ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Μετατροπή Βάσης Δεδομένων Σε Σχήμα XML

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Π. Κολοβός

Αθήνα, Φεβρουάριος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Μετατροπή Βάσης Δεδομένων Σε Σχήμα XML

Database Transformation Into XML Schema

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Π. Κολοβός

Επιβλέπων : Γεώργιος Β. Στασσινόπουλος
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Μετατροπή Βάσης Δεδομένων Σε Σχήμα XML

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Π. Κολοβός

Επιβλέπων : Γεώργιος Β. Στασσινόπουλος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 13η Φεβρουαρίου 2009.

.....	.....	.....
Γεώργιος Β. Στασσινόπουλος	Μ. Θεολόγου	Ε. Συκάς
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2009

.....
Κωνσταντίνος Π. Κολοβός

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η μετατροπή μιας βάσης δεδομένων, η οποία αποτελεί μέρος του Microsoft SQL Server, σε ένα αρχείο XML συγκεκριμένων προδιαγραφών έτσι ώστε να αποτελεί μια απεικόνιση της βάσης και να μπορεί να χρησιμοποιηθεί αντ'αυτής σε απλές εφαρμογές και συστήματα που δεν είναι σε θέση να διατηρούν έναν SQL Server(όπως για παράδειγμα τα λειτουργικά συστήματα κινητών τηλεφώνων). Βασικός στόχος επίσης είναι η εξοικείωση με το Microsoft .NET framework 2, εργαλείο αιχμής το οποίο χρησιμοποιείται κατά κόρον σήμερα για ανάπτυξη εφαρμογών μεγάλης η και μικρής κλίμακας. Η αξιοποίηση του εργαλείου αυτού γίνεται μέσα από το περιβάλλον Microsoft Visual Studio ενώ όπως είναι αναμενόμενο γίνεται χρήση και του Microsoft SQL Server, πρόγραμμα διαχείρισης βάσεων δεδομένων το οποίο θα διατηρεί τις βάσεις. Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι η C# .NET και ταυτόχρονα γίνεται χρήση της γλώσσας SQL και t-SQL με στόχο την αναζήτηση δεδομένων αλλά και πληροφοριών που χαρακτηρίζουν τη βάση δεδομένων.

Λέξεις Κλειδιά

Βάση Δεδομένων, XML, Microsoft SQL Server, Σύστημα Διαχείρισης Βάσεων Δεδομένων, Microsoft Visual Studio, Microsoft .NET framework 2, C#. NET, SQL, t-SQL.

Abstract

The subject of the following diploma thesis is the transformation of a database which resides in an SQL Server into an XML file of certain specifications given which the XML would be a functional reflection of the database, able to be used instead of it in simple applications and systems which are not capable of maintaining an SQL Server (for example mobile phones operating systems). Primary objective is also to acquaint with Microsoft .NET framework 2, a powerful tool used frequently nowadays as far as either small or extensive application development is concerned. We make use of this tool through the usage of the Microsoft Visual Studio environment whereas Microsoft SQL Server, a database management system, is also used as expected in order to maintain the databases needed. Furthermore, the programming language used is C# .NET while SQL and t-SQL is also used in order to retrieve database data and information which the database and its structure.

Keywords

Database, XML, Microsoft SQL Server, Database Management System, Microsoft Visual Studio, Microsoft .NET framework 2, C# .NET, SQL, t-SQL.

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγικές Έννοιες

1.1 Βάση Δεδομένων

Βάση δεδομένων είναι μια δομημένη συλλογή από εγγραφές η δεδομένα που είναι καταχωρημένα σε ένα υπολογιστικό σύστημα. Η προαναφερθείσα δομή επιτυγχάνεται οργανώνοντας τα δεδομένα βάση ενός μοντέλου. Το πλέον συνηθισμένο μοντέλο που χρησιμοποιείται σήμερα είναι το σχεσιακό μοντέλο.

1.2 Σύστημα Διαχείρισης Βάσης Δεδομένων(DBMS)

Μία βάση δεδομένων σε ένα υπολογιστικό σύστημα στηρίζεται σε λογισμικό προκειμένου να οργανώσει την καταχώρηση των δεδομένων. Το λογισμικό αυτό είναι γνωστό σαν σύστημα διαχείρισης βάσεων δεδομένων. Τα συστήματα αυτά κατηγοριοποιούνται βάση των μοντέλων βάσεων δεδομένων που υποστηρίζουν. Το μοντέλο αυτό τείνει να καθορίζει τις γλώσσες αναζήτησης που είναι σε θέση να έχουν πρόσβαση στη βάση. Μεγάλο μέρος ωστόσο της εσωτερικής δομής ενός ΣΔΒΔ είναι ανεξάρτητο του χρησιμοποιούμενου μοντέλου και επικεντρώνεται κυρίως σε διαχειριστικούς παράγοντες όπως η απόδοση, ο ταυτοχρονισμός, η ασφάλεια και ανάκαμψη από προβλήματα του υλικού. Σε αυτούς τους τομείς υπάρχουν μεγάλες διαφορές μεταξύ των προϊόντων στην αγορά. Ένα σύστημα διαχείρισης βάσεων δεδομένων είναι και ο Microsoft SQL Server.

1.3 XML

Η επεκτάσιμη γλώσσα σήμανσης (Extensible Markup Language – XML) αποτελεί μια προδιαγραφή γενικού σκοπού για τη δημιουργία γλωσσών σήμανσης. Χαρακτηρίζεται ως επεκτάσιμη για το λόγο ότι δίνει στο χρήστη τη δυνατότητα να καθορίσει ο ίδιος τα στοιχεία σήμανσης. Στόχος της είναι να συμβάλλει στο διαμοιρασμό των δομημένων δεδομένων σε πληροφοριακά συστήματα κυρίως μέσω του διαδικτύου, στην κωδικοποίηση εγγράφων και στη σειριοποίηση δεδομένων. Σε συνδυασμό με άλλα πρότυπα κάνει εφικτό τον καθορισμό των περιεχομένων ενός εγγράφου ανεξάρτητα από τη μορφοποίησή του κάνοντας ευκολότερη την επαναχρησιμοποίηση του από διαφορετικές εφαρμογές. Το πλέον σημαντικό είναι ότι η XML παρέχει το βασικό συντακτικό που μπορεί να χρησιμοποιηθεί για τον διαμοιρασμό πληροφοριών μεταξύ υπολογιστών και εφαρμογών διαφορετικού είδους καθώς και διαφόρων οργανισμών χωρίς να είναι αναγκαίο να περάσουμε από πολλά στρώματα μετατροπών.

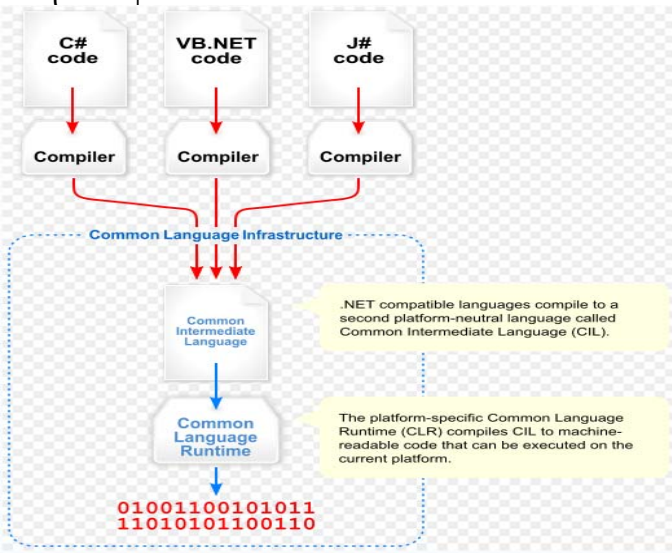
Η XML στηρίζεται σε ετικέτες, τα ονόματα των οποίων καθορίζονται από το χρήση βάση των αναγκών της εφαρμογής του, οι οποίες ανοίγουν και κλείνουν και στο εσωτερικό τους περιέχουν τα ζητούμενα δεδομένα. Οι ετικέτες αυτές ενθυλακώνονται η μία μέσα στην άλλη με αποτέλεσμα να δημιουργείται μια δενδρική μορφή.

Ένα παράδειγμα αρχείου XML είναι το ακόλουθο:

```
<recipe name="bread" prep_time="5 mins" cook_time="3 hours">
   <title>Basic bread</title>
   <ingredient amount="8" unit="dL">Flour</ingredient>
   <ingredient amount="10" unit="grams">Yeast</ingredient>
   <ingredient amount="4" unit="dL" state="warm">Water</ingredient>
   <ingredient amount="1" unit="teaspoon">Salt</ingredient>
   <instructions>
     <step>Mix all ingredients together.</step>
     <step>Knead thoroughly.</step>
     <step>Cover with a cloth, and leave for one hour in warm room.</step>
     <step>Knead again.</step>
     <step>Place in a bread baking tin.</step>
     <step>Cover with a cloth, and leave for one hour in warm room.</step>
     <step>Bake in the oven at 180(degrees)C for 30 minutes.</step>
   </instructions>
 </recipe>
```

1.4 Microsoft .NET

Το Microsoft .NET Framework είναι ένα πλαίσιο λογισμικού που είναι διαθέσιμο με διάφορα λειτουργικά συστήματα της Microsoft. Περιλαμβάνει μια ιδιαίτερα μεγάλη βιβλιοθήκη κλάσεων σε συνηθισμένα προγραμματιστικά προβλήματα και μια εικονική μηχανή που είναι σε θέση να διαχειρίζεται την εκτέλεση προγραμμάτων γραμμένα ειδικά για το πλαίσιο αυτό. Οι έτοιμες κλάσεις που αποτελούν τη βασική βιβλιοθήκη κλάσεων του πλαισίου καλύπτουν ένα μεγάλο εύρος προγραμματιστικών αναγκών σε μια πληθώρα τομέων που συμπεριλαμβάνουν τα εξής: interface, data access, database connectivity, cryptography, web application development, numeric algorithms και network communications. Η βιβλιοθήκη κλάσεων χρησιμοποιείται από προγραμματιστές που τη συνδυάζουν με το δικό τους κώδικα προκειμένου να αναπτύξουν εφαρμογές.
Μια από τις καινοτομίες του .NET Framework είναι το Common Language Infrastructure (CLI). Σκοπός του είναι να παρέχει μια πλατφόρμα ουδέτερη σε γλώσσες προγραμματισμού για την ανάπτυξη και εκτέλεση εφαρμογών συμπεριλαμβάνοντας συναρτήσεις για τη διαχείριση εξαιρέσεων, τη συλλογή «σκουπιδιών» και την ασφάλεια.



1.5 Αντικειμενοστρεφής Προγραμματισμός (OOP)

Αντικειμενοστρεφής προγραμματισμός ονομάζεται ένα προγραμματιστικό υπόδειγμα το οποίο εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε τη δεκαετία του 1990. Αντικατέστησε σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού ενώ πρόκειται για μια μεθοδολογία ανάπτυξης προγραμμάτων όπου ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μιας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται αντικείμενο και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη τύπου δεδομένων που ονομάζεται κλάση. Η κλάση προδιαγράφει τόσο δεδομένα όσο και τις διαδικασίες οι οποίες επιδρούν επάνω τους. Αυτή υπήρξε και η πρωταρχική καινοτομία του αντικειμενοστρεφούς προγραμματισμού.
Υπό αυτό το πρίσμα μπορεί να οριστεί μια προδιαγραφή δομής αποθήκευσης η οποία να περιέχει τόσο ιδιότητες όσο και πράξεις ή χειρισμούς επ'αυτών των ιδιοτήτων. Αξιοσημείωτο είναι ότι τα αντικείμενα καταλαμβάνουν χώρο στη μνήμη του υπολογιστή ενώ οι κλάσεις αποτελούν απλώς «καλούπια». Οι αιτίες που ώθησαν στην ανάπτυξη του αντικειμενοστρεφούς προγραμματισμού ήταν οι ίδιες με αυτές που οδήγησαν στην ανάπτυξη του δομημένου προγραμματισμού π.χ. ευκολία συντήρησης, οργάνωσης, χειρισμού και επαναχρησιμοποίησης κώδικα μεγάλων και πολύπλοκων εφαρμογών. Τελικά όμως η αντικειμενοστρέφεια επικρατήσει και συνεχίζει να επικρατεί καθώς ήταν σε θέση να ανταπεξέλθει σε προγράμματα πολύ μεγαλύτερου όγκου και πολυπλοκότητας.
Σημαντικό χαρακτηριστικό του αντικειμενοστρεφούς προγραμματισμού είναι επίσης η έννοια της κληρονομικότητας. Κληρονομικότητα ονομάζεται η ιδιότητα των κλάσεων να επεκτείνονται σε νέες κλάσεις, ρητά δηλωμένες ως κληρονόμους (υποκλάσεις η θυγατρικές κλάσεις), οι οποίες μπορούν να επαναχρησιμοποιήσουν τις μεταβιβάσιμες μεθόδους και ιδιότητες της γονικής του κλάσεις αλλά και να προσθέσουν δικές τους. Στιγμιότυπα των θυγατρικών κλάσεων μπορούν να χρησιμοποιηθούν όπου απαιτούνται στιγμιότυπα των γονικών (εφόσον η θυγατρική είναι κατά κάποιον τρόπο μία πιο εξειδικευμένη εκδοχή της γονικής), αλλά το αντίστροφο δεν ισχύει. Παράδειγμα κληρονομικότητας είναι μία γονική κλάση Vehicle (=Όχημα) και οι δύο πιο εξειδικευμένες υποκλάσεις της Car (=Αυτοκίνητο) και Bicycle (=Ποδήλατο), οι οποίες λέμε ότι "κληρονομούν" από αυτήν. Πολλαπλή κληρονομικότητα είναι η δυνατότητα που προσφέρουν ορισμένες γλώσσες προγραμματισμού μία κλάση να κληρονομεί ταυτόχρονα από περισσότερες από μία γονικές. Από μία υποκλάση μπορούν να προκύψουν νέες υποκλάσεις που κληρονομούν από αυτήν, με αποτέλεσμα μία ιεραρχία κλάσεων που συνδέονται μεταξύ τους "ανά γενιά" με σχέσεις κληρονομικότητας.

1.6 Αντικειμενοστρεφής Ανάλυση Και Σχεδίαση (OOA&D)

Η αντικειμενοστρεφής ανάλυση και σχεδίαση αποτελούν αναμφισβήτητα τη ραχοκοκαλιά για την ανάπτυξη στιβαρών και λειτουργικών εφαρμογών. Η αντικειμενοστρεφής ανάλυση επιδιώκει τη δημιουργία ενός μοντέλου του προβλήματος με τον προσδιορισμό και την αναγνώριση των αντικειμένων καθώς και των σχετικών κλάσεων, στοιχεία τα οποία προέρχονται από το πεδίο του προβλήματος. Από την άλλη μεριά η αντικειμενοστρεφής σχεδίαση προσδιορίζει τον τρόπο με τον οποίο οι πιο πάνω αφηρημένες έννοιες είναι δυνατό αφενός να αναπαρασταθούν και να αλληλεπιδρούν μεταξύ τους σε επίπεδο λογισμικού και αφετέρου να ενταχθούν σε ένα γενικότερο πλαίσιο που να μοντελοποιεί το υπό εξέταση σύστημα.
Μετά την ευρεία διάδοση του ΑΠ κατά τη δεκαετία του '90, το αντικειμενοστρεφές μοντέλο σχεδίασης (με κλάσεις, κληρονομικότητα, αντικείμενα και τυποποιημένες αλληλεπιδράσεις μεταξύ τους) επικράτησε ακόμη και για μοντελοποίηση που δεν περιελάμβανε καν προγραμματισμό (π. χ. σχήματα βάσεων δεδομένων). Έτσι αναπτύχθηκαν διάφορες πρότυπες γλώσσες μοντελοποίησης λογισμικού οι οποίες τυποποιούσαν οπτικά σύμβολα και συμπεριφορές με στόχο την αφαιρετική περιγραφή της λειτουργίας και της δομής ενός υπολογιστικού συστήματος. Οι γλώσσες αυτές είχαν εξαρχής έναν εμφανή αντικειμενοστρεφή προσανατολισμό. Τελικά οι πιο δημοφιλείς από αυτές ενοποιήθηκαν στο κοινό πρότυπο UML που η πρώτη του έκδοση οριστικοποιήθηκε το 1997.
Η UML παρέχει μια σειρά από εργαλεία για χρήση κατά την αντικειμενοστρεφή ανάλυση και σχεδίαση. Η Γλώσσα Ενιαίας Μοντελοποίησης (Unified Modeling Language – UML) είναι μια διαγραμματική τεχνική για τον προσδιορισμό, την οπτικοποίηση, την κατασκευή και την τεκμηρίωση των συνιστωσών συστημάτων λογισμικού. Χρησιμοποιείται επίσης για τη μοντελοποίηση επιχειρησιακών διεργασιών (business modeling) και για τη μοντελοποίηση γενικότερων συστημάτων. Η UML ωστόσο δεν είναι μια γλώσσα προγραμματισμού, δεν προσδιορίζει συγκεκριμένα εργαλεία λογισμικού και ούτε περιγράφει μια διεργασία λογισμικού μέσω συγκεκριμένων βημάτων.
Με το πέρασμα του χρόνου κωδικοποιήθηκαν κάποιες ανεπίσημες αρχές για την ορθή σχεδίαση αντικειμενοστρεφών συστημάτων λογισμικού. Οι σπουδαιότερες αρχές είναι οι παρακάτω:

- Αρχή ανοιχτότητας-κλειστότητας του δημιουργού της γλώσσας προγραμματισμού Eiffel, Μπέρναρντ Μέιερ.
- Αρχή υποκατάσταση Λίσκοφ της επιστήμονα υπολογιστών Μπάρμπαρα Λίσκοφ.
- Αρχή αντιστροφής εξαρτήσεων, του γνωστού μηχανικού λογισμικού Ρόμπερτ Σέσιλ Μάρτιν.
- Αρχή διαχωρισμού διασυνδέσεων του Ρόμπερτ Σέσιλ Μάρτιν.
- Αρχή μοναδικής αρμοδιότητας των Τομ Ντε Μάρκο και Μέιρ Πέτζ Τζόουνς.

1.7 SQL και t-SQL

Η SQL (Structured Query Language) είναι μια γλώσσα βάσεων δεδομένων σχεδιασμένη για την ανάσυρση και διαχείριση δεδομένων σε συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων. Η SQL είναι μια γλώσσα προγραμματισμού για αναζήτηση και μετατροπή δεδομένων καθώς και διαχείριση βάσεων δεδομένων. Επιτρέπει την ανάσυρση, εισαγωγή, ανανέωση και διαγραφή δεδομένων. Η πρώτη εκδοχή της αναπτύχθηκε στα εργαστήρια της IBM από τους Ντόναλντ Τσάμπερλιν και Ρέυμοντ Μπόυς στις αρχές τις δεκαετίας του 1970 και αρχικά ονομαζόταν SEQUEL. Αρχικά σχεδιάστηκε σα μια δηλωτική γλώσσα για αναζήτηση και διαχείριση δεδομένων ενώ διαφοροποιήσεις δημιουργήθηκαν από σχεδιαστές συστημάτων διαχείρισης βάσεων δεδομένων (DBMS). Συχνά ασκείται κριτική στην SQL λόγω της έλλειψης συμβατότητας μεταξύ πλατφορμών διαφορετικών κατασκευαστών, ακατάλληλη διαχείριση δεδομένων που απουσιάζουν και λόγω της υπερβολικά περίπλοκης και συχνά πολύ αφηρημένης γραμματικής και κανόνων της γλώσσας.
Η transact-SQL (t-SQL) είναι η επέκταση στην SQL που έγινε από τη Microsoft και τη Sybase όπου χρησιμοποιείται στα προϊόντα Microsoft SQL Server και Adaprive Server Enterprise αντίστοιχα. Τα επιπρόσθετα στοιχεία της γλώσσας αυτής είναι τα ακόλουθα:

- Διαχείριση Ροής
- Τοπικές Μεταβλητές
- Ποικίλες υποστηρικτικές συναρτήσεις για επεξεργασία κειμένου, μαθηματικών κλπ
- Βελτιώσεις στις εντολές DELETE και UPDATE.

2. Προδιαγραφές XML Αρχείου Και Βάσης Δεδομένων

2.1 Προδιαγραφές Βάσης Δεδομένων

Αρχική προδιαγραφή για την εφαρμογή που αναπτύσσουμε είναι η χρησιμοποιούμενη βάση δεδομένων να διατηρείται σε ένα σύστημα διαχείρισης βάσεων δεδομένων της μορφής Microsoft SQL Server.
Το γεγονός αυτό υποδεικνύει ότι για την προσπέλαση και ανάγνωση της βάσης θα χρησιμοποιηθούν μέθοδοι και κλάσεις και εμπεριέχονται στο Namespace `System.Data.SqlClient.SqlConnection` το οποία παρέχεται στις βιβλιοθήκες του .NET και αφορά αποκλειστικά τον Microsoft SQL Server.
Δεύτερος λειτουργικός περιορισμός για τη βάση δεδομένων είναι να μην περιέχει whitespace έτσι ώστε να δημιουργείται ένα συντακτικά σωστό XML αρχείο στο οποίο να μπορεί να γίνει parsing χωρίς προβλήματα.
Σημαντικό επίσης είναι ότι η βάση δεδομένων θα πρέπει να είναι σχετικά μικρή σε μέγεθος δεδομένου ότι η κατασκευασθείσα εφαρμογή προορίζεται να είναι «ελαφριά» ώστε να υποστηρίζεται ακόμα και από υπολογιστικά συστήματα περιορισμένων δυνατοτήτων, όπως αυτά των κινητών τηλεφώνων.
Όσον αφορά τα δεδομένα της βάσης πρόκειται κυρίως για δεδομένα κειμένου ενώ δεν έχει ληφθεί ιδαίτερη μέριμνα για βάσεις δεδομένων που περιέχουν binary δεδομένα, εικόνες κλπ.
Τελικά, θα πρέπει να υπάρχει μια ξεκάθαρη, ανοιχτη διαδρομή από πίνακα σε πίνακα όσον αφορά τις σχέσεις μεταξύ τους, κάτι που δεν επιτρέπει τη δημιουργία βρόγχων.

2.2 Προδιαγραφές Επικεφαλίδων XML

Οι επικεφαλίδες του XML, όπως προδιαγράφεται από το συντακτικό της XML, καθορίζουν αρχικά την έκδοση που χρησιμοποιείται καθώς και την κωδικοποίηση. Συνεπώς όλα τα παραχθέντα XML αρχεία θα πρέπει να ξεκινούν με την ακόλουθη επικεφαλίδα:

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

Στη συνέχεια, στο XML αρχείο θα πρέπει να καθορίζεται το Namespace στο οποίο ανήκει ώστε να ξεχωρίζει από όλα τα υπόλοιπα XML αρχεία. Σαν γενική σύμβαση συνήθως χρησιμοποιείται κάποιο URL, στοιχείο που αυτομάτως κάνει το Namespace αυτό μοναδικό. Στη εφαρμογή αυτή χρησιμοποιείται η ακόλουθη δήλωση, η οποία και χρησιμοποιείται σε όλα τα αρχεία μετά την αρχική επικεφαλίδα. Αυτή είναι η ακόλουθη:

```
- <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:cdb="http://www.cn.ntua.gr/cdb"
xmlns:pf="http://www.cn.ntua.gr/cdb_self"
xmlns:rel="http://www.cn.ntua.gr/relationships_self">
```

Έπειτα, η δήλωση που θα χρησιμοποιείται για τα relationships βάσης θα εμπεριέχεται σε ένα tag που θα παίρνει το όνομά του από τον πίνακα που θα παίρνουμε σα βάση για τα relationships και θα ακολουθείται από το Namespace που θα χρησιμοποιείται για το σκοπό αυτό. Ένα παράδειγμα αυτού απεικονίζεται στη συνέχεια παίρνοντας ως πίνακα βάσης τον πίνακα sales της βάσης δεδομένων Pubs(περιγράφεται αναλυτικά στη συνέχεια):

```
- <sales xmlns="http://www.cn.ntua.gr/relationships_self">
```

Πριν ξεκινήσουμε να παραθέτουμε τα δεδομένα της βάσης δεδομένων θα πρέπει να ανοίγει ένα αρχικό tag που θα έχει το όνομα της βάσης δεδομένων ακολουθούμενο από τη συντομογραφία DB και το Namespace που θα χρησιμοποιείται για τα δεδομένα. Η μορφή αυτού θα είναι η ακόλουθη παίρνοντας ως παράδειγμα τη βάση Pubs:

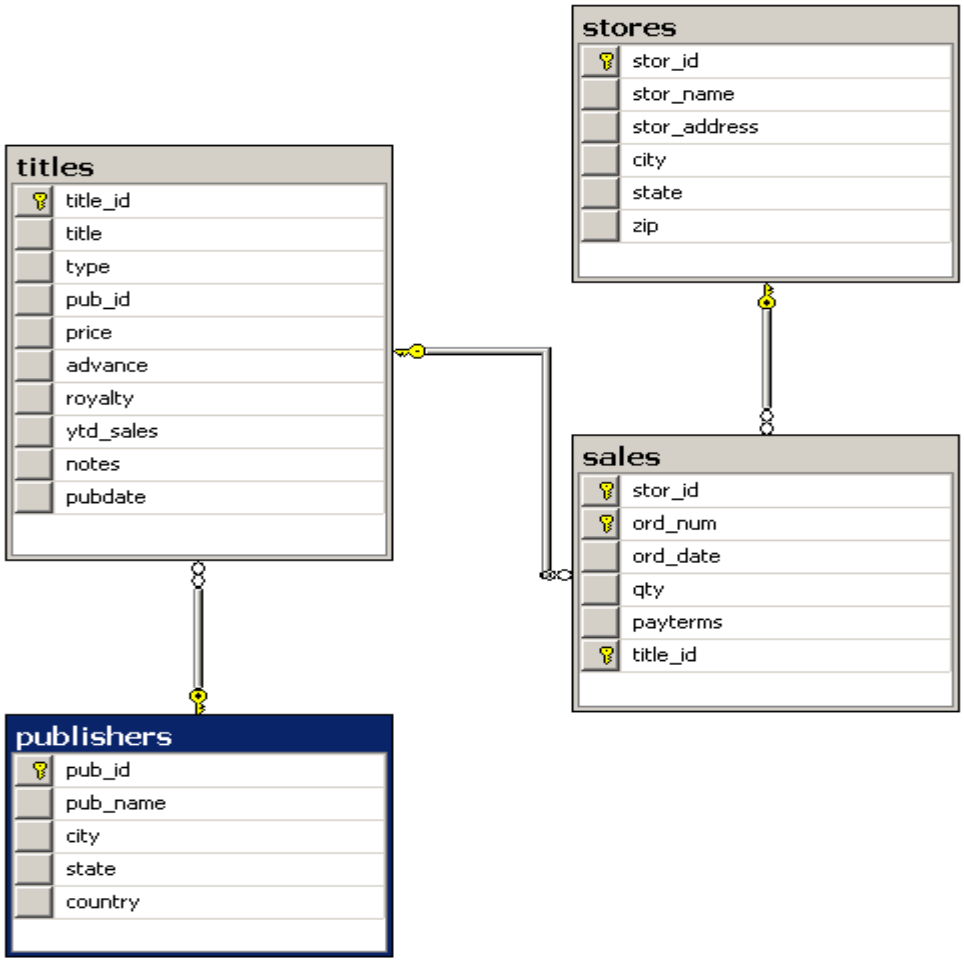
```
- <pubsDB xmlns="http://www.cn.ntua.gr/cdb_self">
```

Τέλος πριν την καταγραφή των δεδομένων του κάθε πίνακα θα ανοίγει ένα tag με το όνομα αυτού καθώς και το Namespace που θα χρησιμοποιείται για την καταγραφή των δεδομένων αυτού του τύπου. Ένα παράδειγμα είναι το ακόλουθο χρησιμοποιώντας τον πίνακα authors της βάσης pubs:

```
- <authors xmlns="http://www.cn.ntua.gr/cdb_self">
```

2.3 Προδιαγραφές Σχέσεων Στο XML

Οι σχέσεις στο XML αρχείο θα καταγράφονται με έναν πιο περίπλοκο τρόπο. Η γενική λογική στηρίζεται στο ότι θα ξεκινάμε από έναν πίνακα βάσης που θα καθορίζεται από το χρήση. Βάση του πίνακα αυτού θα ανοίγει και το αντίστοιχο XML tag, το οποίο θα συμπεριλαμβάνει και το Namespace των Relationships σαν attribute, μέσα στο οποίο θα ενθυλακώνονται tags από πεδία άλλων πινάκων τα οποία αποτελούν primary keys για τον πίνακα βάσης. Στα attributes των ενθυλακωμένων αυτών tags θα δίνονται περισσότερα στοιχεία για τη σχέση μεταξύ των δυο πινάκων στα πεδία FromF, FromT και ThisF που αναφέρονται στα primary field, primary table και foreign field αντίστοιχα. Η λογική των παραπάνω γίνεται πιο εύκολα ξεκάθαρη και κατανοητή με το ακόλουθο παράδειγμα από τη βάση Pubs που αναλυτικά φαίνονται στην ενότητα εφαρμογές.
Από τη βάση Pubs απομονώνουμε το κομμάτι που απεικονίζεται στο παρακάτω σχήμα με τα relationships που περιγράφονται σε αυτό.



Σύμφωνα με το σχήμα αυτό, και παίρνοντας σαν πίνακα βάσης τον πίνακα sales βλέπουμε ότι αυτός αποτελεί foreign table αφενός για τον πίνακα stores και αφετέρου για τον πίνακα titles, ο οποίος με τη σειρά του είναι foreign table για τον πίνακα publishers.

Συνεπώς μετά το άνοιγμα του αρχικού tag που αφορά τον πίνακα βάσης θα ενθυκαλώνονται ουσιαστικά τέσσερις εγγραφές, δυο για τον πίνακα stores – η μία δίνει τις λεπτομέρειες της σχέσης και η δεύτερη είναι το tag <stores/> το οποίο ανοίγει και κλείνει ταυτόχρονα δεδομένου ότι ο πίνακας stores δεν αποτελεί foreign table για κανένα άλλο πίνακα. Αντίθετα, το tag που αφορά που αφορά τον πίνακα titles περιέχει ενθυλακωμένα δεδομένα για τους πίνακες που σχετίζονται με αυτόν.

```
- <sales xmlns="http://www.cn.ntua.gr/relationships_self">
  <cdb:this_from thisF="stor_id" fromT="stores" fromF="stor_id" />
  <stores />
  <cdb:this_from thisF="title_id" fromT="titles" fromF="title_id" />
+ <titles>
  </sales>
```

Κάνοντας expand το tag του πίνακα titles βλέπουμε αναλυτικά τις σχέσεις που αφορούν τον πίνακα titles. Παρατηρούμε ότι το tag του πίνακα publishers ανοίγει και κλείνει ταυτόχρονα δεδομένου ότι αυτό δε σχετίζεται με κανένα άλλο πίνακα. Τελικά, όλο το κομμάτι που αφορά τις σχέσεις φαίνεται παρακάτω:

```
- <sales xmlns="http://www.cn.ntua.gr/relationships_self">
  <cdb:this_from thisF="stor_id" fromT="stores" fromF="stor_id" />
  <stores />
  <cdb:this_from thisF="title_id" fromT="titles" fromF="title_id" />
- <titles>
  <cdb:this_from thisF="pub_id" fromT="publishers" fromF="pub_id" />
  <publishers />
  </titles>
  </sales>
```

2.4 Προδιαγραφές Δεδομένων Βάσης Στο XML

Τα δεδομένα της βάσης θα καταγράφονται στο XML αρχείο ακολουθώντας το ακόλουθο απλό πρότυπο:
Αρχικά θα πρέπει να ανοίγει ένα tag με το όνομα της βάσης δεδομένων ακολουθούμενο από τη συντομογραφία DB. Έπειτα θα ανοίγει ένα tag με το όνομα του πίνακα που καταγράφουμε και τελικά θα ανοίγουν tags με το όνομα record, ένα για κάθε εγγραφή που υπάρχει στον εν λόγω πίνακα. Μέσα σε κάθε tag τύπου record θα ενθυλακώνονται όλα τα στοιχεία της κάθε εγγραφής με διακριτά tags που καθορίζουν το είδος της πληροφορίας που καταγράφεται και το όνομα των οποίον προκύπτει από τα πεδία του κάθε πίνακα. Ένα παράδειγμα φαίνεται στη συνέχεια:

```
- <pubsDB xmlns="http://www.cn.ntua.gr/cdb_self">
- <authors xmlns="http://www.cn.ntua.gr/cdb_self">
- <record>
<au_id>172-32-1176</au_id>
<au_lname>White</au_lname>
<au_fname>Johnson</au_fname>
<phone>408 496-7223</phone>
<address>10932 Bigge Rd.</address>
<city>Menlo Park</city>
<state>CA</state>
<zip>94025</zip>
<contract>True</contract>
</record>
- <record>
<au_id>213-46-8915</au_id>
<au_lname>Green</au_lname>
<au_fname>Marjorie</au_fname>
<phone>415 986-7020</phone>
<address>309 63rd St. #411</address>
<city>Oakland</city>
<state>CA</state>
<zip>94618</zip>
<contract>True</contract>
</record>
```

3. Σχεδιασμός Αλγορίθμου

3.1 Καθορισμός Κλάσεων

Το πρώτο βήμα στην ανάπτυξη μιας τέτοιας εφαρμογής είναι ο εντοπισμός και ο καθορισμός των βασικών διακριτών λειτουργιών που θα επιτελούνται. Αυτές οι βασικές λειτουργίες θα καθορίσουν ταυτόχρονα τον αριθμό αλλά και το είδος των κλάσεων που απαιτούνται για την υλοποίηση της εφαρμογής.
Στην εν λόγω εφαρμογή διαφαίνονται οι ακόλουθες βασικές λειτουργίες:

- Σύνδεση στη βάση δεδομένων
- Ανάγνωση της βάσης δεδομένων
- Κατασκευή του αρχείου XML
- Παρουσίαση των αποτελεσμάτων και στην οθόνη
- Συντονισμός όλων των παραπάνω

Με βάση τα παραπάνω, λοιπόν, θα χρειαστούμε 5 διαφορετικές κλάσεις που θα συνεργάζονται προκειμένου να επιφέρουν το επιθυμητό αποτέλεσμα. Αυτές θα είναι οι ακόλουθες:

- Η κλάση DbConnector
- Η κλάση DbReader
- Η κλάση XmlCreator
- Η κλάση Presenter
- Η κλάση Program

οι οποίες θα επιτελούν αντίστοιχα τις λειτουργίες που αναφέρθηκαν παραπάνω.

3.2 Επιλογή Πεδίων Κλάσεων

Στην ενότητα αυτή θα γίνει μια συζήτηση επάνω στο θέμα της επιλογής των πεδίων της κάθε κλάσης. Θα αναπτυχθεί συνοπτικά το περιεχόμενό τους καθώς και ο λόγος που επιλέχθηκαν ενώ θα δοθούν και μερικά τεχνικά χαρακτηριστικά για αυτές όπου αυτό κριθεί αναγκαίο.

DbConnector

Δεδομένου ότι η κλάση αυτή έχει ως σκοπό να συνδέεται στη βάση δεδομένων θα πρέπει να έχει ως πεδία τις βασικές πληροφορίες που αφορούν την ίδια τη βάση δεδομένων και θα είναι πληροφορίες τις οποίες η κλάση αυτή θα πρέπει να είναι σε θέση να γνωστοποιήσει και σε άλλες κλάσεις. Συνεπώς τα πεδία της θα είναι τα ακόλουθα:

- serverName: Το πεδίο αυτό θα διατηρεί το όνομα του server στον οποίο θα βρίσκεται η βάση. Αν είναι κάτι που θα δηλώνεται από το χρήστη θα δηλωθεί private ώστε οι τιμές που παίρνει να ελέγχονται από μια property. Αυτό γίνεται για λόγους ασφαλείας ώστε να υπάρχει μια μορφή ελέγχου για τις τιμές που δίνονται.
- catalogName: Το πεδίο αυτό θα διατηρεί το όνομα της βάσης δεδομένων. Είναι στοιχείο που θα δίνεται από το χρήστη αλλά θα δηλωθεί private και θα ελέγχεται από την αντίστοιχη property.
- Conn: Είναι το αντικείμενο εκείνο που θα διατηρεί όλες τις πληροφορίες της σύνδεσης. Ανήκει στο namespace System.Data.SqlClient.SqlConnection και αφορά συνδέσεις για τον Microsoft SQL Server.

DbReader

Για την κλάση αυτή θα χρειαστούμε τα ακόλουθα πεδία:

- namesTable: Θα είναι πεδίο τύπου πίνακα και private που θα διατηρεί τα ονόματα όλων των πινάκων της βάσης δεδομένων. Η χρησιμότητά του έγκειται στο γεγονός ότι η αναζήτηση των εγγραφών της βάσης θα γίνονται με εντολές SQL και χρειαζόμαστε τα ονόματα των πινάκων ώστε να πάρουμε τα δεδομένα που αυτοί περιέχουν. Τα δεδομένα του θα ελέγχονται από το ομώνυμο property. Ένα παράδειγμα του πίνακα αυτού για τη βάση δεδομένων pubs φαίνεται παρακάτω:

Name
stores
sales
discounts
titles
royshed
titleauthor
authors
jobs
employee
publishers

pub_info

- relationshipsTable: Θα είναι πεδίο τύπου πίνακα (Αριθμός Σχέσεων)X4 που θα περιέχει τα στοιχεία όλων των σχέσεων του πίνακα (primary key, primary table, foreign key, foreign table). Θα δηλωθεί private και θα ελέγχεται από το αντίστοιχο property. Ένα παράδειγμα του πίνακα αυτού για τη βάση δεδομένων pubs φαίνεται παρακάτω:

Primary Table	Primary Column	Foreign Table	Foreign Column
stores	store_id	discounts	store_id
jobs	job_id	employee	job_id
publishers	pub_id	employee	pub_id
publishers	pub_id	pub_info	pub_id
titles	title_id	royshed	title_id
stores	stor_id	sales	stor_id
titles	title_id	sales	title_id
authors	au_id	title_author	au_id
titles	title_id	title_author	title_id
publishers	pub_id	titles	pub_id

- Myds: Το πεδίο αυτό ανήκει στο namespace System.Data.DataSet και θα αποτελεί μια προγραμματιστική αναπαράσταση της βάσης που θα χρησιμοποιείται στην εφαρμογή ώστε να μη διατηρείται μια μόνιμη σύνδεση με τη βάση. Αυτό κάνει την εφαρμογή πιο αποδοτική δεδομένου ότι οι μόνιμες συνδέσεις δημιουργούν προβλήματα στην προσπέλαση μιας βάσης από μεγάλο αριθμό χρηστών.

XmlCreator

Για την κλάση XmlCreator τα fields που θα χρησιμοποιηθούν θα είναι τα ακόλουθα:

- catalogName: Πρόκειται για το όνομα της βάσης δεδομένων όπως έχει ξαναχρησιμοποιηθεί και σε προηγούμενη κλάση. Δηλώνεται private και η τιμές του προκύπτουν αποκλειστικά και μόνο από άλλες κλάσεις εσωτερικά. Στην εν λόγω κλάση χρησιμοποιείται προκειμένου να παίρνει το όνομά του το XML αρχείο.
- startingTable: Πρόκειται για τον πίνακα βάσης από τον οποίο ξεκινάμε να γράφουμε τις σχέσεις της βάσης δεδομένων στο XML αρχείο.
- fStream: Τελικά, μας είναι αναγκαίο όπως είναι αναμενόμενο, ένα αντικείμενο τύπου StreamWriter έτσι ώστε να μας δίνει τη δυνατότητα να γράφουμε δεδομένα και πληροφορίες του προγράμματος σε ένα εξωτερικό αρχείο και συγκεκριμένα στο XML αρχείο που κατασκευάζουμε.

Presenter

Για την κλάση Presenter η οποία και θα παρουσιάζει όλα τα δεδομένα της διαδικασίας στην οθόνη (κυρίως και για λόγους debugging και ελέγχου της ροής της εφαρμογής) θα χρησιμοποιήσουμε τα παρακάτω πεδία:

- catalogName: Πρόκειται για το όνομα της βάσης δεδομένων.
- namesTable: Είναι ο πίνακας που περιέχει τα ονόματα όλων των πινάκων της βάσης δεδομένων.
- relationshipsTable: Ο πίνακας αυτός διατηρεί τις πληροφορίες για τις σχέσεις που υπάρχουν στη βάση που χρησιμοποιείται.
- myds: Είναι το αντικείμενο που διατηρεί μια προγραμματιστική αναπαράσταση της βάσης δεδομένων ώστε να αποφεύγουμε τη μόνιμη σύνδεση στη βάση δεδομένων. Αυτό το connectionless πρότυπο που χρησιμοποιείται συμβάλλει στην αποφυγή συμφορήσεων κατά της σύνδεση πολλών χρηστών σε έναν εξυπηρετητή που διατηρεί μια βάση δεδομένων.
- fStream: Αντικείμενο τύπου StreamWriter για εγγραφή δεδομένων σε ένα εξωτερικό αρχείο.

Program

Η κλάση Program όπως είναι αναμενόμενο δεν περιέχει δικά της πεδία δεδομένου ότι ρόλος της είναι μόνο ο συντονισμός των υπόλοιπων κλάσεων.

3.3 Επιλογή Μεθόδων Κλάσεων

Οι μέθοδοι που θα πλαισιώσουν την κάθε κλάση είναι άρρηκτα συνδεδεμένες με τις λειτουργίες που θα επιτελεί η κάθε κλάση. Σε αυτή την ενότητα, λοιπόν, θα εντοπίσουμε τις λειτουργίες αυτές και θα καθορίσουμε τις μεθόδους που θα χρειαστούν για το σκοπό αυτό.

DbConnector

Δεδομένου ότι πρόκειται για την κλάση που αφορά τη σύνδεση στη βάση θα χρησιμοποιεί δύο βασικές μεθόδους:

- Μία μέθοδο για να πραγματοποιεί τη σύνδεση. Πρόκειται για τη μέθοδο ConnectToSql τύπου SqlConnection που θα έχει σαν παραμέτρους το όνομα του εξυπηρετητή και το όνομα της βάσης ενώ αυτή θα αρχικοποιεί ένα αντικείμενο τύπου SqlConnection (conn) το οποίο θα περιέχει όλες τις πληροφορίες της σύνδεσης.
- Μια μέθοδο που θα κλείνει τη σύνδεση και θα απελευθερώνει όλους τους πόρους που έχουν δεσμευτεί. Αυτή θα είναι η CloseSqlConnection, η οποία χρειάζεται το αντικείμενο conn που αναφέρθηκε προηγουμένως ως μοναδική παράμετρο προκειμένου να «κλείσει» τη σύνδεση αυτή.

DbReader

Η κλάση DbReader έχει ως βασικό της μέλημα της ανάγνωση της βάσης δεδομένων και τη δημιουργία μιας προγραμματιστικής αναπαράστασης αυτής σε ένα DataSet, την οποία και θα χρησιμοποιήσουμε αργότερα στην κλάση XmlCreator για την κατασκευή του XML αρχείου. Έτσι, λοιπόν, είναι αναγκαίες οι ακόλουθες μέθοδοι:

- Η μέθοδος GetNamesTable η οποία με χρήση εντολών t-SQL ανασύρει τα ονόματα όλων των πινάκων της βάσης δεδομένων με στόχο στη συνέχεια να ανακτηθούν και όλα τα δεδομένα από τους πίνακες αυτούς. Η μόνη παράμετρος που χρειάζεται η μέθοδος αυτή είναι το αντικείμενο conn που περιέχει όλες τις πληροφορίες για τη σύνδεση στη βάση δεδομένων.
- Η μέθοδος GetRelationshipsTable λειτουργεί αντίστοιχα με τη μέθοδο που περιγράφηκε προηγουμένως με τη μόνη διαφορά ότι αυτή ανασύρει όλες τις σχέσεις που υπάρχουν στη βάση δεδομένων.
- Η μέθοδος CreateDataset κατασκευάζει το αντικείμενο τύπου DataSet myds το οποία είναι μια αναπαράσταση της βάσης που εξετάζουμε. Η μέθοδος αυτή χρησιμοποιεί ως παραμέτρους αφενός τη σύνδεση στη βάση, άρα το αντικείμενο conn, και αφετέρου τον πίνακα namesTable ώστε με εντολές SQL του τύπου SELECT και το όνομα του κάθε πίνακα να ανασύρουμε όλα τα δεδομένα της βάσης και να τα τοποθετήσουμε στο Dataset μας.

XmlCreator

Η κλάση XmlCreator κατασκευάζει το αρχείο XML και για αυτό το λόγο χρειάζεται τις παρακάτω μεθόδους:

- CreateXmlFile: Η μέθοδος αυτή δημιουργεί το XML αρχείο δίνοντας του το όνομα της βάσης δεδομένων. Είναι τύπου void διότι δε χρειάζεται καμία παράμετρο για τη λειτουργία της.
- CloseXmlFile: Στη μέθοδο αυτή γράφεται το τελευταίο tag κλεισίματος του XML αρχείου και ταυτόχρονα απελευθερώνονται οι πόροι του συστήματος.
- WriteXmlHeaders: Κατά την εκτέλεση της μεθόδου αυτής γράφονται οι επικεφαλίδες του XML αρχείου. Οι επικεφαλίδες αυτές είναι στατικές και ίδιες για όλα τα αρχεία.
- WriteXmlRelationships: Η μέθοδος αυτή εξετάζει αν υπάρχει σχέσεις στη βάση προκειμένου να τις περάσει στο XML αρχείο και συγκεκριμένα αν υπάρχει στις σχέσεις (και μάλιστα ως foreign key) ο πίνακας βάσης. Αν αυτό συμβαίνει τότε ο έλεγχος ροής οδηγείται στη μέθοδο XmlRelationshipsWriter.
- XmlRelationshipsWriter: Στη μέθοδο αυτή με μια αναδρομική διαδικασία καταγράφονται οι σχέσεις της βάσης δεδομένων ακολουθώντας τις απαιτούμενες προδιαγραφές. Η μέθοδος αυτή δηλώνεται private δεδομένου ότι εκτελεί μια εσωτερική λειτουργία και για λόγους ασφαλείας και αξιόπιστης κατασκευής λογισμικού δεν πρέπει να είναι προσβάσιμη από μεθόδου εκτός της εν λόγω κλάσης.
- WriteXmlData: Στην τελευταία αυτή μέθοδο καταγράφονται όλα τα δεδομένα των εγγραφών που βρίσκονται στο αντικείμενο myds. Αυτό επιτυγχάνεται με επαναληπτικές διαδικασίες που προσπελαύνουν όλα τα δεδομένα σειριακά.

Presenter

Η κλάση Presenter παρουσιάζει όλα τα δεδομένα που χρησιμοποιούνται στο πρόγραμμα. Αυτό γίνεται με χρήση των παρακάτω μεθόδων:

- GiveNamesTable: Παρουσιάζει τον πίνακα που έχουμε κατασκευάσει χρησιμοποιώντας t-SQL και περιέχει τα ονόματα όλων των πινάκων της βάσης.
- GiveRelationshipsTable: Παρουσιάζει τον πίνακα που έχουμε κατασκευάσει χρησιμοποιώντας t-SQL και περιέχει όλες τις σχέσεις τις βάσης.
- GiveDataSet: Παρουσιάζει της προγραμματιστική αναπαράσταση της βάσης δεδομένων που βρίσκεται στο αντικείμενο myds.
- GiveXmlOutput: Παρουσιάζει το XML αρχείο που έχει κατασκευαστεί στην οθόνη του χρήστη.

Program

Η κλάση Program ως «γενικός συντονιστής» όλων των υπόλοιπων κλάσεων χρειάζεται μια μόνο μέθοδο, τη μέθοδο main, από την οποία και ξεκινάει η λειτουργία του προγράμματος.

3.4 Υλοποίηση Κλάσεων/Βασικών Μεθόδων

Στην παράγραφο αυτή με χρήση είτε ψευδοκώδικα είτε διαγραμμάτων ροής θα ξεκαθαρίσουμε τον τρόπο με τον οποίο λειτουργούν κάποιες μέθοδοι προκειμένου να επιτελέσουν τις λειτουργίες τους.

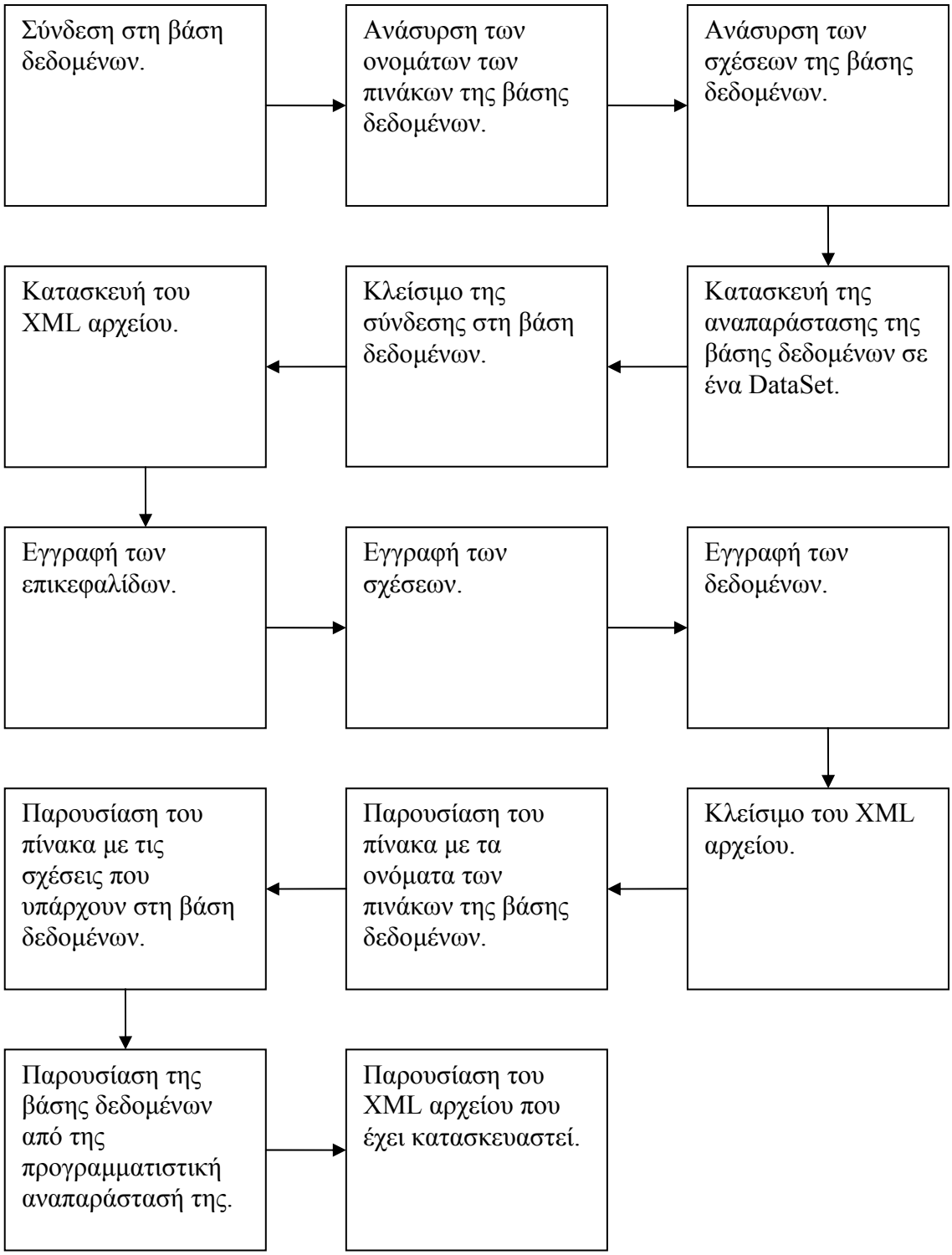
XmlCreator

→WriteXmlRelationships/XmlRelationshipsWriter
Η εγγραφή των σχέσεων γίνεται βάση του ψευδοκώδικα που ακολουθεί:

1. *Έλεγχος αν ο πίνακας βάσης υπάρχει στα foreign tables του πίνακα σχέσεων.*
2. *Εάν δεν υπάρχει η μέθοδος τερματίζεται. Εάν υπάρχει ο έλεγχος ροής συνεχίζει στην κλάση XmlRelationshipsWriter με την παράμετρο tester=0 που υποδεικνύει ότι μπαίνουμε για πρώτη φορά.*
3. *Αν η παράμετρος tester=0 ανοίγει το tag για το ζητούμενο πίνακα και μαζί του γράφονται και η επικεφαλίδες για το κατάλληλο namespace. Αν η παράμετρος tester=1 ανοίγει το tag για το ζητούμενο πίνακα χωρίς το namespace(ο έλεγχος ροής έχει καταλήξει εδώ από αναδρομή και όχι από τη μέθοδο WriteXmlRelationships).*
4. *Ψάχνουμε το ζητούμενο πίνακα στα foreign tables του πίνακα των σχέσεων.*
5. *Αν δε βρεθεί άλλο κλείνει το tag του ζητούμενου πίνακα και τελειώνει η συνάρτηση.*
6. *Αν βρεθεί άλλο γράφονται τα στοιχεία της σχέσης βάση των προδιαγραφών και ανοίγει ένα καινούργιο tag για τον εν λόγω πίνακα. Έπειτα καλείται η ίδια συνάρτηση με όρισμα το primary table που αντιστοιχεί στο foreign table που βρέθηκε και ο έλεγχος ροής μεταβαίνει στο βήμα 3.*

Program

Στο παρακάτω σχήμα φαίνεται το διάγραμμα ροής που ακολουθείται από τη μέθοδο main της κλάσης Program προκειμένου να συντονίσει τις υπόλοιπες κλάσεις και να επιφέρει το επιθυμητό αποτέλεσμα.



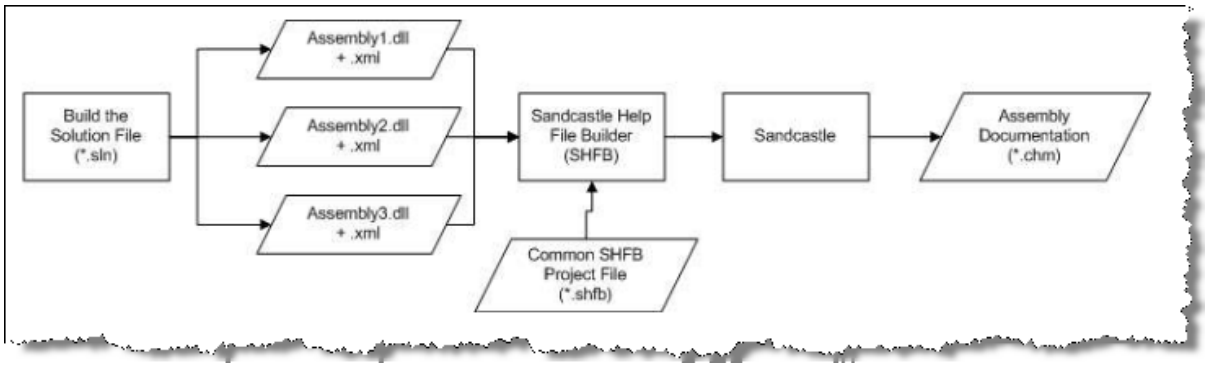
4. Σύνταξη Documentation

4.1 Η Εφαρμογή Microsoft Sandcastle

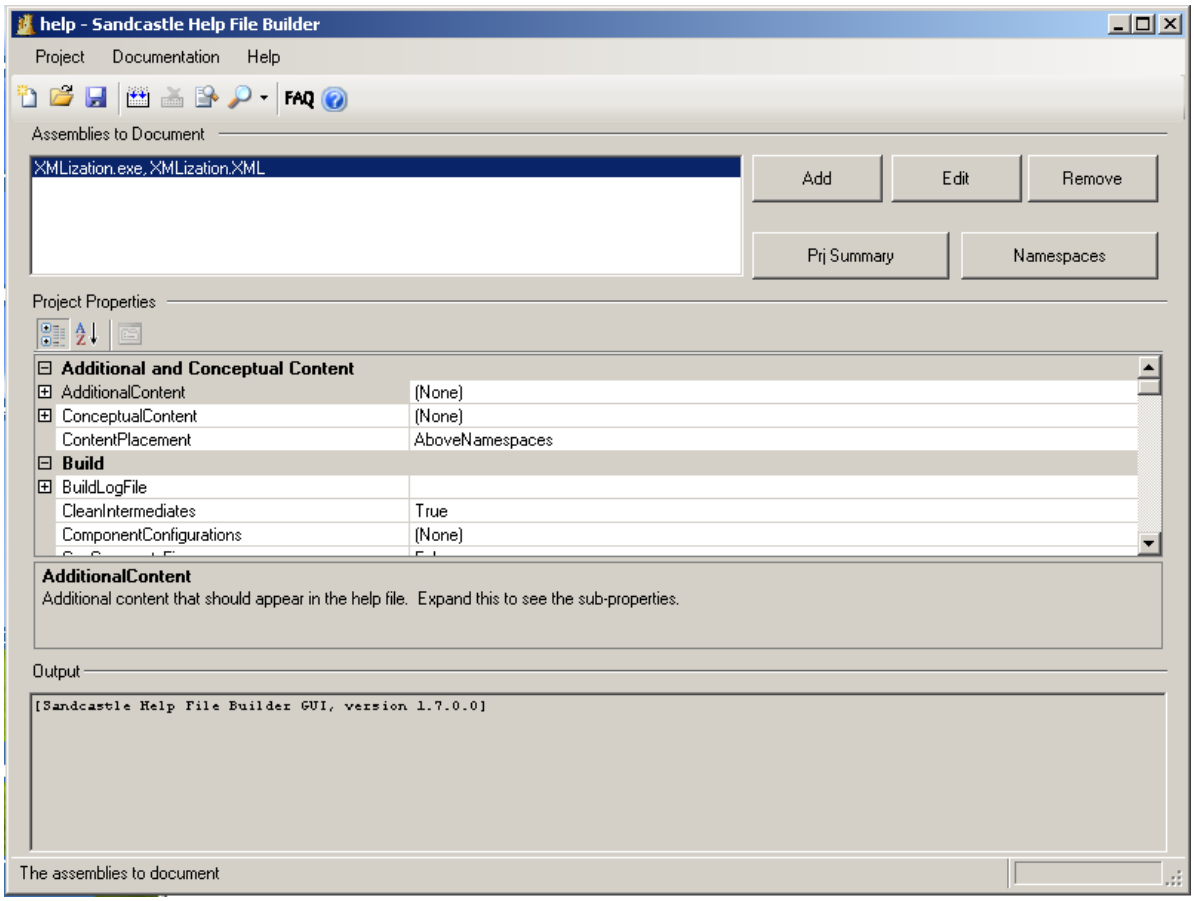
Το Sandcastle είναι μια γεννήτρια αρχείων τεκμηρίωσης (Documentation) της Microsoft η οποία παράγει αυτόματα τεκμηρίωση στο ύφος της επίσημης τεκμηρίωσης που χρησιμοποιείται από τον επίσημο φορέα της Microsoft, MSDN (Microsoft Developers Network). Η τεκμηρίωση αυτή προκύπτει αφενός από πληροφορίες που παρέχονται από το ίδιο το .NET και αφετέρου από σχόλια τεκμηρίωσης σε μορφή XML που εντοπίζονται στον πηγαίο κώδικα της εφαρμογής που έχει αναπτυχθεί. Το Sandcastle είναι ένα σύνολο εντολών, αρχείων διαμόρφωσης και αρχείων μετασχηματισμού XSLT τα οποία συνεργάζονται προκειμένου να μετατρέψουν την βασισμένη σε XML τεκμηρίωση σε αρχεία βοήθειας κατάλληλα να προσπελαστούν σε ένα σύστημα βοήθειας. Τυπικά χρησιμοποιείται για να παράγει αυτόματα κατάλληλη για το διαδίκτυο και συμβατή με XML τεκμηρίωση τύπου HTML σε ένα από τα τρία ενσωματωμένα στυλ παρουσίασης. Επειδή ωστόσο η παρούσα κατάσταση του Sandcastle είναι αρκετά περίπλοκη στην χρήση της έχουν αναπτυχθεί διάφορα εργαλεία και scripts που αυτοματοποιούν αυτή τη διαδικασία. Μια από αυτές τις εφαρμογές είναι και το Sandcastle Help File Builder.

4.2 Η Εφαρμογή Sandcastle Help File Builder

Το Sandcastle Help File Builder είναι ένα φιλικό προς τον χρήστη εργαλείο που παρέχει γραφικές επιλογές για την κατασκευή της τεκμηρίωσης σύμφωνα με τις επιθυμίες του developer.
Ο τρόπος που λειτουργεί η εφαρμογή αυτή φαίνεται συνοπτικά στο παρακάτω διάγραμμα που ξεκινά από τη δημιουργία ενός solution file στο visual studio μέχρι την παραγωγή της τεκμηρίωσης.



Στην επόμενη εικόνα παρουσιάζεται το GUI (Graphical User Interface) που διαθέτει η εφαρμογή Sandcastle Help File Builer.



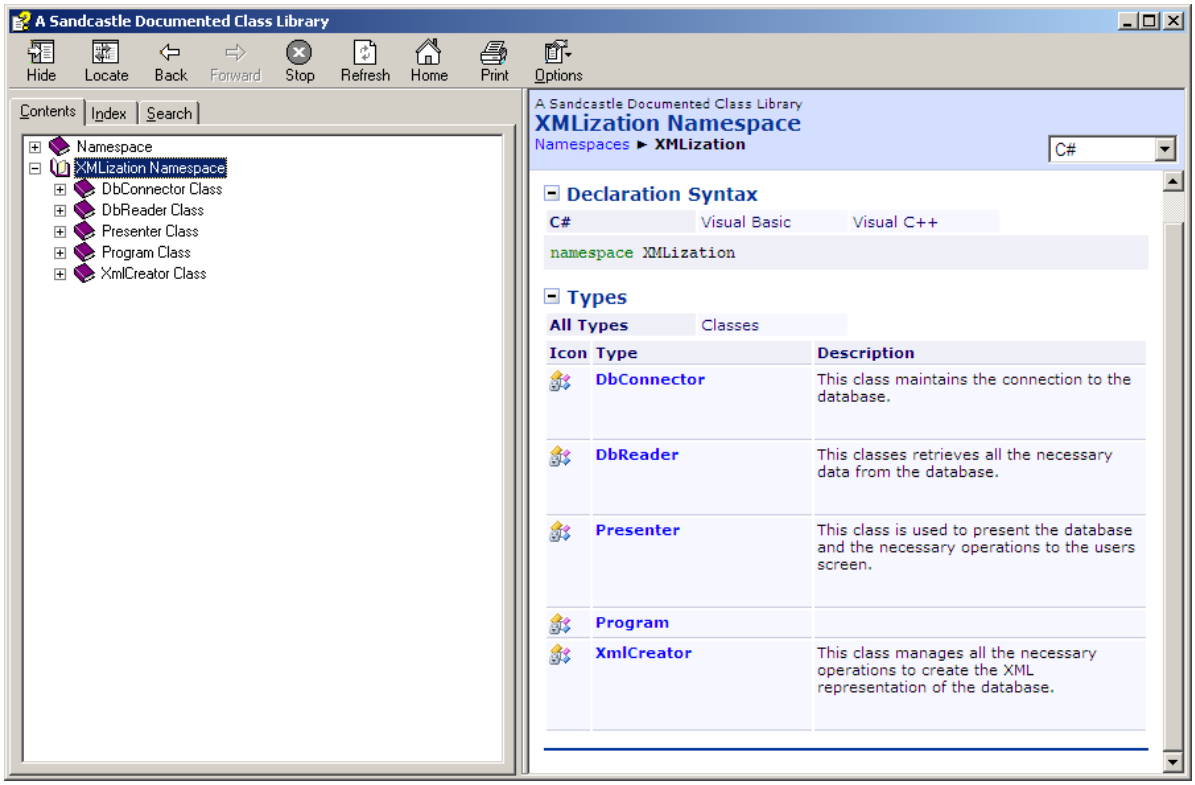
4.3 Χρήση XML Tags Στον Κώδικα

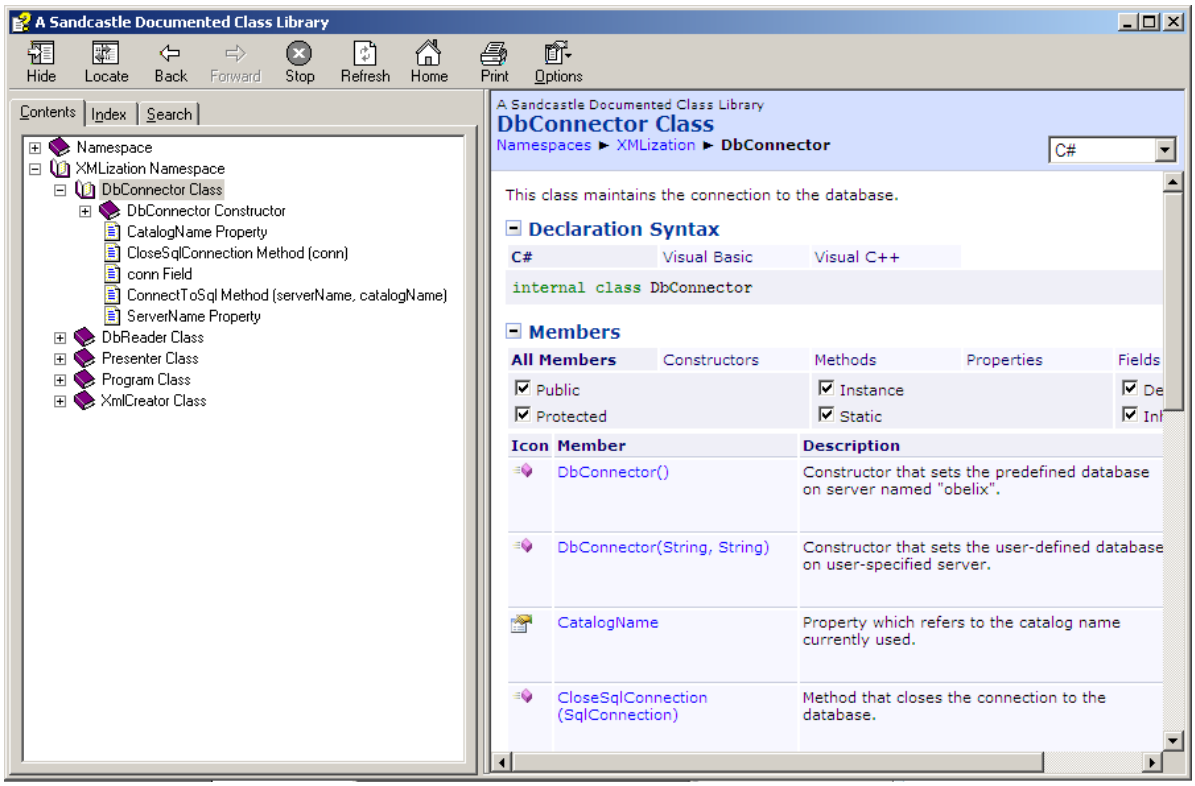
Προκειμένου λοιπόν να παραχθεί αυτόματα η τεκμηρίωση της εφαρμογής που αναπτύχθηκε ενσωματώσαμε XML tags στον κώδικα τα οποία περιγράφουν κάθε στοιχείο που εμπεριέχεται: κλάσεις, μεθόδους, μεταβλητές, properties κλπ. Τα XML tags αυτά προηγούνται πάντα του στοιχείου που περιγράφουν και έχουν την ακόλουθη σύνταξη:

```
        /// <summary>
        /// Constructor that sets the user-defined database on user-
specified server.
        /// </summary>
        /// <param name="server">Server Name</param>
        /// <param name="catalog">Catalog Name</param>
        public DbConnector(string server, string catalog)
        {
            ServerName = server;
            CatalogName = catalog;
        }
```

4.4 Παρουσίαση Αρχείου Documentation

Μετά τη διαδικασία αυτή παράχθηκε ένα αρχείο τύπου CHTML Help File (Compiled HTML Help File) με τη χρήση των XML tags του κώδικα το οποίο είναι ιδιαίτερα φιλικό στο χρήση ώστε εύκολα και γρήγορα, με χρήση υπερσυνδέσμων να μπορεί να περιηγηθεί μέσα σε αυτό και να εντοπίσει τις πληροφορίες που θέλει όσον αφορά τη λειτουργικότητα της εφαρμογής. Αξιοσημείωτο είναι ότι για όλα τα στοιχεία έχει χρησιμοποιηθεί ο προκαθορισμένος συμβολισμός(πχ για τις μεθόδους το μωβ τετράγωνο κλπ) που χρησιμοποιεί και η ίδια η Microsoft στην τεκμηρίωση όλου του .NET. Αποτέλεσμα αυτού είναι να υπάρχει μια ενιαία μορφή τεκμηρίωσης που ακολουθείται από όλους τους developers και είναι εύκολα κατανοητή σε όλους τους χρήστες. Στη συνέχεια φαίνονται δύο εικόνες από το documentation που παράχθηκε για την εφαρμογή που αναπτύχθηκε:

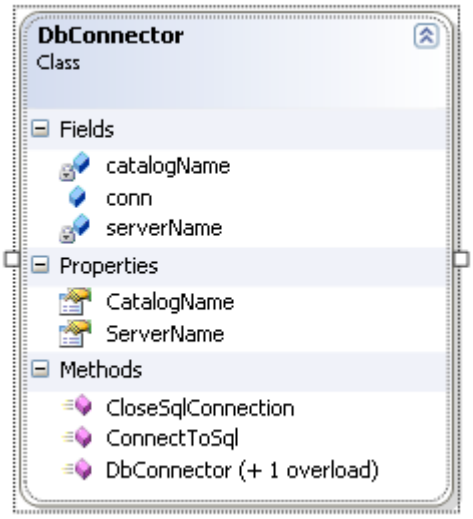



A Sandcastle Documented Class Library
Hide
Locate
Back
Forward
Stop
Refresh
Home
Print
Options
Contents
Index
Search
Namespace
XMLization Namespace
DbConnector Class
DbConnector Constructor
CatalogName Property
CloseSqlConnection Method (conn)
conn Field
ConnectToSql Method (serverName, catalogName)
ServerName Property
DbReader Class
Presenter Class
Program Class
XmlCreator Class
A Sandcastle Documented Class Library
DbConnector Class
Namespaces ► XMLization ► DbConnector
C#
This class maintains the connection to the database.
Declaration Syntax
C#
Visual Basic
Visual C++
internal class DbConnector
Members
All Members
Constructors
Methods
Properties
Fields
Public
Protected
Instance
Static
Icon
Member
Description
DbConnector()
Constructor that sets the predefined database on server named "obelix".
DbConnector(String, String)
Constructor that sets the user-defined database on user-specified server.
CatalogName
Property which refers to the catalog name currently used.
CloseSqlConnection (SqlConnection)
Method that closes the connection to the database.

5. Συγγραφή Κώδικα

Στην ενότητα αυτή παρουσιάζουμε τον κώδικα που δημιουργήσαμε συνοδευόμενο και με το αντίστοιχο UML διάγραμμα για κάθε κλάση.

5.1 Η Κλάση DbConnector

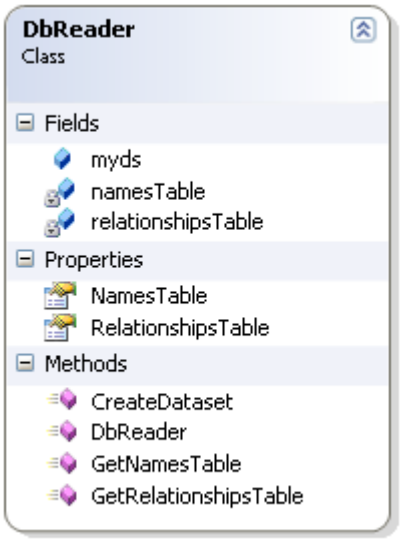



```
using System;
using System.Collections.Generic;
using System.Text;
using System.Timers;

namespace XMLization
{
    /// <summary>
    /// This class maintains the connection to the database.
    /// </summary>
    class DbConnector
    {

        private static string serverName, catalogName;
        /// <summary>
        /// The connection object.
        /// </summary>
        public System.Data.SqlClient.SqlConnection conn;

        /// <summary>
        /// Constructor that sets the predefined database on server named
"obelix".
        /// </summary>
        public DbConnector()
        {
            ServerName = "obelix";
            CatalogName = "carsdatabase";
        }


        /// <summary>
```

```
        /// Constructor that sets the user-defined database on user-
specified server.
        /// </summary>
        /// <param name="server">Server Name</param>
        /// <param name="catalog">Catalog Name</param>
        public DbConnector(string server, string catalog)
        {
            ServerName = server;
            CatalogName = catalog;
        }

        /// <summary>
        /// Property which refers to server name currently used.
        /// </summary>
        public string ServerName
        {
            get
            {
                return serverName;
            }

            set
            {
                serverName = value;
            }
        }

        /// <summary>
        /// Property which refers to the catalog name currently used.
        /// </summary>
        public string CatalogName
        {
            get
            {
                return catalogName;
            }

            set
            {
                catalogName = value;
            }
        }

        /// <summary>
        /// Method which connects to the appropriate database.
        /// </summary>
        /// <param name="serverName">The name of the server currently
used.</param>
        /// <param name="catalogName">The name of the catalog currently
used.</param>
        /// <returns></returns>
        public System.Data.SqlClient.SqlConnection ConnectToSql(string
serverName, string catalogName)
        {
            System.Data.SqlClient.SqlConnection conn = new
System.Data.SqlClient.SqlConnection();

            conn.ConnectionString = "integrated security=SSPI;data
source=" + serverName + ";" + "persist security info=False;initial
catalog=" + catalogName;
            try
```

```
            {
                conn.Open();
                Console.WriteLine("CONNECTION INITIALIZED\n {0}
{1}",conn.ServerVersion,conn.State);
                return conn;
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to connect to data source.");
                Environment.Exit(-1);
                return null;
            }
        }//end method ConnectToSql

        /// <summary>
        /// Method that closes the connection to the database.
        /// </summary>
        /// <param name="conn"></param>
        public void CloseSqlConnection(System.Data.SqlClient.SqlConnection
conn)
        {
            conn.Close();
            Console.WriteLine("CONNECTION CLOSED");
        }//end method CloseSqlConnection


    }
}
```

5.2 Η Κλάση DbReader



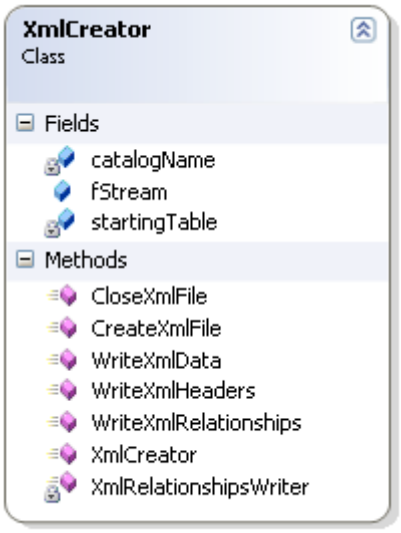

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Collections;

namespace XMLization
{
    /// <summary>
    /// This classes retrieves all the necessary data from the database.
    /// </summary>
    class DbReader
    {
        /// <summary>
        /// The dataset that maintains a representation of the database.
        /// </summary>
        public System.Data.DataSet myds;
        /// <summary>
        /// A table with the names of all the tables of the database.
        /// </summary>
        private string[] namesTable;
        /// <summary>
        /// A table of 4 columns(foreign table, foreign field, primary
table, primary field) with all the relationships of the database.
        /// </summary>
        private string[,] relationshipsTable;

        /// <summary>
        /// Parameterless constructor.
        /// </summary>
        public DbReader()
        {
        }

        /// <summary>
        /// Property which preserves the names of all the tables.
        /// </summary>
        public string[] NamesTable
        {
```

```
            get
            {
                return namesTable;
            }

            set
            {
                namesTable = value;
            }
        }

        /// <summary>
        /// Property which preserves all the relationships in a matrix of
4 columns.
        /// </summary>
        public string[,] RelationshipsTable
        {
            get
            {
                return relationshipsTable;
            }

            set
            {
                relationshipsTable = value;
            }
        }

        /// <summary>
        /// Method which retrieves the names of the tables from the
database using the appropriate SQL command. Then stored in NamesTable
matrix.
        /// </summary>
        /// <param name="conn"></param>
        public void GetNamesTable(System.Data.SqlClient.SqlConnection
conn)
        {
            string command = "sp_tables @table_type = \"'TABLE'\"";

            System.Data.DataSet myds = new System.Data.DataSet();
            System.Data.SqlClient.SqlDataAdapter myadap = new
System.Data.SqlClient.SqlDataAdapter(command, conn);

            myadap.Fill(myds);

            //ELEGXOS ADEIOU
            if (myds.Tables.Count == 0 || myds.Tables[0].Rows.Count == 0)
                namesTable = null;
            else
            {
                ArrayList myList = new ArrayList();

                for (int i = 0; i != myds.Tables[0].Rows.Count; i++)
                {
                    if (String.Format("{0}", myds.Tables[0].Rows[i][2]) ==
"sysdiagrams") continue;
                    myList.Add(String.Format("{0}",
myds.Tables[0].Rows[i][2]));
                }

                namesTable = new string[myList.Count];
```

```
                for (int k = 0; k != myList.Count; k++)
                    namesTable[k] = (string)myList[k];
            }
        }//end method GetNamesTable



        /// <summary>
        /// Method which retrieves all the relationships from the
database. Then stored in RelationshipsTable matrix.
        /// </summary>
        /// <param name="conn"></param>
        public void
GetRelationshipsTable(System.Data.SqlClient.SqlConnection conn)
        {
            string command = "select tblAll.table_name as
PrimaryTableName, tblAll.column_name as PrimaryTableColumn,
tblFK.table_name as ForeignKeyTable, tblFK.column_name as ForeignKeyColumn
from information_schema.constraint_column_usage tblAll inner join
information_schema.referential_constraints tblAllFK on
tblAllFK.unique_constraint_name = tblAll.constraint_name inner join
information_schema.constraint_column_usage tblFK on
tblAllFK.constraint_name=tblFK.constraint_name";

            System.Data.DataSet myds = new System.Data.DataSet();
            System.Data.SqlClient.SqlDataAdapter myadap = new
System.Data.SqlClient.SqlDataAdapter(command, conn);

            myadap.Fill(myds);

            //ELEGXOS ADEIOU
            if (myds.Tables.Count == 0 || myds.Tables[0].Rows.Count == 0)
                relationshipsTable = null;
            else
            {
                relationshipsTable = new string[myds.Tables[0].Rows.Count,
myds.Tables[0].Columns.Count];

                for (int i = 0; i != myds.Tables[0].Rows.Count; i++)
                    for (int j = 0; j != myds.Tables[0].Columns.Count;
j++)
                    {
                        relationshipsTable[i, j] = String.Format("{0}",
myds.Tables[0].Rows[i][j]);
                    }
            }
        }//end method GetRelationshipsTable

        /// <summary>
        /// Method which creates the Dataset of the database. It is a
representation of the whole database.
        /// </summary>
        /// <param name="conn"></param>
        /// <param name="namesTable"></param>
        public void CreateDataset(System.Data.SqlClient.SqlConnection
conn, string[] namesTable)
        {
            string[] commandTable = new string[namesTable.Length];
            myds = new System.Data.DataSet();
```

```
            if (namesTable != null)
            {
                for (int l = 0; l != namesTable.Length; l++)
                    commandTable[l] = "SELECT * from [" + namesTable[l] +
"]";

                System.Data.SqlClient.SqlDataAdapter myadap = new
System.Data.SqlClient.SqlDataAdapter();

                System.Data.SqlClient.SqlCommand selectCmd =
conn.CreateCommand();

                for (int m = 0; m != commandTable.Length; m++)
                {
                    selectCmd.CommandType = System.Data.CommandType.Text;
                    selectCmd.CommandText = commandTable[m];
                    myadap.SelectCommand = selectCmd;

                    myadap.Fill(myds, namesTable[m]);
                }
            }
            else
            {
                myds = null;
            }

        }//end method CreateDataset

    }
}
```

5.3 Η Κλάση XmlCreator



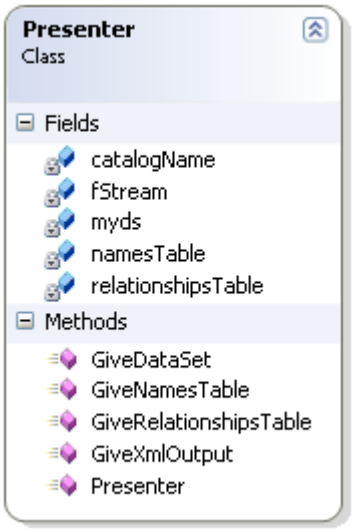

```
using System;
using System.Collections.Generic;
using System.Text;

namespace XMLization
{
    /// <summary>
    /// This class manages all the necessary operations to create the XML
representation of the database.
    /// </summary>
    class XmlCreator
    {
        /// <summary>
        /// StreamWriter object used to print on the user's screen.
        /// </summary>
        public System.IO.StreamWriter fStream;
        private string catalogName;
        private string startingTable;

        /// <summary>
        /// XmlCreator Constructor.
        /// </summary>
        /// <param name="filename">The name of the XML file. We use the
same as the name of database's catalog.</param>
        /// <param name="relationshipsStartingTable">The table with which
we start to write the XML representation of the relationships.</param>
        public XmlCreator(string filename, string
relationshipsStartingTable)
        {
            catalogName = filename;
            startingTable = relationshipsStartingTable;
        }

        /// <summary>
        /// Method which creates the XML file.
        /// </summary>
        public void CreateXmlFile()
        {
```

```
            fStream = new System.IO.StreamWriter(catalogName + ".xml");
        }//end method CreateXmlFile

        /// <summary>
        /// Method which closes the XML file and releases all resources.
        /// </summary>
        /// <param name="fStream">StreamWriter object used to print on the
user's screen.</param>
        public void CloseXmlFile(System.IO.StreamWriter fStream)
        {
            fStream.WriteLine("</xsl:stylesheet>");
            fStream.Close();
        }//end method CloseXmlFile

        /// <summary>
        /// Method which writes the necessary XML headers.
        /// </summary>
        /// <param name="fStream">StreamWriter object used to print on the
user's screen.</param>
        /// <param name="myds">Dataset used to represent the database in
the programm.</param>
        public void WriteXmlHeaders(System.IO.StreamWriter fStream,
System.Data.DataSet myds)
        {
            string header = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<xsl:stylesheet version=\"1.0\"
xmlns:xsl=\"http://www.w3.org/1999/XSL/Transform\"
xmlns:cdb=\"http://www.cn.ntua.gr/cdb\"
xmlns:pf=\"http://www.cn.ntua.gr/cdb_self\"
xmlns:rel=\"http://www.cn.ntua.gr/relationships_self\">";

            fStream.WriteLine(header);
        }//end method WriteXmlHeaders

        /// <summary>
        /// Recursive Method which writes the relationships to the XML
file.
        /// </summary>
        /// <param name="fStream">StreamWriter object used to print on the
user's screen.</param>
        /// <param name="relationshipsTable">The table with all the
relationships in the database.</param>
        /// <param name="searchingTable">The table we are investigating at
this specific moment.</param>
        /// <param name="tester">Variable used to identify whether it is
the first loop or not.</param>
        private void XmlRelationshipsWriter(System.IO.StreamWriter
fStream, string[,] relationshipsTable, string searchingTable, int tester)
        {
            if (tester == 0)
            {
                fStream.WriteLine("<" + searchingTable + "
xmlns=\"http://www.cn.ntua.gr/relationships_self\">");
            }
            else
            {
                fStream.WriteLine("<" + searchingTable + ">");
            }

            for (int i = 0; i != relationshipsTable.GetLength(0); i++)
            {
```

```
                if (relationshipsTable[i, 2] == searchingTable)
                {
                    fStream.WriteLine("<cdb:this_from thisF=\"" +
relationshipsTable[i, 3] + "\" fromT=\"" + relationshipsTable[i, 0] + "\"
fromF=\"" + relationshipsTable[i, 1] + "\" /> ");
                    XmlRelationshipsWriter(fStream, relationshipsTable,
relationshipsTable[i, 0], 1);
                }
            }
            fStream.WriteLine("</" + searchingTable + ">");

        }//end method XmlRelationshipsWriter

        /// <summary>
        /// Method which initiates the writing of the relationships to the
XML file.
        /// </summary>
        /// <param name="fStream">StreamWriter object used to print on the
user's screen.</param>
        /// <param name="relationshipsTable">The table with all the
relationships in the database.</param>
        public void WriteXmlRelationships(System.IO.StreamWriter fStream,
string[,] relationshipsTable)
        {
            //ELEGXOS AN IPARXEI AUTO TO TABLE STA RELATIONSHIPS KAI
EPISTROFI AN DEN IPARXEI
            int escape = 0;

            for (int i = 0; i != relationshipsTable.GetLength(0); i++)
            {
                if (relationshipsTable[i, 2] == startingTable)
                {
                    escape = 1;
                }
            }

            if (escape == 0) return;

            XmlRelationshipsWriter(fStream, relationshipsTable,
startingTable, 0);
        }//end method WriteXmlRelationships

        /// <summary>
        /// Method which write all the database records to the XML file.
        /// </summary>
        /// <param name="fStream">StreamWriter object used to print on the
user's screen.</param>
        /// <param name="myds">Dataset used to represent the database in
the programm.</param>
        public void WriteXmlData(System.IO.StreamWriter fStream,
System.Data.DataSet myds)
        {
            fStream.WriteLine("<" + catalogName + "DB" + "
xmlns=\"http://www.cn.ntua.gr/cdb_self\">"); //ANOIGMA ARXIKOU MEGALOU TAG

            for (int i = 0; i != myds.Tables.Count; i++)
            {
                fStream.WriteLine("<" + myds.Tables[i].TableName + "
xmlns=\"http://www.cn.ntua.gr/cdb_self\">");//ANOIGMA TABLE TAG
                for (int j = 0; j != myds.Tables[i].Rows.Count; j++)
                {
```

```
                    fStream.WriteLine("<record>");//ANOIGMA RECORD
                    for (int k = 0; k != myds.Tables[i].Columns.Count;
k++)
                    {
                        //EGGRAFI EIDOUS KAI TIMIS STOIXEIOU

                        string value = "<" +
myds.Tables[i].Columns[k].ColumnName + ">" + myds.Tables[i].Rows[j][k] +
"</" + myds.Tables[i].Columns[k].ColumnName + ">";
                        fStream.WriteLine(value);
                    }
                    fStream.WriteLine("</record>");//KLEISIMO RECORD
                }
                fStream.WriteLine("</" + myds.Tables[i].TableName +
">");//KLEISIMO TABLE TAG
            }
            fStream.WriteLine("</" + catalogName + "DB" + ">"); //KLEISIMO
MEGALOU ARXIKOU TAG

        }//end method WriteXmlData
    }
}
```

5.4 Η Κλάση Presenter

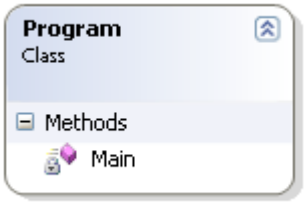



```
using System;
using System.Collections.Generic;
using System.Text;

namespace XMLization
{
    /// <summary>
    /// This class is used to present the database and the necessary
operations to the users screen.
    /// </summary>
    class Presenter
    {
        private string[] namesTable;
        private string[,] relationshipsTable;
        private System.Data.DataSet myds;
        private System.IO.StreamWriter fStream;
        private string catalogName;

        /// <summary>
        /// Class' constructor.
        /// </summary>
        /// <param name="namesTable">Matrix with the names of the tables
in the database.</param>
        /// <param name="relationshipsTable">Matrix with the relationships
in the database.</param>
        /// <param name="myds">The dataset representation of the
database.</param>
        /// <param name="fStream">Streamwriter object to print on user's
screen.</param>
        /// <param name="catalogName">Name of the catalog we are
presenting.</param>
        public Presenter(string[] namesTable, string[,]
relationshipsTable, System.Data.DataSet myds, System.IO.StreamWriter
fStream, string catalogName)
        {
            this.namesTable = namesTable;
            this.relationshipsTable = relationshipsTable;
```

```
            this.myds = myds;
            this.fStream = fStream;
            this.catalogName = catalogName;
        }

        /// <summary>
        /// Method which prints all the names of the tables that exists in
the database.
        /// </summary>
        public void GiveNamesTable()
        {
            //PAROUSIASI ONOMATON
            if (namesTable != null)
            {
                foreach (string it in namesTable)
                {
                    System.Console.WriteLine(it);
                }
            }

        }

        /// <summary>
        /// Method which prints all the names of the relationships that
exists in the database.
        /// </summary>
        public void GiveRelationshipsTable()
        {
            //PAROUSIASI RELATIONSHIPS
            if (relationshipsTable != null)
            {
                System.Console.WriteLine("\n");
                System.Console.WriteLine("Primary Table \t\t Primary
Column \t\t Foreign Table \t\t Foreing Column\n");

                for (int i = 0; i != relationshipsTable.GetLength(0); i++)
                {
                    Console.WriteLine("\n");

                    for (int j = 0; j != relationshipsTable.GetLength(1);
j++)
                    {
                        System.Console.Write("{0} \t\t",
relationshipsTable[i, j]);

                    }
                }
            }

        }

        /// <summary>
        /// Method which prints all the records that exists in the
database.
        /// </summary>
        public void GiveDataSet()
        {
            //PAROUSIASI DATASET
            for (int o = 0; o != myds.Tables.Count; o++)
            {
```

```
                Console.WriteLine("\n\n NEXT TABLE: {0} \n",
myds.Tables[o].TableName);

                for (int g = 0; g != myds.Tables[o].Columns.Count; g++)
                {
                    Console.Write("{0}\t",
myds.Tables[o].Columns[g].ColumnName);
                }

                Console.WriteLine();

                for (int i = 0; i != myds.Tables[o].Rows.Count; i++)
                {
                    Console.WriteLine("\nNEXT ROW \n ");
                    for (int j = 0; j != myds.Tables[o].Columns.Count;
j++)
                    {
                        Console.Write(String.Format("{0} ",
myds.Tables[o].Rows[i][j]));
                    }
                }
            }
        }

        /// <summary>
        /// Method which prints the XML file on the screen.
        /// </summary>
        public void GiveXmlOutput()
        {
            System.IO.StreamReader XmlReader = new
System.IO.StreamReader(catalogName+".xml");
            Console.WriteLine(XmlReader.ReadToEnd());
            XmlReader.Close();
        }

    }
}
```

5.5 Το Κυρίως Πρόγραμμα




```
using System;
using System.Collections.Generic;
using System.Text;

namespace XMLization
{
    class Program
    {

        /// <summary>
        /// The programm has a database input which is transformed to an
XML file of certain specifications.
        /// </summary>
        /// <param name="args"></param>
        ///

        static void Main(string[] args)
        {
            /*
            //ELEGXOS COMMAND-LINE ARGUMENTS
            if (args.Length != 3)
            {
                Console.WriteLine("Please enter all necessary
arguments.");
                Console.WriteLine("Usage: XMLization <server> <catalog>
<starting table>");
                Environment.Exit(-1);
            }

            // Convert the input arguments to strings:
            try
            {
                args[0] = Convert.ToString(args[0]);
                args[1] = Convert.ToString(args[1]);
                args[2] = Convert.ToString(args[2]);
            }
            catch (System.FormatException)
            {
                Console.WriteLine("Please enter all necessary
arguments.");
                Console.WriteLine("Usage: XMLization <server> <catalog>
<starting table>");
                Environment.Exit(-1);
            }

            */

            //ANOIGMA SINDESIS - ANAGNOSI DEDOMENON
```

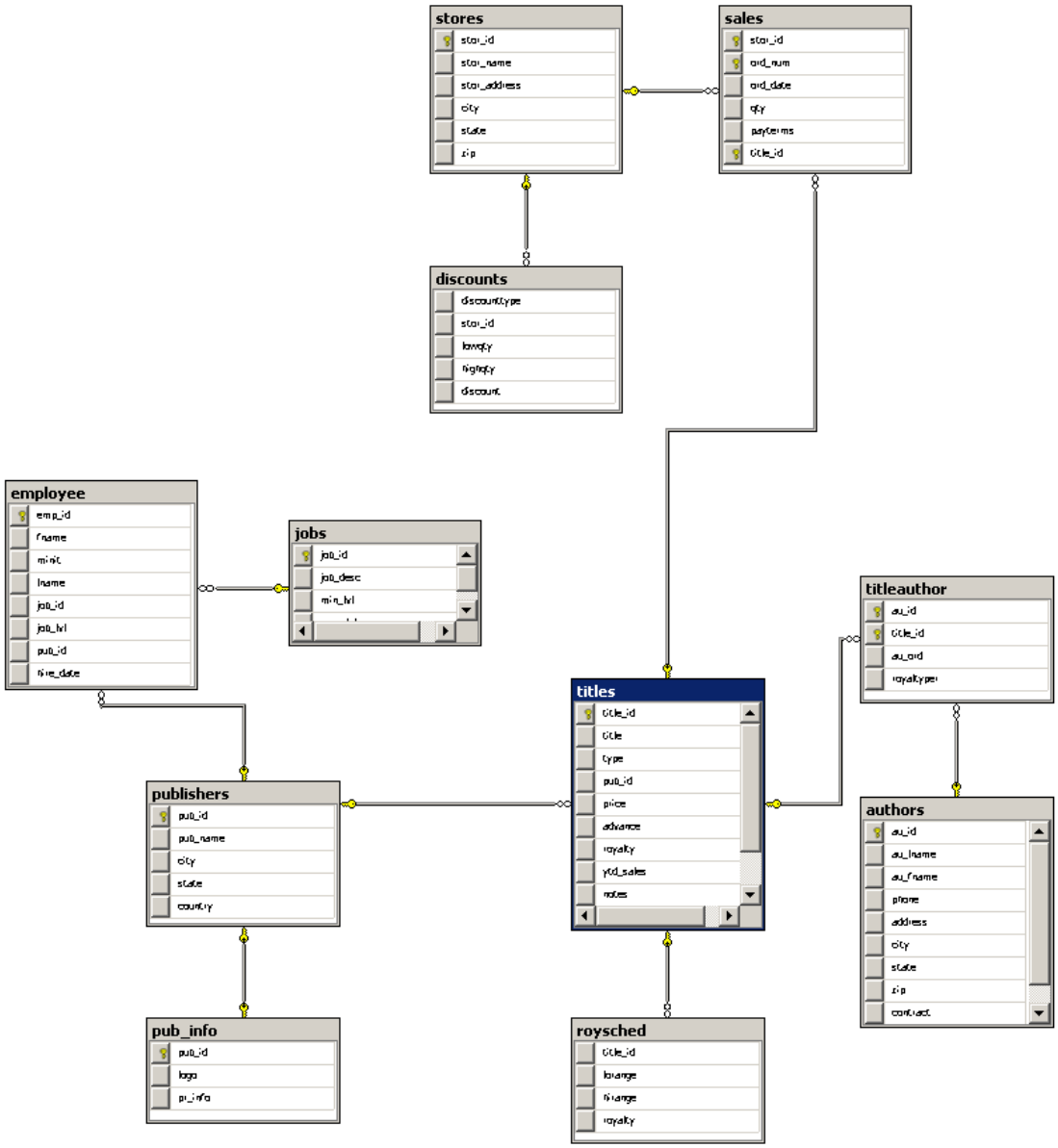
```
            //XRISI COMMAND-LINE ARGUMENTS
            DbConnector connection = new DbConnector("obelix",
"northwind");
            //DbConnector connection = new DbConnector(args[0], args[1]);


            DbReader reader = new DbReader();

            connection.conn =
connection.ConnectToSql(connection.ServerName,connection.CatalogName);

            reader.GetNamesTable(connection.conn);
            reader.GetRelationshipsTable(connection.conn);
            reader.CreateDataset(connection.conn, reader.NamesTable);
            connection.CloseSqlConnection(connection.conn);


            //EGGRAFI XML

            //XRISI COMMAND-LINE ARGUMENTS
            XmlCreator creator = new
XmlCreator(connection.CatalogName,"Territories");
            //XmlCreator creator = new XmlCreator(connection.CatalogName,
args[2]);

            creator.CreateXmlFile();
            creator.WriteXmlHeaders(creator.fStream,reader.myds);
            creator.WriteXmlRelationships(creator.fStream,
reader.RelationshipsTable);
            creator.WriteXmlData(creator.fStream, reader.myds);
            creator.CloseXmlFile(creator.fStream);

            //PAROUSIASI APOTELESMATON
            Presenter proceduresPresent = new
Presenter(reader.NamesTable,reader.RelationshipsTable,reader.myds,
creator.fStream, connection.CatalogName);

            proceduresPresent.GiveNamesTable();
            proceduresPresent.GiveRelationshipsTable();
            proceduresPresent.GiveDataSet();
            proceduresPresent.GiveXmlOutput();

        }
    }
}
```

6. Εφαρμογές

Προκειμένου να εξετάσουμε τη λειτουργικότητα του κώδικα χρησιμοποιήσαμε τις δυο βάσεις δεδομένων που παρέχει η Microsoft για δοκιμές. Πρόκειται για τις βάσεις δεδομένων Pubs και Northwind.

6.1 Η Βάση Δεδομένων Pubs

Η δοκιμαστική βάση δεδομένων pubs έχει στηριχθεί στα στοιχεία μιας εταιρίας έκδοσης βιβλίων.
Στο ακόλουθο σχήμα βλέπουμε ένα διάγραμμα της βάσης στο οποίο φαίνονται όλοι οι πίνακες και τα πεδία της βάσης, το η τα primary keys του κάθε πίνακα καθώς και σχέσεις μεταξύ των πεδίων των πινάκων.



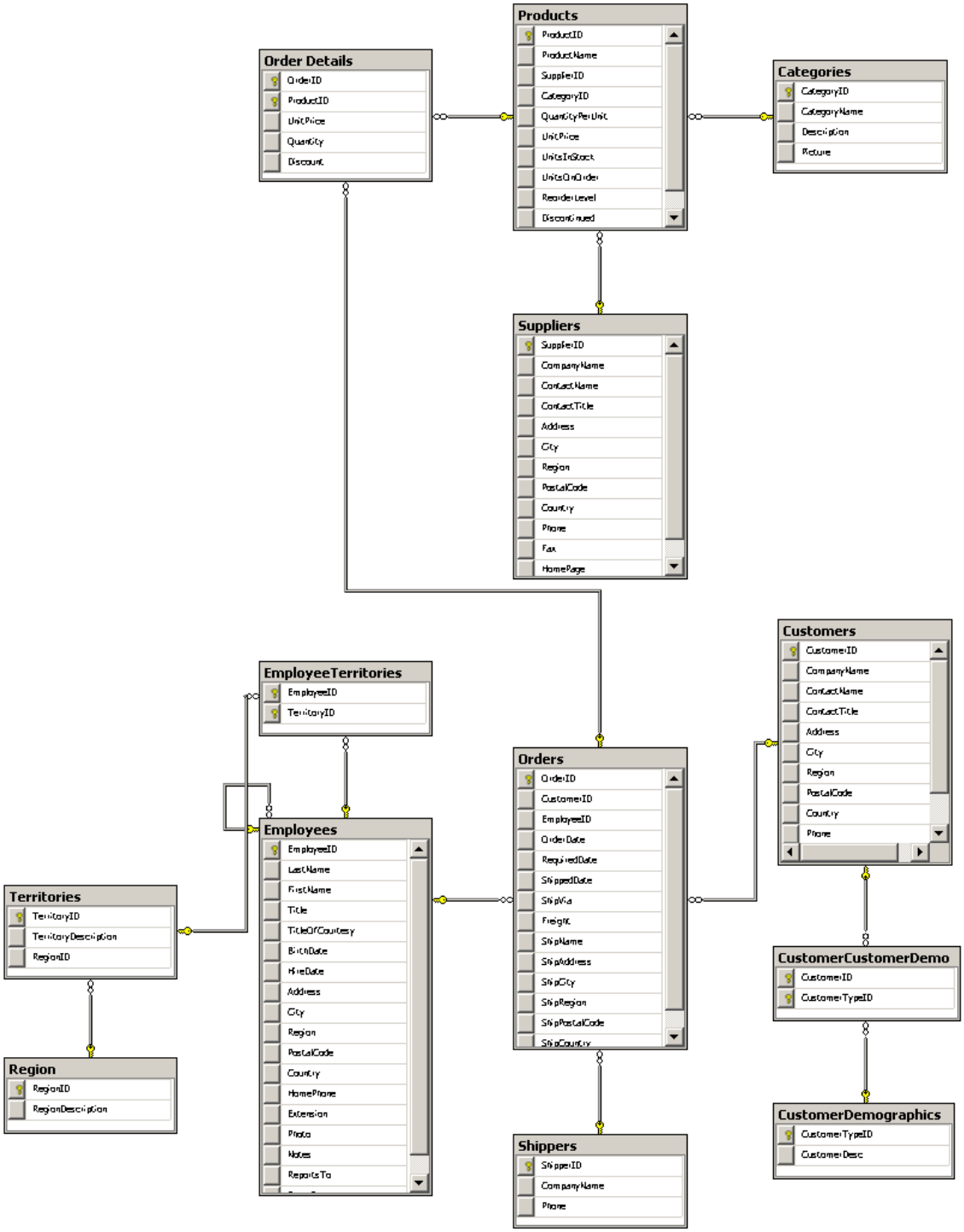
Στη συνέχεια παρατίθεται ένα μικρό μέρος του XML αρχείου που προκύπτει μετά από την εκτέλεση του προγράμματος χρησιμοποιώντας τη βάση δεδομένων Pubs και παίρνοντας ώς αρχικό πίνακα για τις σχέσεις τον πίνακα Sales:

```
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:cdb="http://www.cn.ntua.gr/cdb"
xmlns:pf="http://www.cn.ntua.gr/cdb_self"
xmlns:rel="http://www.cn.ntua.gr/relationships_self">
<sales xmlns="http://www.cn.ntua.gr/relationships_self">
<cdb:this_from thisF="stor_id" fromT="stores" fromF="stor_id" />
<stores>
</stores>
<cdb:this_from thisF="title_id" fromT="titles" fromF="title_id" />
<titles>
<cdb:this_from thisF="pub_id" fromT="publishers" fromF="pub_id" />
<publishers>
</publishers>
</titles>
</sales>
<pubsDB xmlns="http://www.cn.ntua.gr/cdb_self">
<authors xmlns="http://www.cn.ntua.gr/cdb_self">
<record>
<au_id>172-32-1176</au_id>
<au_lname>White</au_lname>
<au_fname>Johnson</au_fname>
<phone>408 496-7223</phone>
<address>10932 Bigge Rd.</address>
<city>Menlo Park</city>
<state>CA</state>
<zip>94025</zip>
<contract>True</contract>
</record>
<record>
<au_id>213-46-8915</au_id>
<au_lname>Green</au_lname>
<au_fname>Marjorie</au_fname>
<phone>415 986-7020</phone>
<address>309 63rd St. #411</address>
<city>Oakland</city>
<state>CA</state>
<zip>94618</zip>
<contract>True</contract>
</record>
<record>
<au_id>238-95-7766</au_id>
<au_lname>Carson</au_lname>
<au_fname>Cheryl</au_fname>
<phone>415 548-7723</phone>
<address>589 Darwin Ln.</address>
```

```
<city>Berkeley</city>
<state>CA</state>
<zip>94705</zip>
<contract>True</contract>
</record>
<record>
<au_id>267-41-2394</au_id>
<au_lname>O'Leary</au_lname>
<au_fname>Michael</au_fname>
<phone>408 286-2428</phone>
<address>22 Cleveland Av. #14</address>
<city>San Jose</city>
<state>CA</state>
<zip>95128</zip>
<contract>True</contract>
</record>
<record>
<au_id>274-80-9391</au_id>
<au_lname>Straight</au_lname>
<au_fname>Dean</au_fname>
<phone>415 834-2919</phone>
<address>5420 College Av.</address>
<city>Oakland</city>
<state>CA</state>
<zip>94609</zip>
<contract>True</contract>
</record>
<record>
<au_id>341-22-1782</au_id>
<au_lname>Smith</au_lname>
<au_fname>Meander</au_fname>
<phone>913 843-0462</phone>
<address>10 Mississippi Dr.</address>
<city>Lawrence</city>
<state>KS</state>
<zip>66044</zip>
<contract>False</contract>
</record>



...




</pubsDB>
</xsl:stylesheet>
```

6.2 Η Βάση Δεδομένων Northwind

Η δοκιμαστική βάση δεδομένων Northwind περιέχει τα δεδομένα πωλήσεων μια εικονικής εταιρίας που ονομάζεται Northwind Traders, η οποία εισάγει και εξάγει ειδικά τρόφιμα από ολόκληρο τον κόσμο.
Όμοια, παρουσιάζεται το αντίστοιχο διάγραμμα για τη βάση δεδομένων Northwind.



Παρακάτω φαίνεται ένα μικρό μέρος του XML αρχείου που παράγεται παίρνοντας τον πίνακα Territories σαν αρχικό πίνακα για τις σχέσεις:

```
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:cdb="http://www.cn.ntua.gr/cdb"
xmlns:pf="http://www.cn.ntua.gr/cdb_self"
xmlns:rel="http://www.cn.ntua.gr/relationships_self">
<Territories xmlns="http://www.cn.ntua.gr/relationships_self">
<cdb:this_from thisF="RegionID" fromT="Region" fromF="RegionID" />
<Region>
</Region>
</Territories>
<northwindDB xmlns="http://www.cn.ntua.gr/cdb_self">
<Categories xmlns="http://www.cn.ntua.gr/cdb_self">
<record>
<CategoryID>1</CategoryID>
<CategoryName>Beverages</CategoryName>
<Description>Soft drinks, coffees, teas, beers, and ales</Description>
<Picture>System.Byte[]</Picture>
</record>
<record>
<CategoryID>2</CategoryID>
<CategoryName>Condiments</CategoryName>
<Description>Sweet and savory sauces, relishes, spreads, and
seasonings</Description>
<Picture>System.Byte[]</Picture>
</record>
<record>
<CategoryID>3</CategoryID>
<CategoryName>Confections</CategoryName>
<Description>Desserts, candies, and sweet breads</Description>
<Picture>System.Byte[]</Picture>
</record>
<record>
<CategoryID>4</CategoryID>
<CategoryName>Dairy Products</CategoryName>
<Description>Cheeses</Description>
<Picture>System.Byte[]</Picture>
</record>
<record>
<CategoryID>5</CategoryID>
<CategoryName>Grains/Cereals</CategoryName>
<Description>Breads, crackers, pasta, and cereal</Description>
<Picture>System.Byte[]</Picture>
</record>
<record>
<CategoryID>6</CategoryID>
<CategoryName>Meat/Poultry</CategoryName>
<Description>Prepared meats</Description>
<Picture>System.Byte[]</Picture>
</record>
```

```
<record>
<CategoryID>7</CategoryID>
<CategoryName>Produce</CategoryName>
<Description>Dried fruit and bean curd</Description>
<Picture>System.Byte[]</Picture>
</record>
<record>
<CategoryID>8</CategoryID>
<CategoryName>Seafood</CategoryName>
<Description>Seaweed and fish</Description>
<Picture>System.Byte[]</Picture>
</record>
</Categories>
<CustomerCustomerDemo xmlns="http://www.cn.ntua.gr/cdb_self">
</CustomerCustomerDemo>
<CustomerDemographics xmlns="http://www.cn.ntua.gr/cdb_self">
</CustomerDemographics>
<Customers xmlns="http://www.cn.ntua.gr/cdb_self">
<record>
<CustomerID>ALFKI</CustomerID>
<CompanyName>Alfreds Futterkiste</CompanyName>
<ContactName>Maria Anders</ContactName>
<ContactTitle>Sales Representative</ContactTitle>
<Address>Obere Str. 57</Address>
<City>Berlin</City>
<Region></Region>
<PostalCode>12209</PostalCode>
<Country>Germany</Country>
<Phone>030-0074321</Phone>
<Fax>030-0076545</Fax>
</record>
<record>
<CustomerID>ANATR</CustomerID>
<CompanyName>Ana Trujillo Emparedados y helados</CompanyName>
<ContactName>Ana Trujillo</ContactName>
<ContactTitle>Owner</ContactTitle>
<Address>Avda. de la Constitucion 2222</Address>
<City>Mexico D.F.</City>
<Region></Region>
<PostalCode>05021</PostalCode>
<Country>Mexico</Country>
<Phone>(5) 555-4729</Phone>
<Fax>(5) 555-3745</Fax>
</record>
<record>
<CustomerID>ANTON</CustomerID>
<CompanyName>Antonio Moreno Taqueria</CompanyName>
<ContactName>Antonio Moreno</ContactName>
<ContactTitle>Owner</ContactTitle>
<Address>Mataderos  2312</Address>
```

```
<City>Mexico D.F.</City>
<Region></Region>
<PostalCode>05023</PostalCode>
<Country>Mexico</Country>
<Phone>(5) 555-3932</Phone>
<Fax></Fax>
</record>


...



</northwindDB>
</xsl:stylesheet>
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] C# : a programmer's introduction / H. M. Deitel

[2] C# for programmers / Harvey M. Deitel, Paul J. Deitel

[3] Microsoft Visual C# .NET : step by step / John Sharp, Jon Jagger

[4] Sams teach yourself XML in 24 hours / Charles Ashbacher

[5] Sams teach yourself Microsoft SQL server 2000 in 21 days / Richard Waymire, Rick Sawtell

[6] MySQL : your visual blueprint to open source database management / by Michael Moncur

[7] Πλήρες εγχειρίδιο της Visual Basic.NET / Evangelos Petroutsos ; απόδοση Γ

[8] Introduction to Microsoft Access 2003 [Online]. Available: http://www.ischool.utexas.edu/technology/tutorials/office/access03/

[9] AulaClic. Access2003 Tutorial [Online]. Available: http://www.teacherclick.com/access2003/index.htm

[10] Microsoft Access Tutorial [Online]. Available: http://www.bcschools.net/staff/AccessHelp.htm

[11] Microsoft Developer Network [Online]. Available: http://msdn.microsoft.com/en-us/library/default.aspx

[12] Database – Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/Database

[13] XML – Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/XML

[14] Microsoft .NET Framework [Online]. Available: http://www.microsoft.com/NET/

[15] .NET Framework – Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/.NET_Framework

[16] Building Sandcastle Documentation [Online]. Available: http://ozgrant.com/2008/02/19/building-sandcastle-documentation-with-team-build/