elementAt(j-1),columnType.elementAt(j-
1),true,false,false,j-1,schema);
                    }else  if(isFKey(columnName.elementAt(j-1))  &&  isPKey(columnName.elementAt(j-
1))){
                       printSchemaElement(columnName.elementAt(j-1),columnType.elementAt(j-
1),false,true,false,j-1,schema);
                    }else{
                       printSchemaElement(columnName.elementAt(j-1),columnType.elementAt(j-
1),false,false,true,j-1,schema);
                    }
                  }//end for j
              printSchemaTableEnd(schema);
}//end getColumnsForEachTable



   public static String trimWhite(String token) //clear all white space before and after of the input
string
   {
      String trimmed="";
         try{
              trimmed = token.replaceAll("\\s+$", "");
            }
         catch(NullPointerException np){ return "";}

      return trimmed;
    } //end trimWhite


   private void printSchemaDeclaration(PrintStream schema){
      schema.println("<?xml version=" + "" + "1.0" +"" + " encoding=" +"" + "utf-8" +"" + "?>");

      schema.println("<xs:schema" + " xmlns:xs=" + "" + "http://www.w3.org/2001/XMLSchema" + ""
+ ">");

      schema.println("\t<xs:element name=" + "" + "root" + "" + ">");
      schema.println("\t\t<xs:complexType>");
      schema.println("\t\t\t<xs:sequence>");
```

```
    }

    private void printSchemaTableStart(String tableName,PrintStream schema){
        schema.println("\t\t\t\t<xs:element name=" + "" + tableName.toLowerCase() + ""+ "
maxOccurs=" + "" + "unbounded" + "" + ">");
        schema.println("\t\t\t\t\t<xs:complexType>");
        schema.println("\t\t\t\t\t\t<xs:sequence>");
    }

    private void printSchemaTableEnd(PrintStream schema){
        schema.println("\t\t\t\t\t\t</xs:sequence>");
        //schema.println("\t\t\t\t<xs:attribute name=" + "" + columnName.elementAt(pkIndex).toLowerCase() + "" + " type=" + "" + "xs:" + columnType.elementAt(pkIndex) + "" + " use=" + "" + "required" + "" + "/>");
        schema.println("\t\t\t\t\t</xs:complexType>");
        schema.println("\t\t\t\t</xs:element>");
    }

    private void printSchemaElement(String columnName, String columnType, boolean isPK,boolean isFK,boolean isElem,int index, PrintStream schema){
        if(isPK){
            schema.println("\t\t\t\t\t\t<xs:element name=" + "" + columnName.toLowerCase() + ""
                    + " type=" + "" +"xs:" + columnType + ""
                        + " minOccurs=" + "" +"1"+ ""+ "/>");
        }else if(isFK){
            schema.println("\t\t\t\t\t\t<xs:element name=" + "" + columnName.toLowerCase() + ""
                    + " type=" + "" +"xs:" + columnType + ""
                        + " minOccurs=" + "" +"1"+ ""+ "/>");
        }else if(isElem){
            schema.println("\t\t\t\t\t\t<xs:element name=" + "" + columnName.toLowerCase() + "" + " type=" + "" +"xs:" + columnType + "" + " minOccurs=" + "" +"0"+ ""+ "/>");
        }
    }

    private void printSchemaRootEnd(PrintStream schema){
        schema.println("\t\t\t</xs:sequence>");
        schema.println("\t\t</xs:complexType>");
        schema.println("\t</xs:element>");
        schema.println("</xs:schema>");
    }
```

```
    // get the XML type that corresponds to the input SQL type
    private String getXMLType(int type) {

            switch(type){
               case Types.ARRAY: return "string";
               case Types.BIGINT: return "long";
               case Types.BINARY: return "base64Binary";
               case Types.BIT: return "short";
               case Types.BLOB: return "base64Binary";
               case Types.BOOLEAN: return "boolean";
               case Types.CHAR: return "string";
               case Types.CLOB: return "string";
               case Types.DATALINK: return "anyURI";
               case Types.DATE: return "date";
               case Types.DECIMAL: return "decimal";
               case Types.DOUBLE: return "double";
               case Types.FLOAT: return "float";
               case Types.INTEGER: return "int";
               case Types.LONGVARBINARY: return "base64Binary";
               case Types.LONGVARCHAR: return "string";
               case Types.NUMERIC: return "decimal";
               case Types.REAL: return "float";
               case Types.SMALLINT: return "short";
               //case Types.TIMESTAMP: return "time";
               case Types.TIMESTAMP: return "dateTime";
               case Types.TINYINT: return "short";
               case Types.VARBINARY: return "base64Binary";
               case Types.VARCHAR: return "string";
               case Types.DISTINCT: return "string";
               case Types.JAVA_OBJECT: return "string";
               case Types.NULL: return "string";
               case Types.OTHER: return "string";
               case Types.REF: return "string";
               case Types.STRUCT: return "string";
               case 100: return "anyType";

            }
      return "string";
    }//end getSqlType
}//end class RelSchema
```

Ο κώδικας Java για την κλάση **RelMap** είναι:

```
import java.sql.*;
import java.util.Vector;
import java.io.*;
/**
 * @author Michail Mpellas
 * email: mmpell@it.teithe.gr
 * class: RelMap
 * @version 1.0
 */
public class RelMap {



   private String     url ="";
   static String     driverClassName="";
   private String     username="";
   private String     passwd ="";
   private Connection dbConnection    = null;
   private Statement  statement       = null;
   private ResultSet rs               = null;
   private PrintStream ps             = null;
   private Vector<String> tableName = new Vector<String>();
   private Vector<String> tableType = new Vector<String>();
   private Vector<String> tablesPK = new Vector<String>();
   private Vector<String> tablesFK = new Vector<String>();
   private Vector<String> columnName = new Vector<String>();
   private Vector<String> columnType = new Vector<String>();

   public RelMap(String _url,String _username,String _password){
      this.url = _url;
      this.username = _username;
      this.passwd = _password;
   }



   //Performs all the necessary tasks to perform the mapping of an RDBMS to XML
   public void StartMappingRDBMStoXML(Connection dbConnection,String location, String
fileName) throws Exception{
      try{
      //get the connection, database metadata and the tables
```

```
        Connection con = dbConnection;
        DatabaseMetaData dmd = getDBMetadata(con);
        getTables(dmd);

        //create new print Stream to write to a file
        try{
            ps = new PrintStream(new FileOutputStream(location + "/" + fileName + ".xml"));
        }catch(IOException ioe){System.out.println(ioe.toString());}
        printXmlDeclaration(ps, fileName);

        //for every table print an xml comment with its name and type, get its data, primary, foreign
keys and columns and print them out
        for(int i=0; i<tableName.size(); i++){
            ps.println("<!--" + tableName.elementAt(i) +" " + tableType.elementAt(i) + "-->\n");

            ResultSet data = getRecords(tableName.elementAt(i),statement);
            getPKeys(dmd,username,tableName.elementAt(i));
            getFKeys(dmd,username,tableName.elementAt(i));
            getColumnsForEachTable(dmd,data,username,tableName.elementAt(i),ps);
        }//end for

printXmlRootEnd(ps);

//closeConnection(statement,con);
        }catch(Exception e){System.out.println(e.toString());}

        }

    //Initiate connection
    public Connection initConnection(String driverClassName) throws ClassNotFoundException,
SQLException{
        try{
    Class.forName (driverClassName);
        dbConnection = DriverManager.getConnection (url, username, passwd);
        statement    = dbConnection.createStatement();
        System.out.println("==>Connection Initiated\n ==>Drivers Loaded: " + driverClassName);
        }catch(SQLException s){System.out.println(s.toString() + "\nDRIVER:" +driverClassName +
"username: " + username + "password" + passwd);}
        return dbConnection;
    }
```

```
  //close connection
  public void closeConnection() throws SQLException{
     System.out.println("  ==>Connection closed on User's Request\n");
     ps.close();
     statement.close();
     dbConnection.close();
  }

  //get the tables of the database
  private void getTables(DatabaseMetaData dbmd) throws SQLException{
     String [] tableTypes = {"TABLE", "TYPED TABLE"};

       //Getting the db tables
        rs = dbmd.getTables("%", username, "%", tableTypes);
       while (rs.next()) {
          try{
            // System.out.println(rs.getString("TABLE_NAME"));
               tableName.add(rs.getString("TABLE_NAME"));
               tableType.add(rs.getString("TABLE_TYPE"));
           }catch(com.ibm.db2.jcc.am.SqlException pse){System.out.println(pse.toString());}
       }
  }//end Tables

  //get the primary keys of a table
  private void getPKeys(DatabaseMetaData dbmd,String username,String tableName) throws
SQLException{
     tablesPK.clear();
     ResultSet pkrs = dbmd.getPrimaryKeys("%", username , tableName);
     while(pkrs.next()){
       tablesPK.add(pkrs.getString("COLUMN_NAME"));
     }
  }

  //check if a column name corresponds to a primary key
  private boolean isPKey(String ColumnName){
     boolean isPK = false;
     for(int i=1; i<=tablesPK.size(); i++){
        if(tablesPK.elementAt(i-1).contains(ColumnName))
           isPK = true;
```

```
        }
        System.out.println("RES IS*PK " + isPK + " IN " + ColumnName);
        return isPK;
    }//end isPKey

    //get the foreign/imported keys of a table
    private void getFKeys(DatabaseMetaData dbmd,String username,String tableName) throws
SQLException{
        tablesFK.clear();
        ResultSet fkrs = dbmd.getImportedKeys("%", username , tableName);
        while(fkrs.next()){
          tablesFK.add(fkrs.getString("FKCOLUMN_NAME"));
          }
    }//end getFKeys

    //check if a column name corresponds to a foreign key
    private boolean isFKey(String ColumnName){
        boolean isFK = false;
        for(int i=1; i<=tablesFK.size(); i++){
            if(tablesFK.elementAt(i-1).contains(ColumnName))
                isFK = true;
        }
        return isFK;
    }//end isFKey

    //get the table records in a resultSet
    private ResultSet getRecords(String tableName, Statement statement) throws SQLException{
        String sqlQuery = "";
        sqlQuery = "SELECT * FROM " + tableName;
        ResultSet dataRS = statement.executeQuery(sqlQuery);
        return dataRS;
    }

    //Getting the database MetaData
    private DatabaseMetaData getDBMetadata(Connection dbConnection) throws SQLException{
        DatabaseMetaData dbmd = dbConnection.getMetaData();
            return dbmd;
    }

    //Getting the table metadata in a resultset MetaData
```

```
   private ResultSetMetaData getRecordsMetaData(ResultSet dataRS) throws SQLException{
      ResultSetMetaData rsmd = dataRS.getMetaData();
         return rsmd;
   }

   //get the columns, the data, and print them all out accordingly
   private void getColumnsForEachTable(DatabaseMetaData dbmd,ResultSet dataRS, String
username, String tableName, PrintStream ps) throws SQLException{

      while(dataRS.next()){ //for every table record
         columnName.clear(); //clear the columnNames every time

         ResultSetMetaData rsmd = getRecordsMetaData(dataRS);
         ResultSet colrs = dbmd.getColumns("%", username, tableName, "%");

            while (colrs.next()) {
               columnName.add(colrs.getString("COLUMN_NAME")); //add column names to a
vector object
               }

         printXmlTableStart(tableName,ps);
            for(int j=1; j<=rsmd.getColumnCount(); j++){ //for every column
               columnType.add(rsmd.getColumnTypeName(j)); //add its type to a vector
               String elementData = trimWhite(dataRS.getString(columnName.elementAt(j-1))); //get
the data
                  printXmlElement(elementData,columnName.elementAt(j-1),ps); //and print them out
            }//end for
         printXmlTableEnd(tableName,ps);
      }//end while
}//end getColumnsForEachTable

   public static String trimWhite(String token) //clear all white space before and after of the input
string
   {
      String trimmed="";
         try{
               trimmed = token.replaceAll("\\s+$", "");
            }
         catch(NullPointerException np){ return "";}
```

```
        return trimmed;
      } //end trimWhite

   private void printXmlDeclaration(PrintStream ps, String fileName){
        ps.println("<?xml version=" + "" + "1.0" +"" + " encoding=" +"" + "utf-8" +"" + " standalone="
+"" + "no" +"" +"?>");

        ps.println("<root " + "xmlns:xsi=" + "" + "http://www.w3.org/2001/XMLSchema-instance" + "" +
                    " xsi:noNamespaceSchemaLocation=" + ""  + fileName +"_schema.xsd" + "" + ">");
   }

   private void printXmlTableStart(String tableName,PrintStream ps){
      ps.println("<" + tableName.toLowerCase()+ ">");
   }

   private void printXmlTableEnd(String tableName,PrintStream ps){
      ps.println("</" + tableName.toLowerCase() + ">\n");
   }

   private void printXmlElement(String elementData, String columnName,PrintStream ps){
      //One way to handle null values by not including null elements, that means using
minOccurs="0" in the schema
      if(!(elementData==null || elementData.equals(""))){
                ps.println("\t<" + columnName.toLowerCase() + ">" + trimWhite(elementData) +  "</" +
columnName.toLowerCase() + ">");
                }
   }

   private void printXmlRootEnd(PrintStream ps){
      ps.println("</root>");
   }

}//end class RelMap
```

Ο κώδικας Java για την κλάση **ORSchema** είναι:

```
import java.sql.*;
import java.util.Vector;
import java.io.*;
/**
 * @author Michail Mpellas
 * email: mmpell@it.teithe.gr
 * class: ORSchema
 * @version 1.0
 */
public class ORSchema {



   private String     url ="";
   static String     driverClassName ="";
   private String     username="";
   private String     passwd ="";
   private Connection dbConnection    = null;
   private Statement  statement       = null;
   private ResultSet rs               = null;
   private PrintStream schema         = null;
   private Vector<String> tableName = new Vector<String>();
   private Vector<String> tableType = new Vector<String>();
   //private Vector<String> tablesPK = new Vector<String>();
   private Vector<String> udtTypeName = new Vector<String>();
   private Vector<String> udtTypeCat = new Vector<String>();
   private Vector<String> funcName = new Vector<String>();
   private Vector<String> udtDistinctTypes = new Vector<String>();
   private Vector<String> udtStructuredTypes = new Vector<String>();
   private Vector<String> udtDistinctBaseTypes = new Vector<String>();
   private Vector<String> columnName = new Vector<String>();
   private Vector<String> columnType = new Vector<String>();//Vector elemData = new Vector();

   public ORSchema(String _url,String _username,String _password){
      this.url = _url;
      this.username = _username;
      this.passwd = _password;
   }

   //Performs all the necessary tasks to perform the mapping of an ORDBMS schema to XML
```

Schema

```
   public void createORSchema(Connection dbConnection,String location, String fileName) throws Exception{
      //get the connection, database metadata, the tables of the database, the UDTs and the functions from the schema
      Connection con = dbConnection;
      DatabaseMetaData dmd = getDBMetadata(con);
      getTables(dmd);
      getUserDefinedTypes();
      getFunctions();

      //create new print Stream to write to a file
      try{
         schema = new PrintStream(new FileOutputStream(location + "/" + fileName + "_schema.xsd"));
      }catch(IOException ioe){System.out.println(ioe.toString());}
      printSchemaDeclaration(schema);

      //for every table print an xml comment with its table name and table type, and its UDT type, get its data and columns and print them out
      for(int i=0; i<tableName.size(); i++){
         schema.println("<!--" + tableType.elementAt(i) +" " + tableName.elementAt(i) + " is of type " + udtStructuredTypes.elementAt(i) +" -->\n");

         ResultSet data = getRecords(tableName.elementAt(i),statement);
         getColumnsForEachTable(dmd,data,username,tableName.elementAt(i),schema);
      }//end for

      printAllTypes(schema); //print all the UDTs found in the schema
      printAllFunctions(schema); //print all the functions found in the schema
      printSchemaRootEnd(schema);



      }


   //get the tables of the database
   private void getTables(DatabaseMetaData dbmd) throws SQLException{
      String [] tableTypes = {"TYPED TABLE"};
```

```
        //Getting the db tables
         rs = dbmd.getTables("%", username, "%", tableTypes);
        while (rs.next()) {
           try{
                tableName.add(rs.getString("TABLE_NAME"));
                tableType.add(rs.getString("TABLE_TYPE"));
                if(rs.getString("TABLE_TYPE").compareTo("TABLE")==0){
                  //getPrimaryKeys();
                  //getForeignKeys();
                }

            }catch(com.ibm.db2.jcc.am.SqlException pse){System.out.println(pse.toString());}
        }
    }//end gateTables

    //get the user defined types from the database
    private void getUserDefinedTypes() throws SQLException{
       ResultSet udt = statement.executeQuery("SELECT * FROM
SYSIBM.USER_DEFINED_TYPES AS USER_DEFINED_TYPES WHERE
USER_DEFINED_TYPE_SCHEMA='"+ username.toUpperCase()+"'");
        while (udt.next()) {
           try{
                udtTypeName.add(udt.getString("USER_DEFINED_TYPE_NAME"));
                udtTypeCat.add(udt.getString("USER_DEFINED_TYPE_CATEGORY"));


if(udt.getString("USER_DEFINED_TYPE_CATEGORY").compareTo("STRUCTURED")==0){
                  udtStructuredTypes.add(udt.getString("USER_DEFINED_TYPE_NAME"));
                  System.out.println("STRUCTURED:"
+udt.getString("USER_DEFINED_TYPE_NAME"));
                }else
if(udt.getString("USER_DEFINED_TYPE_CATEGORY").compareTo("DISTINCT")==0){
                  udtDistinctTypes.add(udt.getString("USER_DEFINED_TYPE_NAME"));
                  udtDistinctBaseTypes.add(udt.getString("ORDERING_ROUTINE_NAME"));
                }
            }catch(com.ibm.db2.jcc.am.SqlException pse){System.out.println(pse.toString());}
        }
    }

    //get the functions found in the schema
```

```
   private void getFunctions() throws SQLException{
   ResultSet functions = statement.executeQuery("SELECT DISTINCT FUNCNAME FROM SYSCAT.Functions WHERE FUNCSCHEMA='"+ username.toUpperCase()+"'");

      while(functions.next()){
         funcName.add(functions.getString("FUNCNAME"));
      }
   }

   //get the table records in a resultSet
   private ResultSet getRecords(String tableName, Statement statement) throws SQLException{
      String sqlQuery = "";
      sqlQuery = "SELECT * FROM " + tableName;
      ResultSet dataRS = statement.executeQuery(sqlQuery);
      return dataRS;
   }

   //Initiate connection
   public Connection initConnection(String driverClassName) throws ClassNotFoundException, SQLException{
   Class.forName (driverClassName);
      dbConnection = DriverManager.getConnection (url, username, passwd);
      statement    = dbConnection.createStatement();
      System.out.println("Connection Initiated...\n Drivers Loaded: " + driverClassName);
      return dbConnection;
   }

   //close connection
   public void closeConnection() throws SQLException{
      schema.close(); //close the PrintStream
      statement.close();
      dbConnection.close();
   }

   //Getting the database MetaData
   private DatabaseMetaData getDBMetadata(Connection dbConnection) throws SQLException{
      DatabaseMetaData dbmd = dbConnection.getMetaData();
      return dbmd;
   }
```

```
   //Getting the table metadata in a resultset MetaData
   private ResultSetMetaData getRecordsMetaData(ResultSet dataRS) throws SQLException{
      ResultSetMetaData rsmd = dataRS.getMetaData();
      return rsmd;
   }

   //get the columns, the data, and print them all out accordingly
   private void getColumnsForEachTable(DatabaseMetaData dbmd,ResultSet dataRS, String
username, String tableName, PrintStream schema) throws SQLException{
      ResultSetMetaData rsmd = null;

         columnName.clear(); //clear the columnNames every time
         rsmd = getRecordsMetaData(dataRS);
         ResultSet colrs = dbmd.getColumns("%", username, tableName, "%"); //get table Columns

            while (colrs.next()) {
               columnName.add(colrs.getString("COLUMN_NAME")); //add column names to a
vector object
               }



         printSchemaTableStart(tableName,schema);
               columnType.clear();
               int oid=0;

            for(int j=1; j<=rsmd.getColumnCount(); j++){
               columnType.add(getXMLType(rsmd.getColumnType(j)));
//rsmd.getColumnTypeName(j) COLUMNTYPENAME METHOD
              if(j-1==0){
                oid = 0;
               }else{
                  printSchemaElement(columnName.elementAt(j-1),columnType.elementAt(j-
1),schema);
               }
            }//end for
         printSchemaTableEnd(schema,oid);
      //end while
}//end getColumnsForEachTable
```

```
   public static String trimWhite(String token) //clear all white space before and after of the input string
   {
      String trimmed="";
         try{
               trimmed = token.replaceAll("\\s+$", "");
            }
         catch(NullPointerException np){ return "";}

      return trimmed;
     } //end trimWhite


   private void printSchemaDeclaration(PrintStream schema){
      schema.println("<?xml version=" + "" + "1.0" +"" + " encoding=" +"" + "utf-8" +"" + "?>");
      schema.println("<xs:schema  xmlns:xs="  +  ""  +  "http://www.w3.org/2001/XMLSchema"  +  "" +">" );
      schema.println("<xs:element name=" + "" + "root" + "" + ">");
      schema.println("\t\t<xs:complexType>");
      schema.println("\t\t\t<xs:sequence>");
   }

   private void printSchemaTableStart(String tableName,PrintStream schema){
      schema.println("\t\t\t\t<xs:element  name="  +  ""  +  tableName.toLowerCase()  +  ""+  " maxOccurs=" + "" + "unbounded" + "" + ">");
      schema.println("\t\t\t\t\t<xs:complexType>");
      schema.println("\t\t\t\t\t\t<xs:sequence>");
   }

   private void printSchemaTableEnd(PrintStream schema, int oid){
      schema.println("\t\t\t\t\t\t</xs:sequence>");
      schema.println("\t\t\t\t\t\t<xs:attribute  name="  +  ""  +  columnName.elementAt(oid)  +  ""  +  " type=" + "" + "xs:" + columnType.elementAt(oid) + "" + " use=" + "" + "required" + "" + "/>");
      schema.println("\t\t\t\t\t</xs:complexType>");
      schema.println("\t\t\t\t</xs:element>");
   }

   private void printSchemaElement(String columnName, String columnType,PrintStream schema){
```

```
        schema.println("\t\t\t\t\t\t\t<xs:element  name=" + "" + columnName.toLowerCase() + "" + "
type=" + "" +"xs:" + columnType + "" + " minOccurs=" +"" + "0" + "" +" />");
    }

    private void printSchemaRootEnd(PrintStream schema){
        schema.println("\t\t\t</xs:sequence>");
        schema.println("\t\t</xs:complexType>");
        schema.println("</xs:element>");
        schema.println("</xs:schema>");
    }

    private void printAllTypes(PrintStream schema){ //print all the UDTs found in the schema
         schema.println("\n<!-- ****** A list of all the User Types ****** -->\n");
        for(int i=0; i<udtTypeName.size(); i++){
            schema.println("<!--" + udtTypeName.elementAt(i) +" -->");
        }
    }

    private void printAllFunctions(PrintStream schema){ //print all the functions
         schema.println("\n\n<!-- ****** A list of all the functions ****** -->\n");
        for(int i=0; i<funcName.size(); i++){
            schema.println("<!--" + funcName.elementAt(i) +" -->");
        }
    }
private String getXMLType(int type) { // get the XML type that corresponds to the input SQL type
        switch(type){
            case Types.ARRAY: return "string";
            case Types.BIGINT: return "long";
            case Types.BINARY: return "base64Binary";
            case Types.BIT: return "short";
            case Types.BLOB: return "base64Binary";
            case Types.BOOLEAN: return "boolean";
            case Types.CHAR: return "string";
            case Types.CLOB: return "string";
            case Types.DATALINK: return "anyURI";
            case Types.DATE: return "date";
            case Types.DECIMAL: return "decimal";
            case Types.DOUBLE: return "double";
            case Types.FLOAT: return "float";
            case Types.INTEGER: return "int";
```

```
            case Types.LONGVARBINARY: return "base64Binary";
            case Types.LONGVARCHAR: return "string";
            case Types.NUMERIC: return "decimal";
            case Types.REAL: return "float";
            case Types.SMALLINT: return "short";
            //case Types.TIMESTAMP: return "time";
            case Types.TIMESTAMP: return "dateTime";
            case Types.TINYINT: return "short";
            case Types.VARBINARY: return "base64Binary";
            case Types.VARCHAR: return "string";
            case Types.DISTINCT: return "string";
            case Types.JAVA_OBJECT: return "string";
            case Types.NULL: return "string";
            case Types.OTHER: return "string";
            case Types.REF: return "string";
            case Types.STRUCT: return "string";
            case 100: return "anyType";
        }
    return "string";
}
//private void getPrimaryKeys() {
//        try{
//                                    ResultSet pkrs = statement.executeQuery("SELECT * FROM
SYSIBM.TABLE_CONSTRAINTS where TABLE_SCHEMA='"+ username.toUpperCase() + "'");
//          // pkrs.beforeFirst();
//
//        while(pkrs.next()){
//                String constraintType = pkrs.getString("CONSTRAINT_TYPE");
//                if(constraintType.compareTo("PRIMARY KEY")==0){
//                   String constraintName = pkrs.getString("CONSTRAINT_NAME");
//                   tablesPK.add(constraintName);
//                System.out.println("COLUMN :" + constraintName + "IS PRIMARY KEY");
//                }
//              }
//
//           }catch(java.sql.SQLException jsql){}
//           //System.out.println(tname +" \n " + pk + "\n");
//    } // end isPrimaryKey

}//end class ORSchema
```

Ο κώδικας Java για την κλάση **ORMap** είναι:

```
import java.sql.*;
import java.util.Vector;
import java.io.*;
/**
 * @author Michail Mpellas
 * email: mmpell@it.teithe.gr
 * class: ORMap
 * @version 1.0
 */
public class ORMap {



   private String     url ="";
   static String     driverClassName = "";
   private String     username="";
   private String     passwd ="";
   private Connection dbConnection    = null;
   private Statement  statement       = null;
   private ResultSet rs               = null;
   private PrintStream ps             = null;
   private Vector<String> tableName = new Vector<String>();
   private Vector<String> tableType = new Vector<String>();
   private Vector<String> tablesPK = new Vector<String>();
   private Vector<String> udtTableName = new Vector<String>();
   private Vector<String> udtTableType = new Vector<String>();
   private Vector<String> columnName = new Vector<String>();
   private Vector<String> columnType = new Vector<String>();



   public ORMap(String _url,String _username,String _password){
      this.url = _url;
      this.username = _username;
      this.passwd = _password;
   }

   //Performs all the necessary tasks to perform the mapping of an ORDBMS to XML
   public void StartMappingORDBMStoXML(Connection dbConnection,String location, String
fileName) throws Exception{
      //get the connection, database metadata, the tables of the database and the UDTs
```

```
        Connection con = dbConnection;
        DatabaseMetaData dmd = getDBMetadata(con);
        getTables(dmd);
        getUserDefinedTypes();

        //create new print Stream to write to a file
           try{
               ps = new PrintStream(new FileOutputStream(location + "/" + fileName + ".xml"));
           }catch(IOException ioe){System.out.println(ioe.toString());}

        printXmlDeclaration(ps,fileName);

           //for every table print an xml comment with its name and type, get its data, primary keys and
columns and print them out
           for(int i=0; i<tableName.size(); i++){
              ps.println("<!--" + tableName.elementAt(i)+" " + tableType.elementAt(i) + "-->\n");
              ResultSet data = getRecords(tableName.elementAt(i),statement);
              getPKeys(dmd,username,tableName.elementAt(i));
              getColumnsForEachTable(dmd,data,username,tableName.elementAt(i),ps);
           }//end for

        printXmlRootEnd(ps);

        }

    //get the tables of the database
    private void getTables(DatabaseMetaData dbmd) throws SQLException{
        String [] tableTypes = {"TYPED TABLE"}; //Only tables with user defined types

          //Getting the db tables
           rs = dbmd.getTables("%", username, "%", tableTypes);
          while (rs.next()) {
             try{
                  tableName.add(rs.getString("TABLE_NAME"));
                  tableType.add(rs.getString("TABLE_TYPE"));
              }catch(com.ibm.db2.jcc.am.SqlException pse){System.out.println(pse.toString());}
          }
    }//end gateTables

    //Get the user defined types from the database
```

```
    private void getUserDefinedTypes() throws SQLException{
        ResultSet udt = statement.executeQuery("SELECT * FROM
SYSIBM.USER_DEFINED_TYPES AS USER_DEFINED_TYPES WHERE
USER_DEFINED_TYPE_SCHEMA='"+ username.toUpperCase()+"'");
          while (udt.next()) {
             try{
                  udtTableName.add(udt.getString("USER_DEFINED_TYPE_NAME"));
                  udtTableType.add(udt.getString("USER_DEFINED_TYPE_CATEGORY"));
              }catch(com.ibm.db2.jcc.am.SqlException pse){System.out.println(pse.toString());}
          }
    }

    //get the primary keys of a table
     private void getPKeys(DatabaseMetaData dbmd,String username,String tableName) throws
SQLException{
        tablesPK.clear();
        ResultSet pkrs = dbmd.getPrimaryKeys("%", username , tableName);
        while(pkrs.next()){
          tablesPK.add(pkrs.getString("COLUMN_NAME"));
          }
    }

     //check if a column name corresponds to a primary key
    private boolean isPKey(String ColumnName){
        boolean isPK = false;
        for(int i=1; i<=tablesPK.size(); i++){
           if(tablesPK.elementAt(i-1).contains(ColumnName))
              isPK = true;
        }
        return isPK;
    }//end isPKey

   //get the table records in a resultSet
    private ResultSet getRecords(String tableName, Statement statement) throws SQLException{
        String sqlQuery = "";
        sqlQuery = "SELECT * FROM " + tableName;
        ResultSet dataRS = statement.executeQuery(sqlQuery);
        return dataRS;
    }
    //Initiate connection
```

```
   public Connection initConnection(String driverClassName) throws ClassNotFoundException,
SQLException{
   Class.forName (driverClassName);
      dbConnection = DriverManager.getConnection (url, username, passwd);
      statement    = dbConnection.createStatement();
      System.out.println("==>Connection Initiated\n ==>Drivers Loaded: " + driverClassName);
      return dbConnection;
   }
   //Close connection
   public void closeConnection() throws SQLException{
      System.out.println("  ==>Connection closed on User's Request\n");
      ps.close();
      statement.close();
      dbConnection.close();
   }

   //Getting the database MetaData
   private DatabaseMetaData getDBMetadata(Connection dbConnection) throws SQLException{
      DatabaseMetaData dbmd = dbConnection.getMetaData();
      return dbmd;
   }

   //Getting the table metadata in a resultset MetaData
   private ResultSetMetaData getRecordsMetaData(ResultSet dataRS) throws SQLException{
      ResultSetMetaData rsmd = dataRS.getMetaData();
      return rsmd;
   }

   //get the columns, the data, and print them all out accordingly
   private void getColumnsForEachTable(DatabaseMetaData dbmd,ResultSet dataRS, String
username, String tableName, PrintStream ps) throws SQLException{

      while(dataRS.next()){ //for every table record
         columnName.clear(); //clear the columnNames every time
         ResultSetMetaData rsmd = getRecordsMetaData(dataRS);
         ResultSet colrs = dbmd.getColumns("%", username, tableName, "%"); //get table Columns

            while (colrs.next()) {
               columnName.add(colrs.getString("COLUMN_NAME")); //add column names to a
vector object
```

```
                    if(isPKey(colrs.getString("COLUMN_NAME"))){ //if the column name corresponds to
a primary key add it to the primary keys vector, tablesPK
                        tablesPK.add(colrs.getString("COLUMN_NAME"));
                    }
                }

            int oid=0;


printXmlTableStart(tableName,columnName.elementAt(oid),trimWhite(dataRS.getString(columnNa
me.elementAt(oid))),ps); //print xml table start structure

                for(int j=1; j<=rsmd.getColumnCount(); j++){ //for every column
                    columnType.add(rsmd.getColumnTypeName(j)); //add its type to a vector
                    String elementData = trimWhite(dataRS.getString(columnName.elementAt(j-1))); //get
the data

                    if(j>1){ //skipping the first column, to print it later as attribute
                        printXmlElement(elementData,columnName.elementAt(j-1),ps);  //print the rest of
the elements
                    }
                }//end for
            printXmlTableEnd(tableName,ps);
        }//end while
}//end getColumnsForEachTable

    public static String trimWhite(String token) //clear all white space before and after of the input
string
    {
        String trimmed="";
            try{
                trimmed = token.replaceAll("\\s+$", "");
            }
            catch(NullPointerException np){ return "";}

        return trimmed;
      } //end trimWhite


    private void printXmlDeclaration(PrintStream ps, String fileName){
```

```
        ps.println("<?xml version=" + "" + "1.0" +"" + " encoding=" +"" + "utf-8" +"" + " standalone=" +""
+ "no" +"" +"?>\n");

        ps.println("<root xmlns:xsi=" + "" + "http://www.w3.org/2001/XMLSchema-instance" + "" +
                    " xsi:noNamespaceSchemaLocation=" + "" + fileName +"_schema.xsd" + "" + ">\n");
    }

    private void printXmlTableStart(String tableName,String columnName, String oid,PrintStream
ps){
        ps.println("<" + tableName.toLowerCase() + " " + columnName + "=" + "" + oid + ""+ ">");

    }

    private void printXmlTableEnd(String tableName,PrintStream ps){
        ps.println("</" + tableName.toLowerCase() + ">\n");
    }

    private void printXmlElement(String elementData, String columnName,PrintStream ps){
        if(!(elementData==null || elementData.equals(""))){
                //One way to handle null values by not including null elements in the XML document,
that means using minOccurs="0" in the schema
                ps.println("\t<" + columnName.toLowerCase() + ">" + trimWhite(elementData) +  "</" +
columnName.toLowerCase() + ">");
                }
    }

    private void printXmlRootEnd(PrintStream ps){
        ps.println("</root>");
    }

}//end class ORMap
```

Καταμέτρηση Λέξεων

Η καταμέτρηση των λέξεων συμπεριλαμβανομένου της βιβλιογραφίας αλλά όχι του παραρτήματος.