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

# ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

# Υλοποίηση Εργαλείου μετατροπής μιας Αντικειμενοστρεφής Βάσης Δεδομένων σε αντίστοιχη XML μορφής.

**Του φοιτητή**

**Μιχαήλ Ζουναρόπουλου**

**Αρ. Μητρώου: 05/2821**

**Επιβλέπων καθηγητής**

**Ευκλείδη Κεραμόπουλος**

**Θεσσαλονίκη 2012**

## ΠΡΟΛΟΓΟΣ

Σκοπός αυτής της πτυχιακής εργασίας είναι να αναδείξει τα ζητήματα που υπάρχουν σχετικά με τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων και τον τρόπο εξαγωγής των δεδομένων του σε XML έγγραφα καθώς και να τα θέσει υπό συζήτηση. Θα περιγραφούν οι βασικές έννοιες των αντικειμενοστρεφών τεχνολογιών και το τι χρειάζεται ένα σύστημα για να χαρακτηριστεί ως αντικειμενοστρεφές σύστημα διαχείρισης βάσεων δεδομένων, θα μελετήσουμε παρόμοιες τεχνολογίες, θα τις αναλύσουμε και θα τις συγκρίνουμε, θα επιλέξουμε μία τεχνολογία για να την αναπτύξουμε και θα εξάγουμε τα δεδομένα του σε XML. Ο λόγος για τον οποίο τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων θέτονται υπό συζήτηση είναι οι εξαιρετικές τους αποδόσεις και το μικρό μέγεθος που έχουν στη μνήμη όταν τρέχουν. Χρησιμοποιώντας γλώσσες προγραμματισμού όπως, Java, C# πετυχαίνουμε μεγάλο βαθμό μεταφερσιμότητας, καθώς παρατηρούμε πως η ανάγκη για γρήγορες και αποδοτικές βάσεις δεδομένων αυξάνετε, αφού πλέον στην αγορά έχουνε μπει δυνατά συσκευές όπως tablet και mobile phones. Έτσι, σημαντικό ήταν να βρούμε μία πλατφόρμα η οποία να μας παρέχει την ικανότητα να προγραμματίσουμε και για τέτοιες συσκευές. Το XML είναι πλέον ένας πολύ σημαντικός τρόπος μεταφοράς δεδομένων μέσω του διαδικτύου. Έτσι, γίνετε σαφές πως τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων θα πρέπει να υποστηρίζουνε την εισαγωγή από XML καθώς και την εξαγωγή των δεδομένων τους σε XML. Επίσης, θα ήταν συνετό να δημιουργηθεί και τρόπος εξαγωγής μεταδεδομένων από τις βάσεις δεδομένων, έτσι ώστε να μπορούμε να επικοινωνούμε με κατανεμημένες βάσεις δεδομένων.

## ΠΕΡΙΛΗΨΗ

Σε αυτή την εργασία ξεκινάμε περιγράφοντας βασικές έννοιες σχετικά με τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων όπως ορίζονται από το ODMG 3.0. Έπειτα, θα αναφέρουμε τα χαρακτηριστικά που χρειάζονται να υλοποιούνται από το σύστημα για να μπορεί να θεωρηθεί ως αντικειμενοστρεφής, τα χαρακτηριστικά που μπορούν να αυξήσουνε την απόδοση του συστήματος αλλά είναι προαιρετικά, καθώς και επιλογές που μπορεί να κάνει ο προγραμματιστής της βάσης δεδομένων για να αυξήσει την απόδοση και την λειτουργικότητα του συστήματος. Μετά, θα μελετήσουμε τις τεχνολογίες αντικειμενοστρεφών βάσεων δεδομένων που υπάρχουνε δωρεάν στην αγορά, θα τις μελετήσουμε, θα τις περιγράψουμε και έπειτα θα τις συγκρίνουμε έτσι ώστε να μπορέσουμε να αποφασίσουμε ποια εφαρμογή θα χρησιμοποιήσουμε και επίσης, να μελετήσουμε τι δεν είναι υλοποιημένο σε αυτές τις εφαρμογές και θα μπορούσε να γίνει για να αυξήσει την απόδοση και τις ανάγκες του προγραμματιστή καθώς και τις ανάγκες τις αγοράς. Αφού αποφασίσουμε ποια εφαρμογή θα χρησιμοποιήσουμε θα αναπτύξουμε τα χαρακτηριστικά της έτσι ώστε να έχουμε πιο σαφή εικόνα, καθώς και τη διαδικασία που πρέπει να ακολουθήσουμε για να εξάγουμε τα μεταδεδομένα της βάσης μας. Έπειτα, θα δείξουμε τον τρόπο λειτουργίας της εφαρμογής και θα εξάγουμε τα δεδομένα του σε XML έγγραφο, όπως επίσης και τα μεταδεδομένα τους, π.χ. ονόματα κλάσης, μεταβλητών, μεθόδων και αντίστοιχα των τύπων τους. Στο τέλος θα παρατεθούν συμπεράσματα και θα τεθούν θέματα για συζήτηση για το πώς θα μπορούσαμε να βελτιώσουμε τη τεχνολογία αντικειμενοστρεφών συστημάτων διαχείρισης βάσεων δεδομένων, καθώς και του τρόπου εξαγωγής των δεδομένων σε XML.

## ABSTRACT

In this paper we will start presenting the basic concepts about object – oriented database management systems as defined by the ODMG 3.0 standard. Next, we will mention the characteristics that an object – oriented database should support so we could name it that way, some optional features that could improve the performance of the system and some open features that can increase the programmer's performance and system too. Later, we will study the object – oriented database technologies which we can find as a freeware on the market, we will describe them and study them and then we are going to compare them, so we could decide which application we will use to create our own database and furthermore we will mention what it's not implemented to those systems and could been done for better performance and the needs of the programmer and the market. After we decide what application to use, we will describe the characteristics of the certain application so we could have a cleaner view of it and the procedure we must follow to export metadata from the database. Later on, we will see the way the application work and then export the data to a XML file and the metadata, i.e. class names, properties, methods and the types of those. In the end we will set conclusions and topics for discussion for how we could improve object – oriented database management systems, and the way we export the data to XML file.

## ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον καθηγητή μου κύριο Ευκλείδη Κεραμόπουλο που μου έδωσε την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα καθώς και για το ενδιαφέρον που έδειξε στη διάρκεια της έρευνας. Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου, που μου συμπαραστέκεται σε ότι και αν κάνω στη ζωή μου.

# Περιεχόμενα

## ΚΕΦΑΛΑΙΟ 1

# Αντικειμενοστρεφείς Συστήματα Διαχείρισης Βάσεων Δεδομένων (OODBMS)

### Εισαγωγή

Σε αυτό το κεφάλαιο θα προσπαθήσουμε να ορίσουμε ένα αντικειμενοστρεφές σύστημα διαχείρισης βάσεων δεδομένων. Θα περιγράψουμε τα κύρια χαρακτηριστικά που πρέπει να έχει ένα σύστημα για να χαρακτηριστεί αντικειμενοστρεφής σύστημα διαχείρισης βάση δεδομένων.

Έχουμε ξεχωρίσει αυτά τα χαρακτηριστικά σε τρείς κατηγορίες:

- Υποχρεωτικά (Mandatory), αυτά που ένα σύστημα βάσεων δεδομένων πρέπει να ικανοποιεί προκειμένου να ονομαστεί ως αντικειμενοστρεφής. Αυτά είναι τα σύνθετα αντικείμενα (complex objects), το αναγνωριστικό του αντικειμένου (object identity), ενθυλάκωση (encapsulation), τύποι και κλάσεις (types and classes), κληρονομικότητα (inheritance), υπερφόρτωση μαζί με αργή σύνδεση (overriding and late binding), επεκτασιμότητα (extensibility), υπολογιστική πληρότητα (computational completeness), διατήρηση (persistence), διαχείριση δευτερεύουσας μνήμης (secondary storage management), ταυτοχρονισμός (concurrency), επαναφορά (recovery) και ad hoc μηχανισμό δημιουργίας ερωτημάτων (query facility).
- Προαιρετικά (Optional), αυτά που μπορούν να χρησιμοποιηθούν για να κάνουν το σύστημα μας καλύτερο, αλλά δεν είναι υποχρεωτικά. Αυτά είναι η πολλαπλή κληρονομικότητα (multiple inheritance), έλεγχος τύπων και τεκμηρίωση (type checking and inferencing), καταμερισμό (distribution), σχεδιασμός συναλλαγών (design transaction) και εκδόσεις (versions).
- Ανοιχτές (Open), τα σημεία όπου ο σχεδιαστής μπορεί να κάνει μια σειρά από επιλογές. Αυτά είναι το παράδειγμα προγραμματισμού, το σύστημα αναπαράστασης, το σύστημα τύπων και της ομοιομορφίας.

## 1.1 Ορισμός αντικειμενοστρεφών βάσεων δεδομένων

Το σύστημα διαχείρισης βάσεων δεδομένων αντικειμένων (ODBMS, που αναφέρεται επίσης ως αντικειμενοστρεφές σύστημα διαχείρισης βάσεων δεδομένων ή OODBMS), είναι ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS) που υποστηρίζει την μοντελοποίηση και τη δημιουργία των δεδομένων ως

αντικείμενα. Αυτό περιλαμβάνει κάποιο είδος υποστήριξης για κλάσεις αντικειμένων και την κληρονομικότητα των ιδιοτήτων και των μεθόδων από τις υποκλάσεις και τα αντικείμενά τους.

Στο έγγραφό τους, The Object - Oriented Database System Manifesto [1], ο Malcolm Atkinson και άλλοι ορίζουν ένα OODBMS, το 1995, ως εξής : Ένα αντικειμενοστρεφές σύστημα βάσεων δεδομένων πρέπει να πληροί δύο κριτήρια : πρέπει να είναι ένα σύστημα διαχείρισης βάσεων δεδομένων και θα πρέπει να είναι ένα αντικειμενοστρεφές σύστημα, δηλαδή, στο μέτρο του δυνατού, πρέπει να είναι συνεπής (consistent) με τις τρέχουσες εκδόσεις των αντικειμενοστρεφών γλωσσών προγραμματισμού. Το πρώτο κριτήριο μεταφράζεται σε πέντε χαρακτηριστικά : persistence, secondary storage management, concurrency, recovery και ad hoc query facility. Το δεύτερο μεταφράζεται σε οκτώ χαρακτηριστικά : complex objects, object identity, encapsulation, types or classes, inheritance, overriding combined with late binding, extensibility και computational completeness. Το έγγραφό τους περιγράφει κάθε ένα από αυτά τα χαρακτηριστικά με λεπτομέρεια.

Το ΑΣΔΒΔ αρχικά θεωρήθηκε ότι θα αντικαθιστούσε το ΣΣΔΒΔ επειδή ταιριάζουνε καλύτερα με τις αντικειμενοστρεφές γλώσσες προγραμματισμού. Ωστόσο, λόγο του υψηλού κόστους μεταστροφής, η ένταξη των αντικειμενοστρεφών χαρακτηριστικών σε ένα ΣΣΔΒΔ για την μετατροπή τους σε αντικειμενοσχεσιακά ΣΔΒΔ, και η εμφάνιση αντικειμενοσχεσιακών mappers (ORMs) έχουν κάνει τα ΣΣΔΒΔ να υπερασπίζονται επιτυχώς την κυριαρχία τους στα κέντρα δεδομένων για server – side persistence. Οι αντικειμενοστρεφής βάσεις δεδομένων έχουν πλέον καθιερωθεί ως ένα συμπλήρωμα και όχι μια αντικατάσταση για σχεσιακές βάσεις δεδομένων. Βρήκαν τη θέση τους ως ενσωματωμένες λύσεις σε συσκευές, σε clients, σε πακέτα λογισμικού, σε συστήματα ελέγχου πραγματικού χρόνου καθώς και σε δυναμικές ιστοσελίδες.

## 1.2 Υποχρεωτικά χαρακτηριστικά – Mandatory features

### 1.2.1 Εισαγωγή

Ένα αντικειμενοστρεφής σύστημα διαχείρισης βάσεων δεδομένων πρέπει να πληροί δυο κριτήρια : πρέπει να είναι ένα σύστημα διαχείρισης βάσεων δεδομένων και πρέπει να είναι ένα αντικειμενοστρεφής σύστημα.

### 1.2.1 Σύνθετα αντικείμενα – Complex objects

Τα σύνθετα – πολύπλοκα αντικείμενα είναι φτιαγμένα από απλούστερα αντικείμενα εφαρμόζοντας κατασκευαστές σε αυτά. Τα πιο απλά αντικείμενα είναι τύπου ακέραιων, χαρακτήρων, συμβολοσειρές οποιουδήποτε μήκους, κινητής υποδιαστολής και λογικές. Υπάρχει μεγάλη ποικιλία από κατασκευαστές σύνθετων

αντικειμένων όπως : tuples, sets, bags, lists και πίνακες, είναι ένα παράδειγμα. Το ελάχιστο σύνολο των κατασκευαστών που πρέπει να έχει το σύστημα είναι set, tuple και list. Τα set είναι κρίσιμα επειδή είναι ένας φυσικός τρόπος να αναπαραστήσεις συλλογές από τον πραγματικό κόσμο. Τα tuple είναι κρίσιμα επειδή είναι ένας φυσικός τρόπος της αναπαράστασης ιδιοτήτων μιας οντότητας. Φυσικά και τα δύο sets, tuples είναι σημαντικά επειδή έχουν κερδίσει ευρεία αποδοχή ως κατασκευαστές αντικειμένων στο σχεσιακό μοντέλο. Οι λίστες ή οι πίνακες είναι σημαντικοί διότι συλλαμβάνουν την σειρά, το οποίο συμβαίνει στον πραγματικό κόσμο και επίσης προκύπτουν σε πολλές επιστημονικές εφαρμογές, όπου άνθρωποι χρειάζονται μήτρες ή χρονοσειρές δεδομένων. Οι κατασκευαστές αντικειμένων πρέπει να είναι ανεξάρτητοι και να μην επηρεάζουν τα άλλα χαρακτηριστικά : οποιοσδήποτε κατασκευαστής θα μπορεί να εφαρμοστεί σε οποιοδήποτε αντικείμενο. Οι κατασκευαστές στο σχεσιακό μοντέλο δεν είναι ανεξάρτητοι, επειδή ο κατασκευαστής του set μπορεί να εφαρμοστεί μόνο σε tuples και οι κατασκευαστές των tuples μπορούν να εφαρμοστούν μόνο σε ατομικές τιμές. Άλλα παραδείγματα είναι η μη – πρώτη κανονική μορφή του σχεσιακού μοντέλου στο οποίο ο αρχικός κατασκευαστής πρέπει να είναι πάντοτε μια σχέση.

### 1.2.2 Ταυτότητα αντικειμένου – Object identity

Η ταυτότητα του αντικειμένου υπάρχει πολύ καιρό στις γλώσσες προγραμματισμού. Η ιδέα είναι ποιο πρόσφατη στις βάσεις δεδομένων, π.χ. (Hall et al. 76) [2], (Maier and Price 84) [3], (Khoshafian and Copeland 86) [4]. Η ιδέα είναι η ακόλουθη : σε ένα μοντέλο με ταυτότητα αντικειμένου, το αντικείμενο έχει μια υπόσταση που είναι ανεξάρτητη από την τιμή του. Έτσι υπάρχουν δύο έννοιες της ισοτιμίας αντικειμένων : δύο αντικείμενα μπορούν να είναι πανομοιότυπα (είναι τα ίδια αντικείμενα) ή μπορούν να είναι ίσα (έχουν την ίδια τιμή). Αυτό έχει δύο συνέπειες : ένα είναι η κοινή χρήση του αντικειμένου και το άλλο η ενημέρωσή του.

Κοινή χρήση αντικειμένων : σε ένα μοντέλο με βάση την ταυτότητα, δύο αντικείμενα μπορούν να μοιράζονται ένα συστατικό. Έτσι, η γραφική αναπαράσταση ενός σύνθετου αντικειμένου είναι ένας γράφος, ενώ περιορίζεται να είναι ένα δέντρο σε ένα σύστημα χωρίς ταυτότητες αντικειμένων. Ας σκεφτούμε το ακόλουθο παράδειγμα : Ένα άτομο έχει όνομα, μία ηλικία και ένα σύνολο από παιδιά. Υποθέστε πώς ο Peter και η Susan έχουν και οι δύο ένα παιδί με όνομα John ηλικίας 15 ετών. Στην κανονική ζωή δύο καταστάσεις μπορούν να προκύψουν : Η Susan και ο Peter είναι γονείς του ίδιου παιδιού ή υπάρχουν δύο παιδιά που συμμετέχουν. Σε ένα σύστημα χωρίς ταυτότητα, ο Peter αναπαριστάτε έτσι :

(peter, 40, {(john, 15, {})})

Και η Susan αναπαριστάτε έτσι :

(susan, 41, {john, 15, {})}).

Έτσι δεν υπάρχει τρόπος να εκφραστεί αν ο Peter και η Susan είναι τελικά οι γονείς του ίδιου παιδιού. Σε ένα μοντέλο με βάση την ταυτότητα, αυτές η δύο δομές μπορούν να μοιραστούν το κοινό κομμάτι (john, 15, {}) ή όχι, έτσι, καταγράφονται δύο καταστάσεις.

Ενημέρωση αντικειμένου : υποθέτουμε ότι ο Peter και η Susan είναι πράγματι γονείς ενός παιδιού ονόματι John. Σε αυτή την περίπτωση, όλες οι ενημερώσεις στο παιδί της Susan θα εφαρμόζονται στο αντικείμενο John και κατά συνέπεια, επίσης στο παιδί του Peter. Σε ένα σύστημα value – based και τα δύο αντικείμενα πρέπει να ενημερωθούν ξεχωριστά. Η ταυτότητα του αντικειμένου είναι ένα θεμελιώδη στοιχείο διαχείρισης δεδομένων, που μπορεί να είναι η βάση διαχείρισης συνόλων, πλειάδων και αναδρομικών σύνθετων αντικειμένων (Abiteboul and Kanellakis 89) [5].

Το να υποστηρίζει ταυτότητα υποδηλώνει ότι θα παρέχονται λειτουργίες όπως ανάθεση αντικειμένου, αντιγραφή αντικειμένου (deep and shallow copy) και δοκιμές για την ταυτότητα και την ισότητα των αντικειμένων (deep and shallow equality).

Φυσικά, κάποιος μπορεί να προσομοιώσει τη ταυτότητα του αντικειμένου σε ένα value – based σύστημα, εισάγοντας σαφή προσδιορισμό αντικειμένων. Ωστόσο, η προσέγγιση αυτή επιβαρύνει τον χρήστη στο να διασφαλίσει τη μοναδικότητα των αναγνωριστικών αντικειμένων και να διατηρήσει την αναφορική ακεραιότητα των δεδομένων (αυτό το φορτίο μπορεί να είναι σημαντικό για λειτουργίες όπως ο garbage collector).

### 1.2.3 Ενθυλάκωση – Encapsulation

Η ιδέα της ενθυλάκωσης προέρχεται από (1) την ανάγκη για μια καθαρή διάκριση μεταξύ των προδιαγραφών και της υλοποίησης μιας λειτουργίας και (2) την ανάγκη για διαμερισμό. Ο διαμερισμός αυτός, είναι αναγκαίος για να κατασκευάσουμε σύνθετες εφαρμογές σχεδιασμένες και υλοποιημένες από μία ομάδα προγραμματιστών. Είναι επίσης χρήσιμο σαν εργαλείο για προστασία και αδειοδότηση. Υπάρχουν δύο απόψεις σχετικά με την ενθυλάκωση : η άποψη από την μεριά των γλωσσών προγραμματισμού (που είναι και η πρώτη άποψη αφού η ιδέα προέρχεται από εκεί) και η αναπροσαρμογή των βάσεων δεδομένων σε αυτή.

Η ιδέα της ενθυλάκωσης στις γλώσσες προγραμματισμού προέρχεται από τους αφηρημένους τύπους δεδομένων. Κατά την άποψη αυτή, ένα αντικείμενο έχει μία

διασύνδεση και μία υλοποίηση. Το κομμάτι της διασύνδεσης είναι ο προσδιορισμός του συνόλου των λειτουργιών που μπορούν να εφαρμοστούν σε ένα αντικείμενο. Είναι το μόνο εμφανές μέρος ενός αντικειμένου. Το τμήμα της υλοποίησης έχει ένα μέρος δεδομένων και ένα διαδικασιών. Το μέρος των δεδομένων είναι η αναπαράσταση της κατάστασης του αντικειμένου και η διαδικασία περιγράφει, σε κάποιες γλώσσες προγραμματισμού, την υλοποίηση αυτών των λειτουργιών. Η αρχή της ενθυλάκωσης ενός αντικειμένου στη βάση δεδομένων είναι πως ενθυλακώνει δεδομένα και πρόγραμμα. Στον κόσμο των βάσεων δεδομένων, δεν είναι σαφές πότε η δομή ενός τύπου είναι ή όχι μέρος της διασύνδεσης (αυτό εξαρτάται από το σύστημα), ενώ στις γλώσσες προγραμματισμού η δομή των δεδομένων είναι ξεκάθαρα μέρος της υλοποίησης και όχι της διασύνδεσης.

Ας σκεφτούμε, για παράδειγμα, έναν υπάλληλο. Σε ένα σχεσιακό σύστημα, ένας υπάλληλος αναπαρίσταται από μια πλειάδα χαρακτηριστικών. Γίνονται ερωτήματα πάνω του χρησιμοποιώντας μια σχεσιακή γλώσσα (SQL) και ίσως ένας προγραμματιστής να γράψει ένα πρόγραμμα που θα ενημερώνει αυτή την εγγραφή, όπως να αυξήσει τον μισθό ή να απολύσει τον εργαζόμενο. Αυτές είναι γενικά γραμμένες είτε με επιτακτικές γλώσσες προγραμματισμού με ενσωματωμένα DML γεγονότα (Data Manipulation Language), ή με γλώσσες προγραμματισμού τέταρτης γενιάς και αποθηκεύονται σε ένα κλασικό σύστημα αρχείων και όχι στην βάση δεδομένων. Έτσι με αυτή την προσέγγιση, υπάρχει ένας σαφής διαχωρισμός μεταξύ προγράμματος και δεδομένων, και μεταξύ των γλωσσών ερωτημάτων και των γλωσσών προγραμματισμού. Σε ένα αντικειμενοστρεφές σύστημα, ορίζουμε έναν υπάλληλο σαν αντικείμενο που έχει ένα κομμάτι δεδομένων (πιθανώς παρόμοιο με το κομμάτι που είχε γραφτεί για το σχεσιακό σύστημα) και ένα κομμάτι με λειτουργίες όπως η αύξηση μισθού ή η απόλυση, καθώς και άλλες για να έχουμε πρόσβαση στα δεδομένα του υπαλλήλου. Όταν αποθηκεύουμε έναν υπάλληλο στην βάση τότε αποθηκεύονται μαζί του τα δεδομένα και οι λειτουργίες του. Έτσι δημιουργείται μόνο ένα μοντέλο δεδομένων και λειτουργιών και οι πληροφορίες παραμένουν κρυφές. Καμία λειτουργία έξω από ότι ορίζεται στην διασύνδεση δεν μπορεί να λειτουργήσει. Αυτός ο περιορισμός ισχύει και για την λειτουργία ενημέρωσης και της ανάκτησης.

Η ενθυλάκωση παρέχει μία μορφή λογικής ανεξαρτησίας των δεδομένων : μπορούμε να αλλάξουμε την υλοποίηση ενός τύπου χωρίς να αλλάξουμε κανένα πρόγραμμα που χρησιμοποιεί αυτόν τον τύπο. Έτσι, τα προγράμματα της εφαρμογής είναι προστατευμένα από αλλαγές της υλοποίησης στα κατώτερα στρώματα του συστήματος. Τελικά, έχουμε ενθυλάκωση όταν έχουμε καταφέρει να μπορούμε να δούμε τις λειτουργίες, αλλά όχι τα δεδομένα καθώς και την υλοποίηση των λειτουργιών που είναι κρυμμένα μέσα στο αντικείμενο. Από την άλλη όμως, υπάρχουνε φορές που η ενθυλάκωση δεν χρειάζεται και η χρήση του συστήματος μπορεί να απλοποιηθεί σημαντικά εάν το σύστημα αφήσει να παραβιαστεί η ενθυλάκωση κάτω από συγκεκριμένες συνθήκες. Για παράδειγμα, στα ad hoc queries η ανάγκη για ενθυλάκωση μειώνετε καθώς θέματα όπως η

συντηρησιμότητα δεν είναι σημαντικά. Όμως στα αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων ο μηχανισμός ενθυλάκωσης πρέπει να παρέχεται, αλλά υπάρχουν περιπτώσεις που η επιβολή του δεν είναι αναγκαία.

### 1.2.4 Τύποι και Κλάσεις – Types and Classes

Υπάρχουν δύο κύριες κατηγορίες αντικειμενοστρεφών συστημάτων, αυτά που υποστηρίζουνε την έννοια της κλάσης και αυτά που υποστηρίζουνε την έννοια των τύπων. Στην πρώτη κατηγορία, ανήκουν συστήματα όπως Smalltalk [6], Gemstone [7], Vision [8] και πιο γενικά, όλα τα συστήματα που ανήκουν στην οικογένεια της Smalltalk, Orion [9], Flavors [10], G-Base [11], Lore [12] και γενικά όλα τα συστήματα που προήρθαν από την Lisp. Στην δεύτερη κατηγορία βρίσκουμε συστήματα όπως C++ [13], Simula [14], Trellis/Owl [15], Vbase [16] και η o2 [17].

Ο τύπος, στα αντικειμενοστρεφή συστήματα, συνοψίζει τα κοινά στοιχεία από ένα σύνολο αντικειμένων με κοινά χαρακτηριστικά. Ανταποκρίνεται στην ιδέα ενός αφηρημένου τύπου δεδομένων. Έχει δύο μέρη : την διασύνδεση και την υλοποίηση. Μόνο η διασύνδεση είναι ορατή στους χρήστες του τύπου, η υλοποίηση του αντικειμένου είναι εμφανής μόνο από τον σχεδιαστή του τύπου αυτού. Η διασύνδεση αποτελείτε από μία λίστα λειτουργιών μαζί με τις υπογραφές τους. Η υλοποίηση του τύπου αποτελείτε από τα δεδομένα και τις λειτουργίες. Στα δεδομένα, ο καθένας περιγράφει την εσωτερική δομή των δεδομένων του αντικειμένου. Η πολυπλοκότητα της δομής των δεδομένων εξαρτάται από τις ικανότητες του συστήματος. Οι λειτουργίες αποτελούνται από διεργασίες που υλοποιούν τις λειτουργίες της διασύνδεσης. Στις γλώσσες προγραμματισμού, οι τύποι είναι εργαλεία για να αυξήσουν την παραγωγικότητα του προγραμματιστή, διασφαλίζοντας την ορθότητα του προγράμματος. Αναγκάζοντας τον χρήστη να δημιουργήσει τους τύπους των μεταβλητών και τις εκφράσεις που θα διαχειριστεί, το σύστημα μας ενημερώνει σχετικά με την ορθότητα των προγραμμάτων που χτίζονται με αυτούς τους τύπους. Εάν, το σύστημα των τύπων έχει σχεδιαστεί προσεκτικά, το σύστημα μπορεί να κάνει τον έλεγχο των τύπων στην μεταγλώττιση, αλλιώς θα πρέπει να το αναβάλει. Έτσι, οι τύποι χρησιμοποιούνται κυρίως στην μεταγλώττιση για να ελεγχθεί η ορθότητα των προγραμμάτων.

Η ιδέα της κλάσης είναι διαφορετική από ότι στους τύπους. Οι προδιαγραφές του είναι παρόμοιες με αυτές των τύπων, αλλά αντίθετα με τους τύπους, αυτή στρέφεται προς την εκτέλεση του προγράμματος. Περιέχει δύο πτυχές : Ένα κατασκευαστή αντικειμένων και μία αποθήκη αντικειμένων. Το εργοστάσιο αντικειμένων χρησιμοποιείται για να δημιουργήσει νέα αντικείμενα εφαρμόζοντας τον τελεστή new στην κλάση ή κλωνοποιώντας κάποιο πρωτότυπο που αντιπροσωπεύει μια κλάση. Οι αποθήκες αντικειμένων σημαίνουν ότι μαζί με την κλάση επισυνάπτεται και η κατάληξή του, π.χ., το σύνολο των αντικειμένων όπου είναι στιγμιότυπα της κλάσης. Ο χρήστης μπορεί να διαχειριστεί την αποθήκη εφαρμόζοντας λειτουργίες σε όλα τα στοιχεία της κλάσης. Οι κλάσεις δεν

χρησιμοποιούνται για να επιβεβαιώσουν την ορθότητα ενός προγράμματος, αλλά για την δημιουργία και τον χειρισμό των αντικειμένων. Στα περισσότερα συστήματα που ενσωματώνουν τον μηχανισμό των κλάσεων, αυτές μπορούν να τις διαχειριστούμε στον χρόνο που εκτελείτε όπως να το ενημερώσουμε ή να του περάσουμε παράμετρο. Στις περισσότερες περιπτώσεις, όσο παρέχουμε στο σύστημα αυξημένη ευελιξία και ομοιογένεια, καθιστάτε αδύνατος ο έλεγχος στην μεταγλώττιση. Φυσικά, υπάρχουν πολλές ομοιότητες μεταξύ τύπων και κλάσεων, τα ονόματα τους έχουν χρησιμοποιηθεί και με τις δύο σημασίες και οι διαφορές τους μπορεί να είναι διακριτικές σε κάποια συστήματα.

### 1.2.5 Ιεραρχίες κλάσεων και τύπων – Inheritance

Η κληρονομικότητα έχει δύο πλεονεκτήματα : είναι ένα πολύ ισχυρό εργαλείο μοντελισμού, επειδή δίνει μια συνοπτική και ακριβή περιγραφή του κόσμου και βοηθάει να παράγονται κοινές προδιαγραφές και υλοποιήσεις σε εφαρμογές.

Ένα παράδειγμα θα βοηθήσει να διευκρινίσουμε το ενδιαφέρον που έχει ένα σύστημα που παρέχει ένα μηχανισμό κληρονομικότητας. Ας θέσουμε ότι έχουμε υπαλλήλους και μαθητές. Κάθε υπάλληλος έχει όνομα, ηλικία άνω τον 18 και μισθό, αυτός μπορεί να πεθάνει, να παντρευτεί και να πληρωθεί. Κάθε μαθητής έχει μία ηλικία, ένα όνομα και ένα σύνολο από βαθμούς. Και αυτός μπορεί να πεθάνει, να παντρευτεί και να υπολογιστεί η τελική του βαθμολογία. Σε ένα σχεσιακό σύστημα, ο σχεδιαστής της βάσης δεδομένων καθορίζει τις σχέσεις για τον υπάλληλο, τις σχέσεις για τον μαθητή, γράφει κώδικα για τις λειτουργίες πεθαίνω, παντρεύομαι, πληρώνομαι για της σχέσεις του υπαλλήλου και γράφει ξανά κώδικα για τις λειτουργίες πεθαίνω, παντρεύομαι και υπολογισμό τελικού βαθμού. Σύνολο, ο προγραμματιστής γράφει έξι προγράμματα. Σε ένα αντικειμενοστρεφές σύστημα, που υποστηρίζει κληρονομικότητα, αναγνωρίζουμε πως ο υπάλληλος και ο μαθητής είναι άνθρωποι, οπότε έχουν κάτι κοινό (το γεγονός ότι είναι άνθρωποι) και επίσης έχουν και κάτι συγκεκριμένο. Εισάγουμε ένα τύπο άνθρωπος, όπου έχει χαρακτηριστικά το όνομα και την ηλικία και γράφουμε τις λειτουργίες πεθαίνω και παντρεύομαι για αυτόν. Έπειτα, δηλώνουμε πως ο υπάλληλος είναι μια ξεχωριστή ομάδα ανθρώπων, που κληρονομεί χαρακτηριστικά και λειτουργίες του ανθρώπου, αλλά έχει ξεχωριστό χαρακτηριστικό τον μισθό και μία συγκεκριμένη λειτουργία την πληρωμή. Ομοίως, αυτά ισχύουν και για τον μαθητή. Έτσι έχουμε λοιπόν καταφέρει να δημιουργήσουμε ένα πιο δομημένο και πιο προφανή σχήμα και έχουμε γράψει μόνο τέσσερα προγράμματα. Στην τελική η κληρονομικότητα βοηθάει στο ότι ο κώδικας να είναι επαναχρησιμοποιήσιμος, διότι κάθε πρόγραμμα είναι σε επίπεδο στο οποίο ο μεγαλύτερος αριθμός αντικειμένων μπορεί να τον μοιραστεί.

Υπάρχουν τουλάχιστον τέσσερις τύποι κληρονομικότητας : κληρονομικότητα με αντικατάσταση, κληρονομικότητα με ενσωμάτωση, κληρονομικότητα με

περιορισμούς και κληρονομικότητα με εξειδίκευση. Στην κληρονομικότητα με αντικατάσταση, λέμε ότι ένας τύπος *t* κληρονομεί από έναν τύπο *t'*, εάν μπορούμε να εφαρμόσουμε περισσότερες λειτουργίες στα αντικείμενα τύπου *t* από ότι στα *t'*. Οπότε, οπουδήποτε έχουμε ένα αντικείμενο τύπου *t'* μπορούμε να το αντικαταστήσουμε με ένα αντικείμενο τύπου *t*. Αυτός ο τύπος κληρονομικότητας είναι βασισμένος στην συμπεριφορά και όχι στις τιμές. Στην κληρονομικότητα με ενσωμάτωση ανταποκρινόμαστε στην έννοια της κατάταξης. Δηλώνει ότι, το *t* είναι υποτύπος του *t'* αν κάθε αντικείμενο του *t* είναι και αντικείμενο του *t'*. Αυτός ο τύπος κληρονομικότητας είναι βασισμένος στην δομή και όχι στις λειτουργίες. Η κληρονομικότητα με περιορισμούς είναι υποκατηγορία της κληρονομικότητας με ενσωμάτωση. Ένας τύπος *t* είναι υποτύπος του *t'*, αν αποτελείτε από όλα τα αντικείμενα του *t* που να πληρούν κάποιον περιορισμό. Ένα παράδειγμα είναι ενός έφηβου που είναι υποκατηγορία του ανθρώπου, δεν έχει παραπάνω χαρακτηριστικά ή λειτουργίες, αλλά πρέπει να πληροί κάποιους συγκεκριμένους περιορισμούς (ηλικία μεταξύ 13 – 19). Και τέλος, με την κληρονομικότητα με εξειδίκευση, ένας τύπος t είναι υποτύπος ενός t', εάν τα αντικείμενα του τύπου t είναι αντικείμενα t' τα οποία περιέχουν ποιο λεπτομερείς πληροφορίες.

### 1.2.6 Υπερφόρτωση – Αργή σύνδεση, Overriding – Late binding

Υπάρχουν περιπτώσεις που ο προγραμματιστής χρειάζεται να χρησιμοποιεί το ίδιο όνομα για διαφορετικές μεθόδους. Σκεφτείτε για παράδειγμα μια λειτουργία εμφάνισης, παίρνει ένα αντικείμενο σαν παράμετρο και επιστρέφει έξοδο στην οθόνη. Ανάλογα το είδος του αντικειμένου, θέλουμε να χρησιμοποιήσουμε διαφορετικές λειτουργίες εμφάνισης. Εάν το αντικείμενο είναι εικόνα θέλουμε να εμφανιστεί στην οθόνη, αν είναι άνθρωπος θέλουμε να μας εμφανίσει τα χαρακτηριστικά του και αν το αντικείμενο είναι γράφος θέλουμε να μας εμφανίσει την αναπαράσταση του. Σκεφτείτε τώρα να εμφανίσουμε ένα σύνολο (set), το είδος των οποίων μελών του είναι αδιευκρίνιστο στην φάση της μεταγλώττισης. Σε μια εφαρμογή που χρησιμοποιεί ένα συμβατικό σύστημα, θα έχουμε τρεις λειτουργίες : εμφάνισε-εικόνα, εμφάνισε-άνθρωπο, εμφάνισε-γράφο. Ο προγραμματιστής θα ελέγξει τον τύπο κάθε αντικείμενου στο σύνολο και θα χρησιμοποιήσει την αντίστοιχη λειτουργία εμφάνισης. Αυτό αναγκάζει τον προγραμματιστή, να είναι ενήμερος για όλους τους πιθανούς τύπους στο σύνολο, να είναι ενήμερος για όλες τις πιθανές λειτουργίες που σχετίζονται και να τις χρησιμοποιήσει αναλόγως.

Σε ένα αντικειμενοστρεφές σύστημα, ορίζουμε την λειτουργία εμφάνισης στο επίπεδο του αντικειμένου (το ποιο γενικό τύπο στο σύστημα). Οπότε, η λειτουργία έχει ένα όνομα και μπορεί να χρησιμοποιηθεί διαφορετικά από το γράφο, άνθρωπο και εικόνα. Ωστόσο, ξαναορίζουμε την υλοποίηση της λειτουργίας για κάθε τύπο ξεχωριστά σύμφωνα με αυτό που θέλουμε να εκτελέσει. Αυτός ο ξαναορισμός ονομάζεται overriding. Αυτό καταλήγει σε ένα όνομα μεθόδου που

υποδηλώνει τρία διαφορετικά προγράμματα. Αυτό ονομάζεται υπερφόρτωση (overloading). Για να εμφανίσουμε το σύνολο των στοιχείων, απλά εφαρμόζουμε την λειτουργία στο καθένα από αυτά και αφήνουμε το σύστημα να επιλέξει την κατάλληλη υλοποίηση στην φάση της εκτέλεσης.

Εδώ, κερδίζουμε ένα διαφορετικό πλεονέκτημα : οι προγραμματιστές που υλοποίησαν τους τύπους αυτούς ακόμα γράφουν το ίδιο πλήθος προγραμμάτων. Αλλά, ο προγραμματιστής δεν χρειάζεται να ανησυχεί για τα τρία διαφορετικά προγράμματα. Από την άλλη μεριά, ο κώδικας είναι πιο απλός γιατί δεν υπάρχει κανένα σενάριο για επιλογή κατάλληλης μεθόδου. Επίσης, ο κώδικας είναι ποιο συντηρήσιμος, αφού άμα εισάγουμε έναν καινούργιο τύπο, το πρόγραμμα εμφάνισης θα συνεχίσει να δουλεύει χωρίς αλλαγές (προϋπόθεση να έχει γίνει override της μεθόδου εμφάνισης στο νέο τύπο).

Προκειμένου να δοθεί αυτή η νέα λειτουργικότητα, το σύστημα δεν μπορεί να συνδέσει τα ονόματα των λειτουργιών με τα προγράμματα στη μεταγλώττιση. Γι’ αυτό, τα ονόματα των λειτουργιών πρέπει να μεταφραστούν σε διευθύνσεις προγραμμάτων στον χρόνο εκτέλεσης. Αυτή η καθυστερημένη μετάφραση ονομάζεται late binding.

### 1.2.7 Υπολογιστική πληρότητα – Computational completeness

Από την οπτική γωνία μιας γλώσσας προγραμματισμού, αυτή η ιδιότητα είναι προφανής : σημαίνει πως μπορούμε να χρησιμοποιήσουμε οποιαδήποτε υπολογιστική συνάρτηση, χρησιμοποιώντας την DML του συστήματος της βάσεως δεδομένων. Από την οπτική γωνία μιας βάσης δεδομένων, αυτό αποτελεί καινοτομία, αφού η SQL δεν είναι ολοκληρωμένη. Δεν υποστηρίζουμε ότι οι σχεδιαστές αντικειμενοστρεφών βάσεων δεδομένων σχεδιάζουν νέες γλώσσες προγραμματισμού: η υπολογιστική πληρότητα μπορεί να εισαχθεί από μια προφανή σύνδεση με τις υπάρχουσες γλώσσες. Τελικά, τα περισσότερα συστήματα χρησιμοποιούν μια υπάρχουσα γλώσσα προγραμματισμού.

### 1.2.8 Επεκτασιμότητα –Extensibility

Η βάση δεδομένων έρχεται με ένα σύνολο από ορισμένους τύπους. Οι τύποι αυτοί μπορούν να χρησιμοποιηθούν ελεύθερα από τους προγραμματιστές για να γράψουν τις εφαρμογές τους. Αυτό το σύνολο τύπων πρέπει να είναι επεκτάσιμοι με την ακόλουθη ερμηνεία : υπάρχει ανάγκη να ορίσουμε νέους τύπους και δεν υπάρχει περιορισμός στη χρήση, ορισμένων από το σύστημα ή από τον χρήστη, τύπων. Φυσικά, θα υπάρχει μια ισχυρή διαφορά στον τρόπο που θα χειρίζεται το σύστημα τους τύπους που είναι ορισμένοι από το σύστημα με αυτούς του χρήστη,

αλλά αυτό πρέπει να μην είναι εμφανές στην εφαρμογή και στον προγραμματιστή της εφαρμογής. Θυμηθείτε ότι αυτός ο ορισμός τύπων συμπεριλαμβάνει τον ορισμό λειτουργιών στους τύπους. Σημειώστε πως οι προδιαγραφές της ενθυλάκωσης καθορίζει ότι θα υπάρχει ένας μηχανισμός για τον ορισμό νέων τύπων. Αυτές οι προδιαγραφές ενισχύουν αύτη τη δυνατότητα λέγοντας πως νέοι τύποι πρέπει να έχουν την ίδια κατάσταση με αυτούς που υπάρχουν. Ωστόσο, δεν χρειαζόμαστε οι τύποι συλλογών (set, tuple ,list κτλ) να είναι επεκτάσιμοι.

### 1.2.9 Διατηρησιμότητα –Persistence

Αυτή η απαίτηση είναι εμφανής από την οπτική γωνία μιας βάσης δεδομένων, αλλά καινοτομία από την οπτική γωνία της γλώσσας προγραμματισμού[18]. Η διατηρησιμότητα είναι η ικανότητα να έχει ο προγραμματιστής διατηρήσιμα τα δεδομένα του μετά από την εκτέλεση μιας διεργασίας, για να τα ξαναχρησιμοποιήσει σε μία άλλη. Κάθε αντικείμενο, ανεξάρτητα από τον τύπο του, έχει την δυνατότητα να γίνει διατηρήσιμο ως έχει. Επίσης ο χρήστης δεν θα πρέπει να χρειάζεται να μετακινήσει ή να αντιγράψει τα δεδομένα για να γίνουν διατηρήσιμα.

### 1.2.10 Διαχείριση δευτερεύουσας μνήμης – Secondary storage management

Η διαχείριση της δευτερεύουσας μνήμης είναι ένα κλασικό χαρακτηριστικό των συστημάτων βάσεων δεδομένων. Συνήθως, υποστηρίζεται από ένα πλήθος μηχανισμών. Αυτά είναι διαχείριση ευρετηρίου (index), συσταδοποίηση (data clustering), data buffering, access path selection και query optimization. Κανένα από αυτά δεν είναι εμφανή στο χρήστη, είναι απλά χαρακτηριστικά βελτιστοποίησης. Όμως, είναι τόσο κρίσιμα όσον αφορά την απόδοση, που η απουσία τους θα κρατήσει το σύστημα από την εκτέλεση ορισμένων εργασιών (απλά γιατί χρειάζονται πολύ χρόνο). Το ποιο σημαντικό είναι ότι δεν είναι εμφανή. Ο προγραμματιστής δεν χρειάζεται να γράψει κώδικα για να διατηρήσει τους δείκτες, για την κατανομή του δίσκου αποθήκευσης ή για την μετακίνηση δεδομένων από το δίσκο στην κύρια μνήμη. Έτσι, θα πρέπει να υπάρχει μια σαφή ανεξαρτησία μεταξύ λογικού και φυσικού επιπέδου στο σύστημα.

### 1.2.11 Ταυτοχρονισμός – Concurrency

Με σεβασμό στην διαχείριση πολλαπλών χρηστών που αλληλεπιδρούν ταυτόχρονα με το σύστημα, το σύστημα πρέπει να τους παρέχει το ίδιο επίπεδο εξυπηρέτησης που το τρέχων σύστημα βάσεων δεδομένων παρέχει. Πρέπει

επιπλέον να εξασφαλίζει την αρμονική συνύπαρξη μεταξύ των χρηστών που εργάζονται ταυτόχρονα στη βάση δεδομένων. Το σύστημα, ως εκ τούτου, υποστηρίζει την έννοια της ατομικότητας μιας ακολουθίας λειτουργιών και της ελεγχόμενης κατανομής. Η σειριοποιησιμότητα των λειτουργιών πρέπει να παρέχετε, αν και μπορεί να παρέχονται λιγότερο αυστηρές εναλλακτικές.

### 1.2.12 Επαναφορά –Recovery

Εδώ πάλι, το σύστημα πρέπει να παρέχει το ίδιο επίπεδο υπηρεσιών με όπως τα σημερινά συστήματα βάσεων δεδομένων. Ως εκ τούτου, σε περίπτωση βλάβης υλικού ή λογισμικού, το σύστημα πρέπει να επανέρχεται, δηλαδή να επαναφέρει τα δεδομένα σε μια συνεπή κατάσταση. Η βλάβες υλικού περιέχουν και τις βλάβες επεξεργαστή και των δίσκων.

### 1.2.13 Μηχανισμός ερωτημάτων–Ad hoc query facility

Το κύριο πρόβλημα εδώ είναι να παρέχουμε την λειτουργικότητα μιας γλώσσας ερωτημάτων. Δεν χρειάζεται να το επιτύχουμε χρησιμοποιώντας την μορφή των γλωσσών ερωτημάτων, απλά να παρέχεται η υπηρεσία αυτή. Για παράδειγμα, ένας γραφικός περιηγητής θα μπορούσε να είναι επαρκής για την εκπλήρωση αυτής της λειτουργίας. Η υπηρεσία αυτή δίνει την δυνατότητα στον χρήστη να κάνει απλές ερωτήσεις στην βάση δεδομένων εύκολα. Το προφανές σημείο αναφοράς είναι φυσικά τα σχεσιακά συστήματα, έτσι η δοκιμή είναι να λάβει μία σειρά αντιπροσωπευτικών σχεσιακών ερωτημάτων και να ελέγξει εάν μπορούν να διατυπωθούν με τον ίδιο φόρτο εργασίας. Σημειώστε, πως αυτός ο μηχανισμός μπορεί να υποστηρίζεται από την γλώσσα διαχείρισης δεδομένων (DML) ή ένα υποσύνολό της. Πιστεύεται πως ο μηχανισμός ερωτημάτων πρέπει να πληροί τα ακόλουθα τρία κριτήρια : (i) Πρέπει να είναι υψηλού επιπέδου, π.χ., ένας πρέπει να μπορεί να εκφράσει (με λίγα λόγια ή με μερικά πατήματα του ποντικιού) συνοπτικά τα ερωτήματα. Αυτό σημαίνει ότι πρέπει να είναι αρκετά δηλωτική, δηλαδή, πρέπει να δίνει έμφαση στο *ΤΙ* παρά στο *ΠΩΣ*. (ii) Θα πρέπει να είναι αποτελεσματική. Δηλαδή, η διατύπωση του ερωτήματος θα πρέπει να επιδέχεται κάποια μορφή βελτιστοποίησης του. (iii) Θα πρέπει να είναι ανεξάρτητο της εφαρμογής, π.χ., θα πρέπει να λειτουργεί σε οποιαδήποτε βάση δεδομένων. Αυτές οι τελευταίες απαιτήσεις εξαλείφουν τους μηχανισμούς ερωτημάτων που είναι εξαρτώμενες από την εφαρμογή ή που απαιτούν κώδικα για την υλοποίηση εναλλακτικών διαδικασιών και κάθε τύπο που ορίζει ο χρήστης.

## 1.3 Προαιρετικά χαρακτηριστικά – Optional Features

### 1.3.1 Εισαγωγή

Σε αυτή την ενότητα θα ασχοληθούμε με πράγματα που βελτιώνουν το σύστημα, αλλά δεν είναι υποχρεωτικά για να καθιστούν ένα σύστημα ως αντικειμενοστρεφές σύστημα βάσεων δεδομένων. Μερικά από αυτά τα χαρακτηριστικά είναι αντικειμενοστρεφούς χαρακτήρα, όπως η πολλαπλή κληρονομικότητα. Περιλαμβάνονται σε αυτή την κατηγορία γιατί, έστω και αν κάνουν το σύστημα ποιο αντικειμενοστρεφή, δεν εμπεριέχονται στις βασικές προϋποθέσεις. Άλλα χαρακτηριστικά είναι αυτά των βάσεων δεδομένων, όπως ο σχεδιασμός και η διαχείριση των συναλλαγών. Αυτά τα χαρακτηριστικά, συνήθως βελτιώνουν την λειτουργικότητα ενός συστήματος βάσεων δεδομένων, αλλά δεν είναι από τα βασικά χαρακτηριστικά ενός συστήματος βάσεων δεδομένων και δεν έχουν καμία σχέση από την πλευρά της αντικειμενοστρεφής λογικής. Γενικά, τα περισσότερα από αυτά έχουν σκοπό να εξυπηρετήσουν νέες εφαρμογές, όπως (CAD/CAM, CASE, office automation, κτλ.) που είναι πιο προσανατολισμένα στην εφαρμογή παρά στην τεχνολογία. Επειδή, πολλά αντικειμενοστρεφή συστήματα βάσεων δεδομένων στοχεύουν αυτή τη στιγμή σε αυτές τις νέες εφαρμογές, υπάρχει μια σύγχυση μεταξύ αυτών των χαρακτηριστικών και της αντικειμενοστρεφής φύσης του συστήματος.

### 1.3.2 Πολλαπλή κληρονομικότητα – Multiply inheritance

Το αν θα παρέχετε πολλαπλή κληρονομικότητα είναι επιλογή του συστήματος. Δεδομένου ότι συμφωνία στην αντικειμενοστρεφή κοινότητα για την πολλαπλή κληρονομικότητα δεν έχει επιτευχθεί, θεωρούμε πως η παροχή του είναι προαιρετική. Σημειώνουμε ότι αν κάποιος αποφασίσει να χρησιμοποιήσει ένα σύστημα που υποστηρίζει πολλαπλή κληρονομικότητα, υπάρχουν πολλές πιθανές λύσεις για την αντιμετώπιση του προβλήματος της επίλυσης συγκρούσεων.

### 1.3.3 Έλεγχος τύπων και τεκμηρίωση – Type checking and type inferencing

Ο βαθμός του ελέγχου των τύπων που θα εκτελέσει το σύστημα κατά την μεταγλώττιση έχει μείνει ανοιχτό, αλλά όσο περισσότερο τόσο το καλύτερο. Η βέλτιστη κατάσταση είναι εκείνη όπου ένα πρόγραμμα το οποίο έχει γίνει δεκτό από τον μεταγλωττιστή δεν παράγει σφάλματα κατά την εκτέλεση. Η ποσότητα τεκμηρίωσης του τύπου είναι επίσης ανοιχτό στο σχεδιαστή του συστήματος, όσο περισσότερο τόσο το καλύτερο. Η ιδανική κατάσταση είναι αυτή που μόνο η

βασική τύποι πρέπει να δηλωθούν και το σύστημα συμπεραίνει τους προσωρινούς τύπους.

### 1.3.4 Κατανομή – Distribution

Πρέπει να είναι εμφανής ότι αυτό το χαρακτηριστικό είναι συναφές με τη φύση του αντικειμενοστρεφούς συστήματος. Έτσι ένα σύστημα βάσεων δεδομένων μπορεί να είναι κατανεμημένο ή όχι.

### 1.3.5 Σχεδιασμός συναλλαγών – Design transactions

Στις περισσότερες νέες εφαρμογές, το μοντέλο των κλασσικών επαγγελματικών συστημάτων βάσεων δεδομένων δεν είναι ικανοποιητικός. Οι συναλλαγές έχουν την τάση να είναι αρκετά μεγάλες και το σύνηθες κριτήριο σειριοποιησιμότητας δεν είναι επαρκής. Έτσι, πολλά συστήματα διαχείρισης βάσεων δεδομένων υποστηρίζουν τον σχεδιασμό συναλλαγών (μεγάλων συναλλαγών ή εμφωλευμένων συναλλαγών).

### 1.3.6 Εκδόσεις – Versions

Οι περισσότερες νέες εφαρμογές (CAD/CAM και CASE) συνεπάγουν μια δραστηριότητα σχεδιασμού και απαιτούν κάποια μορφή εκδόσεων. Έτσι, πολλά αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων υποστηρίζουνε τις εκδόσεις. Υπενθυμίζουμε πως το να παρέχουμε ένα μηχανισμό για εκδόσεις δεν είναι κομμάτι τον βασικών χαρακτηριστικών του συστήματος.

## 1.4 Ανοιχτά Χαρακτηριστικά

### 1.4.1 Εισαγωγή

Κάθε σύστημα που υποστηρίζει τους κανόνες από το ένα ως το δεκατρία δικαιούται να φέρει τον χαρακτηρισμό του αντικειμενοστρεφούς συστήματος διαχείρισης βάσεων δεδομένων. Όταν σχεδιάζονται τέτοια συστήματα, υπάρχουν πολλές επιλογές σχεδιασμού που μπορούν να γίνουν. Αυτό είναι ο βαθμός ελευθερίας που μπορούν να έχουν οι δημιουργοί του αντικειμενοστρεφούς

συστήματος βάσεων δεδομένων. Αυτά τα χαρακτηριστικά διαφέρουν από τα υποχρεωτικά χαρακτηριστικά, υπό την έννοια ότι δεν υπάρχει συναίνεση στην επιστημονική κοινότητα σχετικά με αυτές. Διαφέρουν επίσης, από τα προαιρετικά χαρακτηριστικά στο ότι δεν γνωρίζουμε ποια από τα εναλλακτικά είναι περισσότερο ή λιγότερο αντικειμενοστρεφή.

### 1.4.2 Παράδειγμα προγραμματισμού – Programming paradigm

Δεν βλέπουμε κανένα λόγο γιατί θα πρέπει να επιβάλουν ένα παράδειγμα προγραμματισμού περισσότερο από κάποιο άλλο. Το στυλ του λογικού προγραμματισμού (Bancilhon 86) [19], (Zaniolo 86) [20], το στυλ του διαδικαστικού προγραμματισμού (Albano et al. 1986) [21], (Banerjee et al. 87) ή το στυλ του επιτακτικού προγραμματισμού (Stroustrup 86), (Eiffel 87) [22], (Atwood 85), θα μπορούσαν όλες να επιλεχθούν για παράδειγμα προγραμματισμού. Άλλη λύση είναι το σύστημα να είναι ανεξάρτητο από τη μεθοδολογία του προγραμματισμού και να υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού (Skarra et al. 86) [23], (Bancilhon et al. 88).Επίσης, ο τρόπος σύνταξης είναι ελεύθερος και οι άνθρωποι θα μαλώνουν συνέχεια αν πρέπει κάποιος να γράψει “ john hire” ή “john.hire” ή “hire john” ή “hire(john)”.

### 1.4.3 Αναπαράσταση συστήματος – Representation system

Το σύστημα αναπαράστασης καθορίζεται από ένα σύνολο ατομικών τύπων και ένα σύνολο κατασκευαστών. Αν και έχουμε δώσει ένα ελάχιστο σύνολο ατομικών τύπων και κατασκευαστών (στοιχειώδη τύποι από τις γλώσσες προγραμματισμού και σύνολα, πλειάδες και κατασκευαστές λίστας) διαθέσιμοι για την περιγραφή της αναπαράστασης των αντικειμένων, μπορούν να επεκταθούν με πολλούς διαφορετικούς τρόπους.

### 1.4.4 Σύστημα τύπων – Type system

Εδώ, υπάρχει επίσης ελευθερία με σεβασμό στους διαμορφωτές των τύπων. Ο μόνος μηχανισμός σχηματισμού τύπων είναι η ενθυλάκωση. Μπορούν να υπάρξουν και άλλες διαμορφώσεις τύπων όπως οι γενικοί τύποι ή γεννήτριες τύπων (όπως ένα σύνολο set[T], όπου το T μπορεί να είναι ένας αυθαίρετος τύπος), περιορισμού, ένωσης και βέλους (λειτουργίες).

### 1.4.5 Ομοιομορφία – Uniformity

Υπάρχει μια έντονη συζήτηση σχετικά με τον βαθμό της ομοιομορφίας που πρέπει να περιμένει ο καθένας από αυτά τα συστήματα : είναι ένας τύπος αντικείμενο; Είναι μια μέθοδος αντικείμενο; Ή πρέπει αυτές οι έννοιες να αντιμετωπίζονται διαφορετικά; Μπορούμε να δούμε αυτό το πρόβλημα σε τρία διαφορετικά επίπεδα : το επίπεδο υλοποίησης, το επίπεδο γλώσσας προγραμματισμού και το επίπεδο διασύνδεσης. Στο επίπεδο της υλοποίησης, πρέπει κανείς να αποφασίσει αν η πληροφορία πρέπει να αποθηκεύεται ως πληροφορία είτε αν ένα σύστημα ad hoc πρέπει να υλοποιηθεί. Αυτό είναι ένα θέμα που και οι προγραμματιστές συστημάτων σχεσιακών βάσεων δεδομένων πρέπει να αντιμετωπίσουν όταν πρέπει να αποφασίσουν αν θα αποθηκεύσουνε το σχήμα σαν πίνακα ή σε κάποιο ad hoc τρόπο. Η απόφαση πρέπει να γίνεται με βάση την απόδοση και την ευκολία υλοποίησης. Όποια και αν είναι, η απόφαση παίρνεται, ωστόσο, ανεξάρτητα από εκείνη που επιλέχθηκε στο προηγούμενο επίπεδο. Στο επίπεδο της γλώσσας προγραμματισμού η ερώτηση είναι η εξής : είναι οι τύποι πρωτεύον οντότητες κλάσης στην σημασιολογία της γλώσσας; Η περισσότερη συζήτηση γίνεται σχετικά με αυτή την ερώτηση. Υπάρχουν, πιθανώς, πολλά στυλ ομοιομορφίας (συντακτικά ή σημασιολογικά). Η πλήρης ομοιομορφία σε αυτό το επίπεδο είναι ασυνεπής με το στατικό έλεγχο των τύπων. Εν κατακλείδι, στο επίπεδο της διασύνδεσης, μια άλλη ανεξάρτητη απόφαση πρέπει να παρθεί. Κάποιος ίσως θέλει να παρουσιάσει στο χρήστη μια ενιαία όψη των τύπων, αντικειμένων, μεθόδων, ακόμα και στη σημασιολογία της γλώσσας προγραμματισμού. Αυτές είναι έννοιες μιας διαφορετικής φύσης. Αντιθέτως, κάποιος θα μπορούσε να τα παρουσιάσει σαν διαφορετικές οντότητες, αν και οι γλώσσες προγραμματισμού τις βλέπουν σαν ίδια. Αυτή η απόφαση πρέπει να ληφθεί από ανθρώπινα κριτήρια.

## Επίλογος

Πολλοί άλλοι συγγραφείς, (Kim 88) [24] και (Dittrich 1986) [25] διαφωνούσαν ότι ένα αντικειμενοστρεφές σύστημα βάσεων δεδομένων είναι ένα σύστημα διαχείρισης βάσεων δεδομένων με ένα αντικειμενοστρεφές μοντέλο δεδομένων να διέπεται. Εάν κάποιος λάβει την έννοια ενός μοντέλου δεδομένων στην κοινή λογική που αυτά ειδικά περιλαμβάνουν τις συμπληρωματικές πτυχές πέραν από το record – orientation, αυτή η άποψη είναι σίγουρα σύμφωνη με την προσέγγισή μας. Ο Dittrich, (Dittrich 1986) και (Dittrich 1988) [26], εισήγαγε μια κατηγοριοποίηση των αντικειμενοστρεφών μοντέλων δεδομένων (και ακόλουθα των αντικειμενοστρεφών συστημάτων βάσεων δεδομένων). Αν υποστηρίζεται σύνθετα αντικείμενα, το μοντέλο ονομάζεται δομημένο αντικειμενοστρεφές

μοντέλο. Αν παρέχεται η επεκτασιμότητα, ονομάζεται συμπεριφορικό αντικειμενοστρεφές μοντέλο. Ένα πλήρες αντικειμενοστρεφές μοντέλο πρέπει να παρέχει και τα δύο χαρακτηριστικά. Ο ορισμός πρέπει επίσης να συμπεριλάβει την διατηρησιμότητα, την δευτερεύουσα μνήμη, τον ταυτοχρονισμό και την επαναφορά από βλάβη. Αυτό υποθέτει εμμέσως όλα τα άλλα χαρακτηριστικά (που είναι εφαρμόσιμα, ανάλογα την διαφορετικότητα των κλάσεων). Τελικά, αυτή η προσέγγιση είναι ποιο ελεύθερη από αυτή που εμείς δίνουμε. Ωστόσο, αφού τα περισσότερα τωρινά συστήματα και πρωτότυπα δεν καλύπτουν όλες τις προϋποθέσεις που υπαγορεύουμε στον ορισμό μας, αυτή η κατηγοριοποίηση μας παρέχει ένα χρήσιμο υπόβαθρο για να συγκρίνουμε την υπάρχουσα και την τρέχουσα δουλειά μας.

Προτείνουμε, μια συλλογή από χαρακτηριστικά που ορίζουν ένα αντικειμενοστρεφές σύστημα βάσεων δεδομένων. Οι ορισμοί που περιγράφηκαν παραπάνω είναι η ποιο λεπτομερείς για το τι ακριβώς είναι ένα αντικειμενοστρεφές σύστημα βάσεων δεδομένων. Η επιλογή αυτών των χαρακτηριστικών και η διερμηνεία τους προέρχεται από τη εμπειρία του καθορισμού και υλοποίησης τέτοιων συστημάτων. Επιπλέον εμπειρία με την σχεδίαση, υλοποίηση και τυποποίηση των αντικειμενοστρεφών συστημάτων βάσεων δεδομένων σίγουρα θα αλλάξει και θα να αναπροσδιορίσουμε την στάση μας. Στόχος μας είναι να εκφραστεί μια συγκεκριμένη πρόταση, να συζητηθεί, να σχολιαστεί και να αναλυθεί από την επιστημονική κοινότητα.

Στο επόμενο κεφάλαιο θα παρουσιάσουμε μερικά αντικειμενοστρεφή συστήματα βάσεων δεδομένων, θα δούμε τα χαρακτηριστικά που υποστηρίζουν, θα τα σχολιάσουμε και θα τα συγκρίνουμε τα οκτώ πιο επικρατέστερα στην αγορά, για να δούμε πόσα από τα χαρακτηριστικά που περιγράψαμε υποστηρίζονται από αυτά και ποια είναι αυτά.

# ΚΕΦΑΛΑΙΟ 2

## Συγκριτική μελέτη Αντικειμενοστρεφών συστημάτων διαχείρισης βάσεων δεδομένων

### Εισαγωγή

Τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων προέρχονται από τις αντικειμενοστρεφές γλώσσες προγραμματισμού. Επιπλέον, τα αντικειμενοστρεφή συστήματα βάσεων δεδομένων υποστηρίζουν τα χαρακτηριστικά που έκαναν τις αντικειμενοστρεφές γλώσσες προγραμματισμού να επιτύχουν και επιπλέον αυτά ενοποιούν αυτές παρέχοντας την δυνατότητα να αποθηκεύουν μόνιμα όπως σε κάθε σύστημα διαχείρισης βάσεων δεδομένων.

Την τελευταία δεκαετία πολλά αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων έχουν αναπτυχτεί. Τα ποιο γνωστά από αυτά είναι : Gemstone (Smalltalk) [27], Jdoinstruments (objectDB) [28], Versant Object Database [29], ObjectStore [30], Jade [31], Matisse [32], db4objects [33], Objectivity [34], Progress Software [35], EyeDB Object Oriented Database Management System [36], McObject Perst [37], Cache [38], ODABA [39], Eloquera [40], Generic Object Oriented Database System (GOODS) [41], JODB (Java Objects Database) [42], MyOODB [43], NeoDatis ODB [44], Orient ODBMS [45], Ozone Database Project [46], Statice [47], VOSS (Virtual Object Storage System) [48], Obsidian Dynamics (DTS/S1) [49].

Σε αυτό το κεφάλαιο θα οργανώσουμε τα χαρακτηριστικά που ένα αντικειμενοστρεφές σύστημα διαχείρισης βάσεων δεδομένων πρέπει να υποστηρίζει και θα σχεδιάσουμε ένα συγκριτικό πίνακα μεταξύ των οκτώ πιο σημαντικών API για Object Persistence και αντικειμενοστρεφών συστημάτων βάσεων δεδομένων. Η δομή του κεφαλαίου είναι η ακόλουθη. Θα ακολουθήσει μια ανάλυση των χαρακτηριστικών των αντικειμενοστρεφών συστημάτων διαχείρισης βάσεων δεδομένων, καθώς και γιατί τα χρησιμοποιούμε, έπειτα θα αναλύσουμε τα οκτώ επικρατέστερα API και στη επόμενη παράγραφο θα παρουσιάσουμε την σύγκριση ανάμεσα σε αυτά τα APIs και τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων. Και τελικά, θα αναφερθούμε στον επίλογο για τα συμπεράσματά μας.

### 2.1 Χαρακτηριστικά ενός συστήματος διαχείρισης βάσεων δεδομένων

Σε αυτό το κομμάτι θα οργανώσουμε τα χαρακτηριστικά που πρέπει να υποστηρίζει ένα αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων σαν

αποτέλεσμα από την έρευνά μας. Σίγουρα, είναι πολύ σημαντικό για ένα σύστημα διαχείρισης βάσεων δεδομένων να υποστηρίζει το **ODMG 3.0 standard** [50]. Αυτό εμπεριέχει τα βασικά συστατικά του ODMG 3.0 standard, ονομαστικά **Object Definition Language (ODL)**, **Object Query Language (OQL)**, και επιπλέον ισχυροί τύποι όπως οι συλλογές και οι σύνθετοι τύποι. Στο πρότυπο ODMG 3.0 προτείνονται πέντε τύποι συλλογών, **set**, **bag**, **list**, **array** και **dictionary**, που σημαίνει ότι περιμένουμε από ένα σύστημα διαχείρισης βάσεων δεδομένων να τα υποστηρίζει τα περισσότερα από αυτά. Σύνθετοι τύποι δεδομένων που αναφέρονται καθορίζονται από τον χρήστη όπως τις κλάσης ή τις δομές. Επιπλέον, ένα μεγάλο ζήτημα στο ODMG 3.0 είναι **ο τύπος της κληρονομικότητας** που υποστηρίζεται. Όλα τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων υποστηρίζουν μονή κληρονομικότητα εκτός από δύο που υποστηρίζουν πολλαπλή κληρονομικότητα.

Είναι επίσης σημαντικό, τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων να υποστηρίζουν τα παραδοσιακά χαρακτηριστικά μιας βάσης δεδομένων. Έτσι το σύστημα πρέπει να υποστηρίζει **έλεγχο ταυτοχρονισμού**, **επαναφορά από βλάβη** και **ευρετήρια**. Από την άλλη πλευρά, είναι χρήσιμο από ένα σύστημα να παρέχει έναν **μηχανισμό ερωτημάτων**. Στην έρευνά μας βρήκαμε να χρησιμοποιούνται τρεις διαφορετικοί τύποι ερωτημάτων, όπως ODMG 3.0 OQL[51], NoSQL[52] και LINQ[53].

Άλλη μια πλευρά αυτής της έρευνας είναι η μεταφερσιμότητα του λογισμικού. Έτσι ελέγχουμε και σε ποιες πλατφόρμες τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων είναι συμβατά (Windows, Mac, Linux, Android). Επιπλέον, ένα ακόμα χαρακτηριστικό που είναι σημαντικό είναι αν μπορούμε να χρησιμοποιήσουμε το σύστημα σε μία αρχιτεκτονική τριών επιπέδων, δηλαδή, αν μπορούμε να εγκαταστήσουμε το σύστημα σε ένα database server και να έχουμε πρόσβαση από τερματικό (client).

Η διασύνδεση που παρέχει στον χρήστη ένα σύστημα διαχείρισης αντικειμενοστρεφών βάσεων δεδομένων, είναι άλλη μια κατηγορία σημαντικών χαρακτηριστικών που πρέπει να υποστηρίζει καθώς γίνεται ποιο αποτελεσματικό όταν υπάρχει ένα φιλικό για τον χρήστη περιβάλλον. Έτσι, τρεις τύποι διασύνδεσης είναι αυτοί που συναντάμε σε αυτά τα συστήματα (α) γραφική διασύνδεση χρήστη (GUI), με σκοπό να μπορούμε να διαχειριστούμε όλες τις υπηρεσίες που παρέχει το σύστημα, (β) διασύνδεση χρήστη βάσης δεδομένων, με σκοπό την σχεδίαση και παρουσίαση της δομής της βάσης δεδομένων και των πραγματικών δεδομένων και (γ) ο εξερευνητής δεδομένων (data explorer), για να παρουσιάσει σε μία φόρμα τα δεδομένα. Επίσης, ένα ακόμα πλεονέκτημα στις βάσεις δεδομένων είναι η δυνατότητα να εξάγουμε τα δεδομένα σε XML έγγραφο που είναι το πρότυπο για πολλές διαδικτυακές εφαρμογές. Εν κατακλείδι, εάν ένα σύστημα διαχείρισης βάσεων δεδομένων υποστηρίζει JDBC[28] μεταδεδομένα από τα δεδομένα των αντικειμένων είναι πιθανό να μπορούμε να δημιουργήσουμε δυναμικές εφαρμογές.

## 2.2 Γιατί να χρησιμοποιήσουμε ένα αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων

Τα πρώτα και ποιο σημαντικά συστήματα βάσεων δεδομένων χρησιμοποιήθηκαν από τις επιχειρήσεις και τους διαχειριστές κυρίως τραπεζικών εφαρμογών για να διατηρήσουν τις πληροφορίες σχετικά με τους πελάτες τους και λογαριασμούς τους και επίσης για τις εφαρμογές που κρατούσαν δεδομένα αρχειοθετώντας τα σαν ένα σύστημα αποθήκης. Στη δεκαετία του 1980, νέες εφαρμογές data - intensive αναδείχθηκαν σαν αποτέλεσμα της καινοτομίας του υλικού. Έπειτα, τα παραδοσιακά συστήματα διαχείρισης βάσεων δεδομένων που βασίζονται σε ένα σχεσιακό μοντέλο δεδομένων, ήταν πλέον ανεπαρκής. Παράδειγμα τέτοιων εφαρμογών είναι :

- Πολυμεσικές βάσεις δεδομένων που χρειαζόντουσαν να αποθηκεύουν κομμάτια ήχου, εικόνας, κειμένου και την ικανότητα να τα συνδυάζει όλα αυτά με ένα συνεπή τρόπο.
- Γεωγραφικά συστήματα πληροφοριών (GIS) που χρειάζονται να αποθηκεύουν διαφορετικούς τύπους χαρτογράφησης και στατιστικών δεδομένων τα οποία μπορούν να υποδιαιρεθούν και να αντιπαραβάλλονται από επικαλυπτόμενες περιοχές.
- Σχεδιασμός βάσεων δεδομένων που αποθηκεύουν δεδομένα και διαγράμματα που αφορούν σύνθετα συστατικά τα οποία μπορεί να συνδέονται μεταξύ τους για περαιτέρω σύνθετα συστατικά.

Τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων έχουν αναπτυχθεί για να καλύψουν αυτές τις ανάγκες που χρειάζονται οι εφαρμογές παρόμοιες με αυτές που παρουσιάσαμε παραπάνω. Η αντικειμενοστρεφή προσέγγιση παρέχει την ελαστικότητα που χρειάζεται γιατί δεν περιορίζεται από στους τύπους δεδομένων και τις γλώσσες ερωτημάτων στα παραδοσιακά συστήματα. Ένα από τα ποιο σημαντικά χαρακτηριστικά ενός αντικειμενοστρεφούς συστήματος διαχείρισης βάσεων δεδομένων είναι η ικανότητα να καθορίζει και την δομή των σύνθετων εφαρμογών αλλά και τις λειτουργίες για να διαχειριστεί αυτές τις δομές.

## 2.3 Παρουσίαση των επικρατέστερων αντικειμενοστρεφών συστημάτων διαχείρισης βάσεων δεδομένων

Δύο αντικειμενοστρεφή συστήματα βάσεων δεδομένων είναι τα JDO (Java Data Objects) και το JPA (Java Persistence API) που μαζί με τα άλλα έξι, είναι από τα ποιο επιτυχημένα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων

που είναι τα db4o, ObjectDB, Objectivity/DB, EyeDB, Perst και ObjectStore, που θα περιγραφούν παρακάτω.

### 2.3.1 Java Persistence API

Το JPA [55] είναι ένα API που μπορεί να χρησιμοποιηθεί σε εφαρμογές τύπου J2EE και J2SE. Παρέχει πρότυπα POJO (Plain Old Java Object) [58] και object relational mapping (OR mapping) [59] για την διατηρησιμότητα των δεδομένων στις εφαρμογές. Αυτή η διατηρησιμότητα, που έχει να κάνει με την αποθήκευση και την ανάκτηση των δεδομένων μιας εφαρμογής, μπορεί πλέον να προγραμματιστεί με το JPA ξεκινώντας από το EJB 3.0 ως αποτέλεσμα από το JSR220. Αυτό το API έχει δανειστεί πολλές ιδέες από κυρίαρχα πλαίσια για την διατηρησιμότητα (persistence frameworks), όπως Toplink (Oracle) [56] και Hibernate (JBoss) [57].

### 2.3.2 Java Data Object

Το Java Data Objects (JDO 2.0) [60] API είναι ένα πρότυπο διασύνδεσης βασισμένο στην αφαιρετική έννοια της διατηρησιμότητας του μοντέλου της JAVA, υλοποιημένο υπό την αιγίδα της κοινότητας διαδικασιών JAVA. Η πρωτότυπη έκδοση του JDO 1.0 είναι Java Specification Request 12 (JSR 12), και η τωρινή JDO 2.0 είναι JSR 243. Ξεκινώντας με την JDO 2.0, η υλοποίηση του API και το Technology Compability Kit (TCK) έγινε μέσα στο APACHE JDO open – source project [61]. Το JDO είναι ένας τρόπος να έχεις πρόσβαση στα διατηρήσιμα δεδομένα στις βάσεις δεδομένων χρησιμοποιώντας POJO για να αναπαραστήσει τα δεδομένα. Αυτή η προσέγγιση χωρίζει την διαχείριση των δεδομένων (γίνεται με πρόσβαση στα Java data members στα Java domain objects) από την διαχείριση των βάσεων δεδομένων (καλώντας τις μεθόδους διασύνδεσης του JDO). Αυτό οδηγεί σε έναν υψηλό βαθμό ανεξαρτησίας για τα δεδομένα από την οπτική γωνία της Java και της οπτικής γωνίας των δεδομένων από τις βάσεις δεδομένων.

### 2.3.3 db4o

Το db4o είναι ένα open source αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων που δίνει το έναυσμα σε προγραμματιστές Java και .NET να αποθηκεύουν και να ανακτούν οποιοδήποτε αντικείμενο με μία γραμμή κώδικα, εξαλείφοντας την ανάγκη για ορισμό ξεχωριστού μοντέλου δεδομένων. Τα αντικείμενα αποθηκεύονται με τον ίδιο ακριβώς τρόπο που ορίζονται στις εφαρμογές με java ή με .NET. Επιπλέον, υποστηρίζει την διατηρησιμότητα οποιουδήποτε αντικειμένου χωρίς να λάβει υπόψη του την πολυπλοκότητα του.

Είναι σχεδιασμένο για να είναι εμφωλευμένο στους clients ή σε άλλο συστατικό λογισμικού που το κάνει αόρατο στον χρήστη. Επιπλέον, η db4o δεν χρειάζεται ξεχωριστό μηχανισμό για εγκατάσταση, αλλά το παίρνουμε σαν μια εύκολα φορτώσιμη βιβλιοθήκη με ένα πολύ χαμηλό αποτύπωμα του ~670kb στην .NET και περίπου 1MB στην Java έκδοση. Η client/server έκδοση επιτρέπει στην db4o να επικοινωνεί μεταξύ των client και τις server – side εφαρμογές. Χρησιμοποιεί TCP/IP για την επικοινωνία και επιτρέπει την ρύθμιση του αριθμού πόρτας. Η υλοποίηση της επικοινωνίας γίνεται μέσω μηνυμάτων.

### 2.3.4 ObjectDB

Το ObjectDB είναι ένα πολύ ισχυρό αντικειμενοστρεφές σύστημα διαχείρισης βάσεων δεδομένων το οποίο είναι αξιόπιστο, εύκολο στην χρήση και τρομερά γρήγορο. Παρέχει όλες τις υπηρεσίες που πρέπει να παρέχει ένα σύστημα διαχείρισης βάσεων δεδομένων (αποθήκευση και ανάκτηση, ρύθμιση συναλλαγών, διαχείριση κλειδαριών, επεξεργασία ερωτημάτων κτλ). Πολλά χαρακτηριστικά του ObjectDB βασίζονται στα JPA και JDO APIs. Επιπλέον, χαρακτηριστικά που είναι σύνηθες στις σχεσιακές βάσεις δεδομένων (π.χ. κύρια κλειδιά κτλ) υποστηρίζονται επίσης από το ObjectDB.

### 2.3.5 Objectivity/DB

Το Objectivity/DB είναι ένα εμπορικό αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων από την Objectivity. Επιτρέπει στις εφαρμογές να κάνουν τα τυπικά αντικείμενα σε C++, Java, Python ή Smalltalk διατηρήσιμα. Το Objectivity/DB υποστηρίζει επίσης SQL/ODBC και XML. Λειτουργεί σε Linux, LynxOS, Unix και Windows.Το Objectivity/DB είναι ένα κατανεμημένο σύστημα βάσεων δεδομένων που παρέχει μια λογική άποψη σε ένα σύνολο βάσεων δεδομένων.

### 2.3.6 EyeDB

Το EyeDB είναι ένα αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων που υποστηρίζει τα πρότυπα του ODMG 3.0, το οποίο αναπτύχθηκε και υποστηρίζετε από την Γαλλική εταιρία Sysra. Το EyeDB παρέχει ένα υψηλό ένα εξελιγμένο μοντέλο δεδομένων (κληρονομικότητα, συλλογές, μέθοδοι, εναύσματα, περιορισμούς), μια γλώσσα ορισμού αντικειμένων βασισμένη στην ODMG ODL, μια γλώσσα ερωτημάτων αντικειμένων βασισμένη στην ODMG OQL και διεπαφές

προγραμματισμού για C++ και Java. Το EyeDB κατανέμεται κάτω από τους όρους του GNU Lesser General Public License.

### 2.3.7 Perst

Το Perst είναι ένα αντικειμενοστρεφές σύστημα διαχείρισης βάσεων δεδομένων ανοιχτού κώδικα το οποίο αναπτύχθηκε από την McObject LLC. Αυτό αποθηκεύει δεδομένα απευθείας σε Java αντικείμενα, που αυξάνει την απόδοση στον χρόνο εκτέλεσης. Είναι συμπαγές, με ένα πυρήνα που εμπεριέχει 5000 γραμμές κώδικα και έτσι απαιτεί ελάχιστους πόρους αφού έχει τόσο μικρό αποτύπωμα. Είναι αξιόπιστο και υποστηρίζει συναλλαγές με τις ACID (Atomicity, Consistency, Isolation, Durability) ιδιότητες. Επιπλέον, παρέχει schema evolution, XML import/export, database replication και υποστηρίζει μεγάλες βάσεις δεδομένων.

### 2.3.8 ObjectStore

Το ObjectStore είναι ένα αντικειμενοστρεφές σύστημα διαχείρισης βάσεων δεδομένων που υποστηρίζει τα κλασικά χαρακτηριστικά ενός συστήματος διαχείρισης βάσεων δεδομένων, όπως αποθήκευση, διαχείριση συναλλαγών, κατανεμημένη πρόσβαση στα δεδομένα και σχεσιακά ερωτήματα. Σχεδιάστηκε για να παρέχει μια διασύνδεση στον προγραμματιστή έτσι ώστε να μπορεί να αποθηκεύει τα δεδομένα (persistently allocated data , δηλαδή, δεδομένα που θα υπάρχουν και μετά την εκτέλεση του προγράμματος) και για δεδομένα που δεν επιζούν μετά την εκτέλεση του προγράμματος (transiently allocated data).

## 2.4 Ανάλυση σύγκρισης

Στον πίνακα 1 έχουμε την συγκριτική ανάλυση από την μελέτη που έχουμε κάνει στα δύο APIs και στα έξι αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων που παρουσιάσαμε στο προηγούμενο κεφάλαιο. Στον πίνακα 1, βλέπουμε τα χαρακτηριστικά που κάθε αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων υποστηρίζει από την μελέτη μας. Το σύμβολο “√” χρησιμοποιείται για να επιδείξει ποιο χαρακτηριστικό υποστηρίζει το αντικειμενοστρεφή σύστημα βάσεων δεδομένων και το σύμβολο “?” για να μας δείξει ότι δεν είναι σαφές αν το χαρακτηριστικό υποστηρίζεται. Το κενό κελί δείχνει πως το αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων δεν υποστηρίζει το συγκεκριμένο χαρακτηριστικό.

| Features / OODBMS's | | JPA | JDO | db4o | ObjectDB | Objectivity | EyeDB | Perst | Objestore |
|---|---|---|---|---|---|---|---|---|---|
| Support of ODMG 3.0 | | √ | √ | | √ | ? | √ | ? | |
| Collections | Set | √ | √ | √ | √ | √ | √ | √ | √ |
| | Bag | | | | | | √ | | |
| | List | √ | √ | √ | √ | √ | | √ | √ |
| | Array | √ | √ | √ | √ | | √ | √ | √ |
| | Dictionary | ? | ? | ? | ? | ? | ? | ? | ? |
| Complex Types | | √ | √ | √ | √ | √ | √ | | |
| Multiple Inheritance | | √ | √ | | | | | | |
| Concurrency Control | | √ | √ | √ | | √ | √ | √ | √ |
| Recovery | | √ | √ | √ | √ | √ | √ | √ | |
| Indexing | | | √ | | √ | √ | √ | √ | √ |
| Query Facility | OQL | √ | √ | √ | √ | √ | √ | √ | |
| | LINQ | √ | √ | √ | √ | √ | | √ | |
| | NoSQL | | | | | | | | |
| XML export | Internal | | | | | √ | ? | √ | |
| | Third Party Program | | | √ | | | ? | | |
| Metadata JDBC type | | √ | √ | | √ | | √ | | |
| Platform | Windows | √ | √ | √ | √ | √ | √ | √ | √ |
| | Android | | | √ | | | | √ | |
| | Linux | √ | √ | √ | √ | √ | √ | √ | √ |
| | Mac | √ | √ | √ | √ | √ | | √ | |
| Client-Server Edition | | √ | √ | √ | √ | | √ | | √ |
| Interface | GUI | | | | | √ | | | |
| | DBMS UI | | | | √ | | √ | | |
| | Data Explorer | | | √ | √ | √ | √ | | |

Πίνακας 1: Συγκριτικός Πίνακας ΑΣΔΒΔ

Από το πίνακα καταλήξαμε στο συμπέρασμα ότι η αξιολόγηση των αντικειμενοστρεφών συστημάτων διαχείρισης βάσεων δεδομένων μας δείχνει ότι υποστηρίζουνε τα περισσότερα χαρακτηριστικά, αλλά κανένα όλα από αυτά. Πιο συγκεκριμένα, τα πρότυπα του ODMG 3.0 υποστηρίζονται από τα JPA, JDO, ObjectDB και το EyeDB αλλά η OQL υποστηρίζονται από db4o, Objectivity και την Perst. Οι τύποι συλλογών υποστηρίζονται από τα περισσότερα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων εκτός από το τύπο bag, που υποστηρίζεται μόνο από το EyeDB και για το τύπο dictionary δεν είναι σαφές αν ένα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων το υποστηρίζει. Στα έξι αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων οι σύνθετοι τύποι μπορούν να δημιουργηθούν εκτός από το Perst και το Objectivity/DB. Μόνο τα δύο APIs επιτρέπουν την πολλαπλή κληρονομικότητα όταν όλα τα άλλα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων περιορίζονται στην μονή κληρονομικότητα. Τα παραδοσιακά χαρακτηριστικά των συστημάτων διαχείρισης βάσεων δεδομένων υποστηρίζονται από την πλειοψηφία αυτών. Τα αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων δεν παρέχουν την δυνατότητα ορισμού ερωτημάτων NoSQL. Μόνο το db4o, Objectivity/DB και το Perst έχουν την δυνατότητα να εξάγουν τα αντικείμενα σε δεδομένα XML. Το JDBC type metadata μπορεί να χρησιμοποιηθεί μόνο από τις εφαρμογές που χρησιμοποιούν ως Object – oriented persistence τα δύο APIs ή στο ObjectDB και στο EyeDB. Μόνο το db4o και το Perst, είναι φορητά και στις τέσσερις πλατφόρμες, ενώ τα περισσότερα αντικειμενοστρεφή συστημάτων διαχείρισης βάσεων δεδομένων μπορούν να χρησιμοποιήσουν την three – tier αρχιτεκτονική.

Και τέλος το db4o, ObjectDB, Objectivity/DB και το EyeDB υποστηρίζεται ένα φιλικό περιβάλλον διασύνδεσης ενώ όλα τα άλλα είναι απλά κείμενα.

## Επίλογος

Σε αυτό το κεφάλαιο παρουσιάσαμε μια λίστα από χαρακτηριστικά που ένα αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων πρέπει να υποστηρίζει. Επίσης, αναφέραμε δύο APIs που μπορούν να χρησιμοποιηθούν από αντικειμενοστρεφές εφαρμογές για διατηρησιμότητα των αντικειμένων και έξι γνωστά αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων. Επιπλέον, σχεδιάσαμε μία συγκριτική ανάλυση σε ένα πίνακα όπου επιδεικνύουμε τα χαρακτηριστικά που πρέπει να υποστηρίζει κάθε αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων. Από αυτή την ανάλυση, συμπεράναμε ότι κανένα αντικειμενοστρεφή σύστημα διαχείρισης βάσεων δεδομένων δεν υποστηρίζει όλα τα χαρακτηριστικά που θα έπρεπε. Στα μελλοντικά μας σχέδια είναι ο σχεδιασμός και η υλοποίηση ενός πρωτότυπου, όπου θα αντιστοιχήσει μια αντικειμενοστρεφή βάση δεδομένων σε μια ισοδύναμη ημιδομημένη βάση δεδομένων XML. Στο επόμενο κεφάλαιο θα αναλύσουμε το db4o καθώς θα είναι και το σύστημα που θα χρησιμοποιήσουμε για να υλοποιήσουμε μία αντικειμενοστρεφή βάση δεδομένων. Οι λόγοι που μας έκαναν να το επιλέξουμε θα εξηγηθούν στο επόμενο κεφάλαιο.

## ΚΕΦΑΛΑΙΟ 3

# Ανάλυση του συστήματος db4o της Versant

### Εισαγωγή

Σε αυτό το κεφάλαιο θα αναπτύξουμε τα χαρακτηριστικά του συστήματος db4o, καθώς είναι το σύστημα που επιλέξαμε για να υλοποιήσουμε την βάση δεδομένων μας. Αυτό που μας βοήθησε στην επιλογή του συγκεκριμένου ήταν ότι από τη ανάλυση και σύγκριση που κάναμε στα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων προέκυψε ότι το db4o είναι ένα από τα ποιο χρησιμοποιούμενα αντικειμενοστρεφή συστήματα, καθώς και με τεράστια βιβλιογραφία στο διαδίκτυο, όπως και οδηγών για τη δημιουργία και υλοποίηση μιας βάσης δεδομένων. Επίσης, είναι από τα πιο ευρέως διαδεδομένα συστήματα διαχείρισης βάσεων δεδομένων, καθώς το χρησιμοποιούν πολλές μεγάλες εταιρίες. Ένας ακόμα λόγος είναι ότι πλέον δημιουργείται η ανάγκη υλοποίησης λογισμικού για κινητές συσκευές, όπως τηλέφωνα και tablet, και έτσι αφού το db4o υποστηρίζει τέτοιες πλατφόρμες του δίνει ένα πλεονέκτημα.

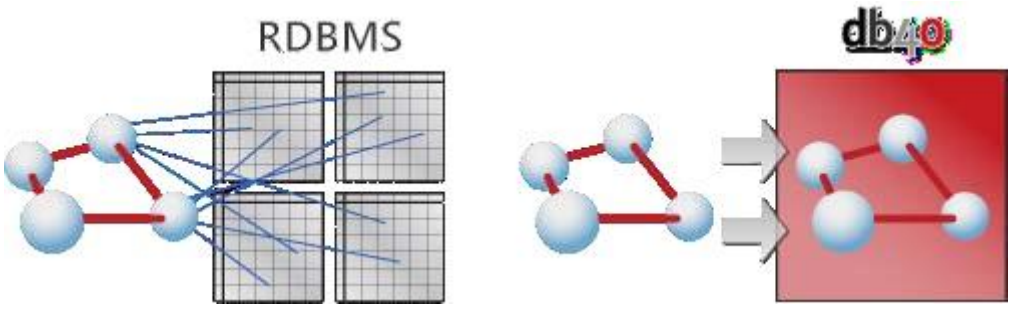
### 3.1 db4o – αντικειμενοστρεφή βάση δεδομένων ανοιχτού κώδικα

Το db4o είναι ένα σύστημα βάσης δεδομένων ανοιχτού κώδικα, το οποίο είναι από τα ποιο ισχυρά και όμως εύκολα στην χρήση συστήματα στον κόσμο. Η δυναμική του προέρχεται από το να αφήνει οσοδήποτε πολυπλοκότητα, ορισμένες εφαρμογές, μοντέλα κλάσεων να μπορούν να αναπαριστούν το σχήμα των δεδομένων σας. Αυτό εξαλείφει την σπατάλη χρόνου και την κουραστική κωδικοποίηση για αντιστοίχηση του μοντέλου και μας επιστρέφει τον χαμένο χρόνο που θα έπαιρνε η επεξεργασία του σε καθαρό χρόνο απόδοσης στην εφαρμογή.

Το db4o, αφήνει την επεξεργασία του προγραμματιστή να επικεντρωθεί στην επιχειρηματική λογική, καθώς αφαιρεί πολυπλοκότητα και επιτυγχάνει πρωτοφανή επίπεδα επιδόσεων. Αυτή η μοναδική σχεδίαση του db4o για εγγενή μηχανισμό αντικειμενοστρεφής βάσης δεδομένων το κάνει να είναι ιδανικό για να είναι ενσωματωμένο σε εξοπλισμό και συσκευές, σε πακέτα λογισμικού που εκτελείτε σε συσκευές, κινητά ή tablet, ή σε συστήματα πραγματικού χρόνου – με λίγα λόγια : σε όλα τα Java και .NET περιβάλλοντα που δεν χρειάζεται διαχείριση της βάσης δεδομένων.

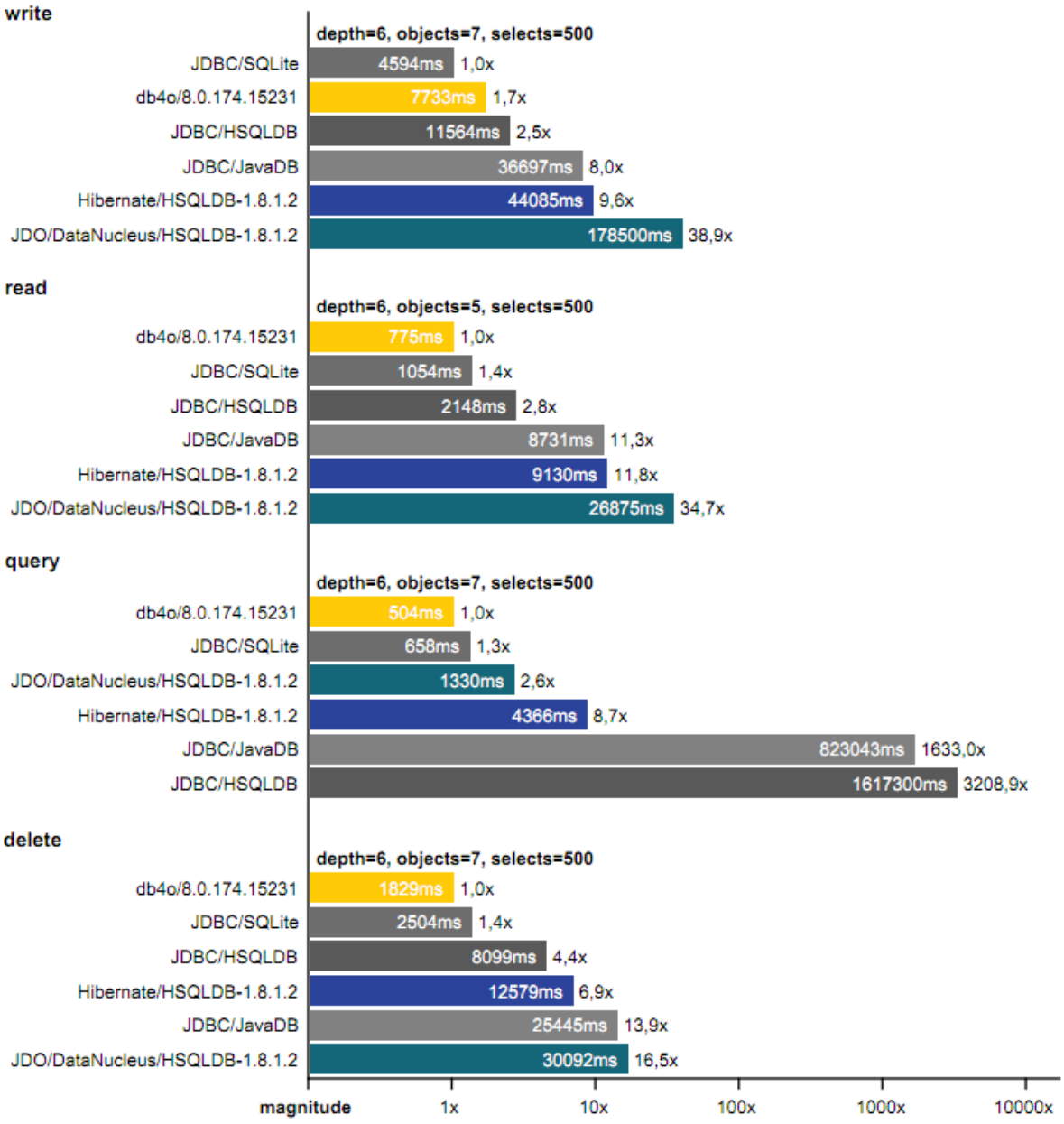
### 3.2 Σχεσιακές βάσεις, αντικειμενοσχεσιακή αντιστοίχηση, και db4o

Όλοι οι προγραμματιστές αντικειμενοστρεφών συστημάτων έχουν γνώση από την δυσκολία τις μετάβασης από τον αντικειμενοστρεφή τρόπο σκέψης στον σχεσιακό για την αποθήκευση των δεδομένων τους. Μέχρι στιγμής, ήταν αναγκασμένη να διαλέγουν μεταξύ ταχύτητας και αντικειμενοστρέφιας : η SQL είναι γρήγορη για απλά πληροφοριακά μοντέλα, αλλά κοπιαστικές και αργές όσο αυξάνεται η πολυπλοκότητα του σχήματος, απαιτώντας μάλιστα και επιπλέον κώδικα. Η αντικειμοσχεσιακές αντιστοιχίσεις προσφέρουν μια κατάλληλη γέφυρα για να αντιμετωπιστεί η πολυπλοκότητα του μοντέλου, αλλά υπάρχει μια υποβάθμιση στην απόδοση και μετακινεί τον προγραμματιστή από την επιχειρηματική λογική στο να κάνει τη λογική μετάφραση και τους περιορισμούς της.




Εικόνα 1

Το db4o εξαλείφει την αντίστροφη σχέση μεταξύ αντικειμενοστρέφιας και απόδοσης, επιτρέπει να αποθηκεύσεις τα πιο σύνθετα αντικείμενα με ευκολία, επιτυγχάνοντας το υψηλότερο επίπεδο απόδοσης. Γνωστά μετροπρογράμματα για βάσεις δεδομένων ανοιχτού κώδικα, μας δείχνουν πως το db4o 8.0 είναι έως και 100 φορές γρηγορότερο από άλλα για μη-τετριμμένες περιπτώσεις χρήσης.




Εικόνα 2 Μετρήσεις μετροπρογραμμάτων

### 3.3 Η εφαρμογή, χαρακτηριστικά και πλεονεκτήματα

Όταν οι σχεσιακές βάσης δεδομένων υπολείπονται της μηδενικής διαχείρισης, του μικρού αποτυπώματος, του ομαλού συγχρονισμού και του απρόσκοπτου refactoring, το db4o δίνει την απάντηση. Εγγενής στη Java και στην .NET, μία βιβλιοθήκη του db4o (.jar/.dll) ενσωματώνεται εύκολα στην εφαρμογή και εκτελεί εφαρμογές υψηλής αξιοπιστίας και κλιμακωτές λειτουργίες αποθήκευσης με μόνο μια γραμμή κώδικα, όσο σύνθετα και αν είναι τα αντικείμενα.

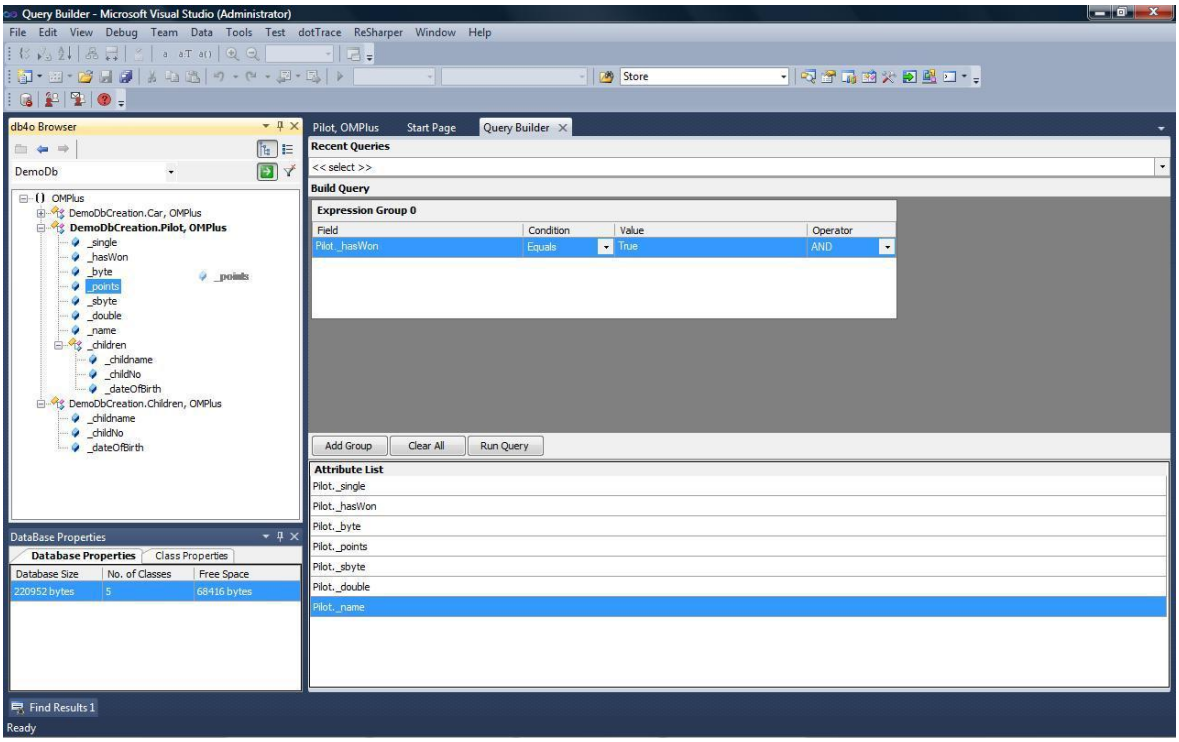
Το db4o προσφέρει τα ακόλουθα πλεονεκτήματα :

- Μένει επικεντρωμένο στην επιχειρηματική λογική, αφού κάνει τα εγγενή δεδομένα διατηρήσιμα, χωρίς περιορισμούς να επιβάλλονται από μια διαδικασία μετατροπής σε πίνακες ή σε OR mapper.
- Απλοποιεί τη διαδικασία και αυξάνει την ποιότητα αποκόπτοντας το μέγεθος των οργανισμών και των ειδικών που πρέπει να εμπλακούν στη διατήρηση των δεδομένων, αφού δεν χρειάζεται κάποιον ειδικό σε DBA, SQL ή σε OR mapping και εξαλείφει το συσχετιζόμενο συντονισμό και ενοποίηση μεταξύ των ομάδων.
- Μπορούμε να σχεδιάσουμε και να παραμετροποιήσουμε εύκολα τα εναύσματα (triggers) δυναμικά, βασισμένοι στην λογική της εφαρμογής χρησιμοποιώντας τον μηχανισμό callback του db4o και τους τυπικούς ακροατές διασύνδεσης της γλώσσας, αντί του άκαμπτου server–based trigger που χρησιμοποιείτε από τις σχεσιακές βάσεις δεδομένων.
- Χρησιμοποιήστε Transparent Persistence (TP) για να απλουσιοποιήσετε σημαντικά των χειρισμό deep object graph σε εφαρμογές που βασίζονται στο db4o παρακολουθώντας τις αλλαγές και τις αποθηκεύσεις όλων τον τροποποιημένων αντικειμένων αυτόματα στο commit time.
- Μειώστε το χρόνο διάθεσης της αγοράς με τα ακόλουθα μηχανικά χαρακτηριστικά :
    - Εξαλείψτε τα εργαλεία και τη συγγραφή κώδικα που είναι απαραίτητα για το αντικειμενοσχεσιακό mapping που έχει αποδειχθεί ότι οδηγεί στην πολυπλοκότητα του κώδικα και κατανάλωση των πόρων, ενώ αναστέλλουν την απόδοση και την συντήρηση του συστήματος. Με το db4o, οι χρήστες αποκτούν μέχρι και 40% περισσότερο χρόνο και μειώνει το κόστος για την ανάπτυξη λογισμικού που σχετίζεται με διατήρηση δεδομένων.
    - Δημιουργήστε εφαρμογές με άψογες ενσωματωμένες συναλλαγές στην βάση δεδομένων που δεν χρειάζεται διαχείριση της παραγωγής και που είναι αξιόπιστο και πολύ πιο γρήγορο από τους συμβατικούς μηχανισμούς των βάσεων δεδομένων.
    - Επωφεληθείτε από τα αντικειμενοστρεφή παραδείγματα στην Java και στην .NET, επωφεληθείτε από τα αντικειμενοστρεφή περιβάλλοντα προγραμματισμού και την ικανότητα να αναπτύξετε πιο σύνθετα, φυσικά και πλούσια σε χαρακτηριστικά αντικειμενοστρεφή μοντέλα χωρίς να οδηγούμαστε στην αύξηση του κόστους και της κατανάλωσης πόρων.
    - Αλλαγές, refactor, επαναχρησιμοποίηση τμημάτων λογισμικού με τη δυνατότητα να προσθέσουμε νέες λειτουργίες χωρίς να πειράξουμε τα παλιά αντικείμενα ή να υποστεί μετατροπή ή αναβάθμιση των σχετικών δαπανών – που παρέχει την σβελτάδα που απαιτείτε για να μείνει μπροστά στον ανταγωνισμό.

Το db4o οδηγείτε από τη μεγαλύτερη κοινότητα του είδους του με περίπου 100000 δηλωμένους προγραμματιστές σε Java και .NET και αυξάνεται. Το πρόγραμμα

έχει κατεβαστεί περίπου 3 εκατομμύρια φορές και έχει ενσωματωθεί με επιτυχία στις μεταφορές, στα δίκτυα, στις φυσικές επιστήμες, στο εμπόριο, στη βιομηχανία και πολλά άλλα. Οι χρήστες και οι πελάτες του db4o προέρχονται από 170 διαφορετικές χώρες και ποικίλουν από παγκόσμιες δυνάμεις, όπως Boeing, Cisco, Indra, Lockheed Martin, Nothrop Grumman, Siemens και Sony μέχρι και σε εταιρίες που τώρα ξεκινάνε.

Φτιαγμένο σε επόμενη γενιά τεχνολογίας αντικειμενοστρεφών βάσεων δεδομένων, το db4o παρέχει ένα ευρύ φάσμα των μοναδικών του δυνατοτήτων η οποία είναι απαράμιλλη στον κόσμο των βάσεων δεδομένων, δηλαδή, επωφελείται από τα αντικειμενοστρεφή περιβάλλοντα προγραμματισμού, αντικειμενοστρεφή peer–to–peer Replication (dRS), αντικειμενοστρεφή ερωτήματα (LINQ, Native Queries, QBE, S.O.D.A) και τεχνολογία object management enterprise για την περιήγηση και τη συντήρηση της βάσης δεδομένων μας.




Εικόνα 2 Object Manager Enterprise

Πάνω από όλα, ας μην ξεχνάμε πως το db4o είναι πολύ εύκολο στην χρήση του και στην υλοποίησή του. Αυτή η ισχυρή μηχανή της βάσης δεδομένων επιτρέπει στους χρήστες να αποθηκεύουν αντικείμενα με μόνο μια γραμμή κώδικα μειώνοντας το κόστος υλοποίησης και σε χρόνο αλλά και σε χρήμα. Αυτά τα πλεονεκτήματα μεγεθύνονται όταν έρχεται η ώρα για να εξελίξουμε λογισμικό με σκοπό να διατηρήσουμε, βελτιώσουμε, ενσωματώσουμε νέα χαρακτηριστικά ή να επαναχρησιμοποιήσουμε συστατικά λογισμικού. Το db4o κάνει αυτόματα refactor μέσω του προγραμματιστικού περιβάλλοντος, επειδή όλα τα μη εγγενή APIs και συμβολοσειρές εξαλείφονται. Ως εκ τούτου, μπορούμε να προσαρμόσουμε δυναμικά το σχήμα από έκδοση σε έκδοση, είτε στην ανάπτυξη, καθώς το db4o τακτοποιεί μόνο του τις αλλαγές στο σχήμα. Όταν ενσωματώνουμε κάποια

ενημέρωση στην εφαρμογή μας αυτό μεταφράζεται σε καμία μεταφορά ή μετατροπή των δεδομένων μας.

## 3.4 Χαρακτηριστικά και πλεονεκτήματα

Στο παρακάτω πίνακα βλέπουμε μερικά από τα βασικά χαρακτηριστικά και πλεονεκτήματα που έχει το db4o και αξίζουν να μελετηθούν. Παρακάτω θα αναπτυχθούν ένα προς ένα αυτά τα σημεία.

| Key Features | Key Benefits |
|---|---|
| **The One-Line-of-Code-Database**<br>• One line of code stores any object<br>• Class model = object schema<br>• Smooth production process<br><br>**Embeddable**<br>• Zero administration<br>• Automatic schema versioning<br>• ~1 MB footprint<br><br>**Multiple platform support**<br>• Native to Java and .NET<br>• Embedded CPU, mobile device, desktop, and server platforms<br>• IDE friendly<br>• 64bit compatible<br><br>**Brings more OO to the database**<br>• Object-oriented replication (dRS) to/from db4o, relational DBs and VOD<br>• LINQ support<br>• Native Queries<br>• Object Manager Enterprise - exploration and querying tool | **40% faster to market** with your application<br><br>**Full ACID transactional capabilities**<br><br>**Slashes 40% of cost** to develop persistence<br><br>Build **lean and pure** object-oriented software<br><br>Deployable in large volumes without local administration<br><br>Build asynchronously-distributed, fully synchronized data architectures<br><br>Fewer errors, better maintainability and software longevity |

Πίνακας 2 : χαρακτηριστικά και πλεονεκτήματα

### 3.4.1 Βάση δεδομένων με μια γραμμή κώδικα – The One – Line – of – Code Database

Ο τρόπος ανάπτυξης μιας βάσης δεδομένων είναι μια πολύ απλή διαδικασία. Εισάγουμε την db4o βιβλιοθήκη (.jar/.dl) στο προγραμματιστικό μας περιβάλλον, έπειτα ανοίγουμε μια βάση δεδομένων και αποθηκεύουμε οποιοδήποτε αντικείμενο – όσο σύνθετο και αν είναι – με μόνο μια γραμμή κώδικα. Αυτός ο ασύγκριτος τρόπος εύκολης ανάπτυξης μειώνει δραστικά τον χρόνο προγραμματισμού.

Εξαλείφει όλη την εργασία του σχεδιασμού, της υλοποίησης και της συντήρησης του σχήματος της βάσης δεδομένων, επειδή το μοντέλο κλάσεων είναι το σχήμα της βάσης. Εξαλείφει την ανάγκη διαχείρισης της επιβάρυνσης που σχετίζεται με την βάση δεδομένων, όπως οι συμβολοσειρές (Strings), XML, ή άλλα μη-εγγενή αρχεία που χρειάζονται προ (POST) – ή μετά (pre) μεταγλώττισης ή βελτιωτικά και κατά συνέπεια, επιβραδύνουν την διαδικασία παραγωγής μας.

Κερδίζουμε πολύ χρόνο στο να γράψουμε το πρόγραμμά μας. Κερδίζουμε όμως περισσότερο χρόνο κάθε φορά που θα χρειαστεί να αλλάξουμε το πρόγραμμα, π.χ. refactoring code, εισαγωγή νέων χαρακτηριστικών, ή επαναχρησιμοποίηση παλιών συστατικών λογισμικού. Το να αλλάξουμε το μοντέλο αντικειμένων, για παράδειγμα, δεν είναι μόνο εξαιρετικά διαφανής (transparent), αλλά και ηλίθιο επειδή η εγγενής και η μη παρεμβατική φύση του db4o αφήνει το προγραμματιστικό περιβάλλον να κάνει τη δουλεία για εσάς! Δεν χρειάζεστε debuggers, καμία διαδικασία build και δεν χρειάζεται να ανησυχείτε για υπάρχουσες υλοποιήσεις, επειδή το db4o τακτοποιεί κάθε μετατροπή στα αντικείμενα για ολόκληρη τη βάση δεδομένων. Το να αλλάζεις το πρόγραμμα παύει να είναι εφιάλτης και γίνεται πιο απολαυστικό, κάνοντας μας πιο παραγωγικούς. Στην ουσία, το db4o καθιστά εύκολο την διατήρηση των αντικειμένων, όπως είναι η χρήση σειριοποιησιμότητας (Serialization), αλλά επίσης σου δίνει το εύρος της λειτουργικότητας μιας βάσης δεδομένων, όπως την αναζήτηση, τις συναλλαγές και κυρίως στο ότι επιτρέπει την αλλαγή του μοντέλου αντικειμένων χωρίς να πρέπει να διακόψουμε την εφαρμογή.

### 3.4.2 Η Ενσωματωμένη Βάση Δεδομένων – Embeddable Database

Το db4o είναι σχεδιασμένο να είναι ενσωματωμένο σε πελάτες (clients) ή άλλα συστατικά λογισμικού αφανή στον τελικό χρήστη. Έτσι, δεν χρειάζεται κάποιο μηχανισμό εγκατάστασης, αλλά το βρίσκουμε σαν μια εύκολα προσαρτήσιμη βιβλιοθήκη με μικρό αποτύπωμα της τάξης του 1MB. Επειδή το db4o τρέχει στην ίδια διαδικασία με την εφαρμογή μας, έχουμε πλήρη πρόσβαση στη διαχείριση της μνήμης και μπορούμε να κάνουμε μετρήσεις ταχύτητας και εντοπισμό σφαλμάτων σε όλο το σύστημα. Εάν η εφαρμογή μας λειτουργεί, λειτουργεί και η βάση δεδομένων μας. Δεν υπάρχουν εξαιρέσεις.

Πιο συγκεκριμένα, το db4o είναι εξαιρετικά ευέλικτο όσον αφορά την ενημέρωση μιας υπάρχουσας βάσης με διαφορετικό μοντέλο αντικειμένων. Θεωρεί πάντα ότι δεν υπάρχει διαχείριση και ως εκ τούτου επιτρέπει στην εφαρμογή να κάνει την αρμονική αλλαγή από το παλιό στο νέο μοντέλο. Σε αντίθεση με οποιαδήποτε άλλη βάση δεδομένων, είτε πρόκειται για σχεσιακές, είτε για μη εγγενής αντικειμενοστρεφή βάσεις δεδομένων, το db4o δεν χρειάζεται καμία μορφή διαχείρισης ενημέρωσης του μοντέλου αντικειμένων.

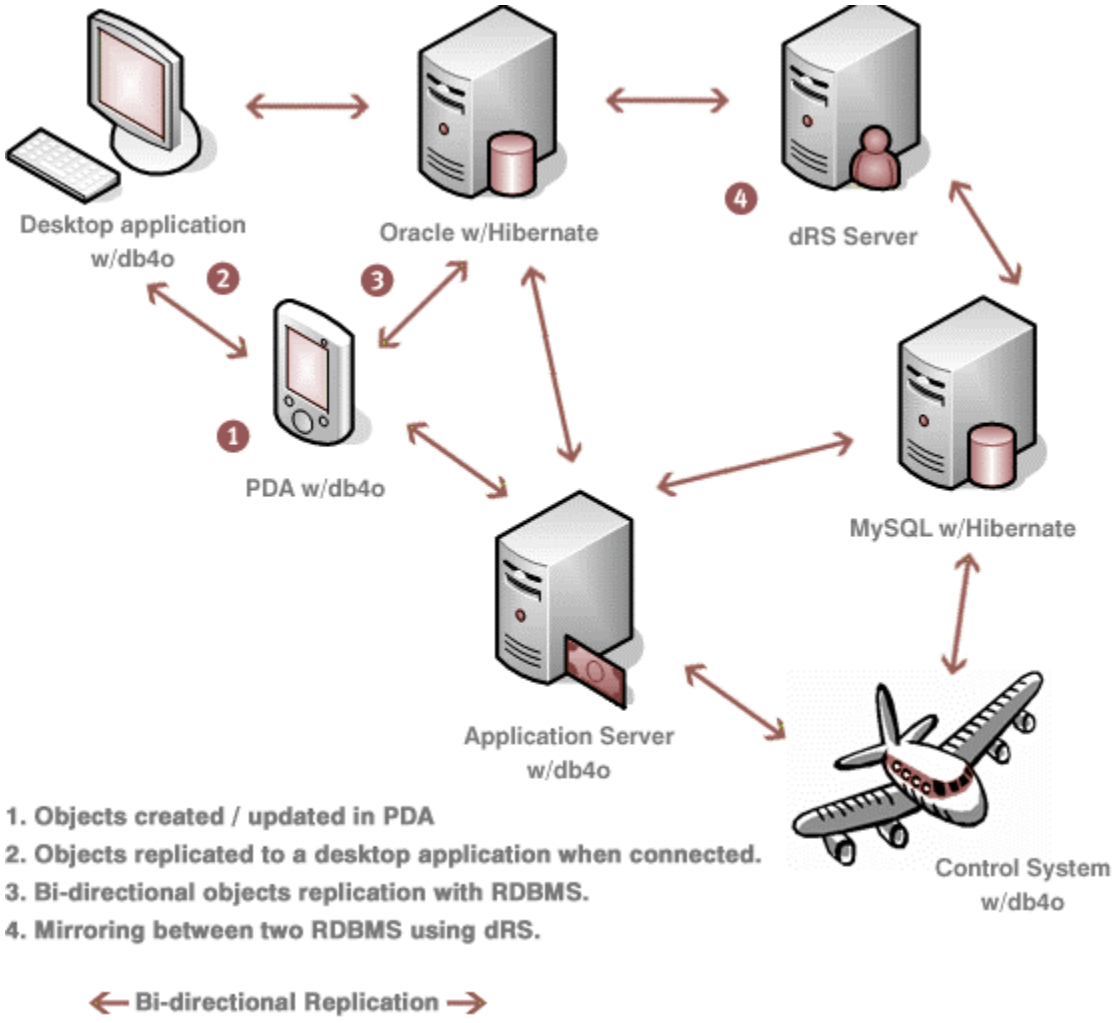
### 3.4.3 Μεταφερσιμότητα – Multiple platform support

Πολύ λίγα ενσωματωμένα συστήματα βάσεων δεδομένων λειτουργούν εγγενώς σε τόσες πολλές αντικειμενοστρεφή πλατφόρμες. Το db4o υποστηρίζει Java JDK 5+ και λειτουργεί σε J2EE και J2SE. Επίσης, λειτουργεί και σε J2ME διαλέκτους που υποστηρίζουν reflection (όπως, CDC profile). Επιπλέον, υποστηρίζει πλήρως το Android OS και είναι διαθέσιμο σαν ένα πακέτο OSGi.Τρέχει σε όλες τις πλατφόρμες που υποστηρίζουν .NET συμπεριλαμβανομένων των .NET και Compact Framework και γλώσσες προγραμματισμού, όπως C#, VB.NET, ASP.NET, Boo και C++. Επιπλέον υποστηρίζει Silverlight που το κάνει ιδανικό για ανάπτυξη λογισμικού για windows phone 7. Παρέχει επίσης έναν πάροχο LINQ που κάνει τα ερωτήματα πιο εγγενή και εύκολα στην .NET πλατφόρμα.

### 3.4.4 Κατανεμημένη Αρχιτεκτονική – Distributed Data Architectures

Ενώ οι αρχικές εφαρμογές ήτανε σε standalone client, όπως ένα Smartphone ή ένα αυτοκίνητο, οι περισσότεροι πελάτες πλέον είναι εν μέρει συνδεμένοι με ενδιάμεσο λογισμικό, servers, ή άλλες συσκευές.

Με στόχο να αυξηθεί η συνδεσιμότητα το db4o, παρέχεται ένα μοναδικό σύστημα Replication (dRS) που επιτρέπει εύκολο συγχρονισμό δεδομένων μεταξύ βάσεων δεδομένων του db4o, VOD (Versant Object Database) και σχεσιακές βάσεις δεδομένων, όπως Oracle ή MySQL. Οι σχεσιακές βάσεις πιάνονται από το OR mapper του Hibernate. Η υλοποίηση είναι και πάλι όσο εύκολη με πριν : συνδέουμε τις βάσεις να ψάξουν και να ενημερώσουν αντικείμενα και να συγχρονιστούν με μια γραμμή κώδικα μόνο. Εν κατακλείδι, στο db4o, οι διαφορετικοί τρόποι εκτέλεσης και η μοναδική λειτουργία του της αντικειμενοστρεφής αναπαραγωγής του (Replication) επιτρέπουν τις πολύ ισχυρές και αποτελεσματικές κατανεμημένες αρχιτεκτονικές δεδομένων που χρειάζονται για service–oriented computing και CEP (complex event processing).




Εικόνα 3 Distributed Data Architecture

### 3.4.5 Ερωτήματα στο db4o – Queries in db4o

Από την έκδοση 5 του db4o, αυτό είναι το πρώτο σύστημα που υλοποιεί Native Queries (NQ), οδηγώντας τη τάση της βιομηχανίας να προσφέρει αναζήτηση στην βάση δεδομένων με τη σημασιολογία των αντικειμενοστρεφών γλωσσών προγραμματισμού που επικυρώθηκε από το LINQ της Microsoft (.NET Language Integrated Queries).

Αντί να χρησιμοποιήσει APIs που βασίζονται στις συμβολοσειρές (Strings), όπως SQL, OQL, JDOQL, JPAQL και S.O.D.A, τα NQ επιτρέπουν στους προγραμματιστές απλά να χρησιμοποιήσουνε την ίδια την γλώσσα, όπως Java, C#, VB.NET, για να έχουν πρόσβαση στη βάση δεδομένων και επωφελούνται έτσι από τον έλεγχο τύπου στην μεταγλώττιση, την πλήρη εκφραστική δύναμη των αντικειμενοστρεφών γλωσσών και της μεγάλης ευκολίας του εξελιγμένου προγραμματιστικού περιβάλλοντος.

### 3.4.6 Ενσωμάτωση – Integration, άλλα APIs, Object Manager Και εκθέσεις– reporting

Η εξαγωγή των δεδομένων σε XML επιτυγχάνεται εύκολα πλέον με οποιαδήποτε βιβλιοθήκη που είναι ικανή να γράψει τα δεδομένα σε XML, όπως το XStream. Επιτρέπει την ενσωμάτωση στο db4o μέσω της αρχιτεκτονικής με μηνύματα.

Το db4o δεν παρέχει κάποια διασύνδεση για SQL, επειδή οι προγραμματιστές δεν χρειάζεται να έχουν απευθείας πρόσβαση στα δεδομένα με κάποιο μη εγγενή API. Ωστόσο, για λόγους συμβατότητας, οι προγραμματιστές αντιγράφουν τα αντικείμενά τους μέσω του dRS σε οποιαδήποτε σχεσιακή βάση δεδομένων για περαιτέρω επεξεργασία.

Το Object Manager Enterprise (OME) είναι ένα δωρεάν εργαλείο εξερεύνησης βάσεων δεδομένων που μπορούμε να ενσωματώσουμε στο Visual Studio ή στο Eclipse. Αυτό είναι σχεδιασμένο για περιήγηση σε βάσεις του db4o και συντήρηση και περιέχει :

- Προβολή Κλάσεων : επίπεδη και ιεραρχική δομή των διατηρημένων κλάσεων
- Προβολή λεπτομερειών κλάσης : πεδία, υποκλάσεις και διασυνδέσεις και στατιστικά (πλήθος εγγραφών στη βάση δεδομένων)
- Προβολή συγκεκριμένου αντικείμενου δενδρικά
- Βοηθός κατασκευής ερωτημάτων (Query Builder) : μας επιτρέπει να φτιάξουμε σύνθετα ερωτήματα χρησιμοποιώντας πεδία σε διαφορετικά επίπεδα της ιεραρχίας συνδυασμένα με λογικούς τελεστές.
- Προβολή αποτελεσμάτων από τα ερωτήματα
- Προβολή ιστορικού : πρόσφατα ερωτήματα ή αντικείμενα
- Προβολή αγαπημένων : καθορισμένα από το χρήστη ερωτήματα ή αντικείμενα

## 3.5 Μειονεκτήματα του db4o

Κάθε σύστημα όπως είναι φυσικό έχει και τα μειονεκτήματα του. Σε αυτή τη παράγραφο θα περιγράψουμε μερικά από αυτά έτσι ώστε να έχουμε μια καλύτερη εικόνα του db4o. Ένα σημαντικό μειονέκτημα του db4o είναι η έλλειψη διαχείρισης των ταυτοτήτων των αντικειμένων. Σε περίπτωση που δουλεύουμε μόνο σε μια βάση δεδομένων και δεν χρειάζεται να στείλουμε αντικείμενα σε κάποια άλλη διεργασία ή να ξανά ανοίξουμε τη βάση δεδομένων αυτό δεν είναι πρόβλημα. Αλλιώς, αυτή η προσέγγιση είναι δύσκολη. Άλλο ένα μειονέκτημα του db4o είναι ότι σε πολύ μεγάλες βάσεις δεν μπορούμε να βρούμε αποδόσεις που προσφέρουν τα σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων για την ευρετηριοποίηση (indexing). Σε κάποιες περιπτώσεις, εκατομμύρια αντικείμενα, η ευρετηριοποίηση γίνεται σχεδόν ανέφικτη στο db4o. Ακόμα ένα είναι πως η πρόσβαση στα δεδομένα γίνεται μέσω ενός νήματος. Δεν υπάρχει η δυνατότητα να κάνουμε αναζητήσεις ή και ενημερώσεις παράλληλα. Τα πολυεπεξεργαστικά συστήματα δεν θα μας προσφέρουν κανένα πλεονέκτημα. Οπότε τα χρονοβόρα ερωτήματα θα είναι το μεγάλο μας πρόβλημα γιατί τα απλά και γρήγορα ερωτήματα θα πρέπει να περιμένουν για την ολοκλήρωση τους. Επίσης, ένα ακόμα θέμα που προκύπτει στο db4o είναι η μεγάλη κατανάλωση πόρων στο σκληρό δίσκο. Διάφορες μετρήσεις που έχουν γίνει μας δείχνουν πως το db4o χρησιμοποιεί παραπάνω πόρους από ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων. Επιπλέον διαπιστώνετε πως τα exceptions του κώδικα του db4o δεν επιλύονται και έτσι αποτρέπεται από το χρήστη να καταλάβει το πρόβλημα που έχει δημιουργηθεί. Σημαντικό είναι επίσης να αναφέρουμε πως υπάρχει ένα πρόβλημα με την αναφορική ακεραιότητα, καθώς αν θέλουμε να διαγράψουμε όλα τα παιδιά από ένα γονέα σε περίπτωση διαγραφής του γονέα θα πρέπει να τοποθετήσουμε των κώδικα στη κλάση του γονέα. Σε πολλά σενάρια αυτό χαλάει την αντικειμενοστρεφή προσέγγιση του κώδικα. Η επικοινωνία μεταξύ τερματικού και διακομιστή είναι πολύ αργή και ανεπαρκής επίσης και αυτό καθιστά αδύνατη τη χρήση client/server σε μια βάση δεδομένων μεσαίου μεγέθους. Επιπλέον, πρέπει να αναφερθεί ότι για BLOB δεδομένα δεν υποστηρίζονται τα ACID, καθώς αυτά δεν αποθηκεύονται στη βάση δεδομένων και δεν είναι ούτε διαθέσιμα για συναλλαγές. Είναι απλά αρχεία.

## Επίλογος

Σε αυτό το κεφάλαιο μελετήσαμε τα χαρακτηριστικά του αντικειμενοστρεφή συστήματος διαχείρισης βάσεων δεδομένων, το db4o και αναπτύξαμε μερικά χαρακτηριστικά του που ήτανε και ο λόγος που μας έκαναν να το επιλέξουμε για να κατασκευάσουμε μια αντικειμενοστρεφή βάση δεδομένων για να πειραματιστούμε και να ανακαλύψουμε τις δυνατότητες και τα πλεονεκτήματα που έχει, καθώς και τα μειονεκτήματα του. Επίσης, ένας άλλος λόγος που επιλέχθηκε αυτό το σύστημα για την υλοποίηση της βάσης είναι ότι βρέθηκε αρκετό υλικό για μελέτη καθώς θεωρείτε από τα καλύτερα συστήματα αντικειμενοστρεφών βάσεων

δεδομένων. Οπότε, στο επόμενο κεφάλαιο θα σας παρουσιάσουμε τον τρόπο που θα αναπτύξουμε μια βάση στο db4o με παραδείγματα κώδικα, καθώς και με εκτενή εξήγηση των εντολών που θα χρησιμοποιήσουμε για την δημιουργία της. Αυτό θα γίνει με την χρήση της γλώσσας προγραμματισμού Java. Η υλοποίηση της βάσης δεδομένων έγινε με την χρήση του Netbeans IDE 7 [62], ενός εργαλείου που παρέχεται δωρεάν από το Common Development and Distribution License (CDDL) v1.0 [63] and GNU General Public License (GPL) v2. [64].

# ΚΕΦΑΛΑΙΟ 4

# Υλοποίηση αντικειμενοστρεφής βάσης δεδομένων με χρήση του db4o

## Εισαγωγή

Σε αυτό το κεφάλαιο θα δημιουργήσουμε μια βάση δεδομένων για να δείξουμε την συμπεριφορά του OODBMS και την λειτουργία του, καθώς θα εξηγήσουμε τον τρόπο λειτουργίας των εντολών και τον βιβλιοθηκών που θα χρησιμοποιήσουμε. Σκοπός του κεφαλαίου είναι να κατανοήσουμε την χρήση του db4o και να μπορέσουμε εύκολα και γρήγορα να αναπτύξουμε μια βάση δεδομένων. Το εργαλείο που θα χρησιμοποιήσουμε για την δημιουργία είναι το Netbeans 7.1. Από τα πιο σημαντικά πράγματα που θα δούμε είναι η κληρονομικότητα, οι σχέσεις μεταξύ αντικειμένων, που αυτά είναι ακριβώς όπως και σε μια αντικειμενοστρεφή γλώσσα προγραμματισμού, η αποθήκευση τους και τα ερωτήματα που θα κάνουμε με χρήση QBE και NQ. Στο τέλος, θα δείξουμε και τον τρόπο εξαγωγής μεταδεδομένων από τη βάση μας.

## 4.1 Περιγραφή του API

Το πρώτο που έχουμε να κάνουμε για να ξεκινήσουμε την δημιουργία της βάσης δεδομένων μας είναι να εγκαταστήσουμε τη βιβλιοθήκη db4o-all.jar στο classpath της εφαρμογής. Αν χρησιμοποιούμε κάποιο IDE (integrated development environment), όπως Netbeans, Eclipse, αντιγράφουμε τη βιβλιοθήκη στο /lib/ φάκελο της εφαρμογής μας. Τα πακέτα com.db4o και com.db4o.query είναι τα μόνα που πρέπει να λάβουμε υπόψη μας.

### 4.1.1 com.db4o

Το πακέτο com.db4o παρέχει όλη τη λειτουργικότητα που θα χρειαστείτε χρησιμοποιώντας το db4o. Δύο αντικείμενα που αξίζει να αναφερθούν είναι οι διασυνδέσεις (interfaces) com.db4o.Db4oEmbedded και com.db4o.ObjectContainer.

Ο μηχανισμός του com.db4o.Db4oEmbedded είναι το σημείο εκκίνησης μας. Στατικές μέθοδοι στην κλάση μας επιτρέπουν να ανοίξουμε ένα αρχείο βάσης δεδομένων και να δημιουργήσουμε μία αρχική διαμόρφωση. Για client/server περιβάλλοντα θα χρειαστεί να χρησιμοποιήσουμε το μηχανισμό com.db4o.cs.Db4oClientServer για να ξεκινήσει ο server ή για να συνδεθούμε

στον server, αλλά αυτό θα το συζητήσουμε αργότερα. Η πιο σημαντική διασύνδεση που θα χρησιμοποιούμε είναι η com.db4o.ObjectContainer, αυτή είναι η db4o βάση δεδομένων μας. Ένα ObjectContainer μπορεί να είναι είτε μια βάση δεδομένων για έναν χρήστη ή μια σύνδεση client στον db4o server. Κάθε ObjectContainer διαχειρίζεται μία συναλλαγή. Όταν ανοίγουμε ένα ObjectContainer έχουμε μία συναλλαγή, όταν κάνουμε commit() ή rollback(), ξεκινάει αυτόματα η επόμενη συναλλαγή. Κάθε ObjectContainer κρατάει τις αναφορές του στα αποθηκευμένα ή αρχικοποιημένα αντικείμενα του. Κάνοντας αυτό, διαχειρίζεται τις ταυτότητες των αντικειμένων και μπορεί να επιτύχει υψηλό επίπεδο απόδοσης. Τα ObjectContainer προορίζονται να είναι ανοιχτά για όσο διάστημα εργάζεστε εις βάρος τους. Όταν το κλείνετε, close(), όλες οι αναφορές που βρίσκονται στην μνήμη RAM θα αποβάλλονται.

### 4.1.2 com.db4o.ext

Η διασύνδεση του db4o παρέχετε με δύο τρόπους και αυτό γιατί είναι πιο εύκολο να ξεκινήσεις επειδή το com.db4o δίνει έμφαση στις βασικές μεθόδους, είναι εύκολο άλλα προγράμματα να χρησιμοποιήσουν το db4o και είναι ένα παράδειγμα πόσο ελαφριά έκδοση θα μπορούσε να είναι. Κάθε αντικείμενο com.db4o.ObjectContainer είναι επίσης και com.db4o.ext.ObjectContainer. Μπορούμε να το κάνουμε cast σε ExtObjectContainer ή να χρησιμοποιήσουμε τις μεθόδους του για να πάρουμε τα πιο σύνθετα χαρακτηριστικά του.

### 4.1.3 com.db4o.config

Αυτό το πακέτο περιέχει τύπους και κλάσεις που χρειάζονται για να διαμορφώσουμε το db4o. Τα αντικείμενα και οι διασυνδέσεις θα συζητηθούν αργότερα.

### 4.1.4 com.db4o.query

Αυτό, περιέχει την κλάση Predicate που μας επιτρέπει να δημιουργούμε NQ. Η διασύνδεση για NQ είναι η πρωταρχική του db4o και πρέπει να προτιμάτε από τη χρήση του SODA API.

## 4.2 Δημιουργία οντοτήτων

Ας ξεκινήσουμε λοιπόν να εξηγήσουμε την δημιουργία των οντοτήτων στο db4o. Στη συνέχεια θα δείξουμε πώς γίνεται η αποθήκευση, η ανάκτηση, η ενημέρωση και η διαγραφή των δεδομένων.

Θα περιγράψουμε τώρα την δημιουργία μιας οντότητας :

```
1 public class Person {
2
3   private String name;
4   private String surname;
5   private String country_of_origin;
6
7    public Person() {
8    }
9
10    public Person(String name, String surname,String country_of_origin)
11 {
12        this.name = name;
13        this.surname = surname;
14        this.country_of_origin = country_of_origin;
15    }
16
17    public String getCountry_of_origin() {
18        return country_of_origin;
19    }
20
21    public void setCountry_of_origin(String country_of_origin) {
22        this.country_of_origin = country_of_origin;
23    }
24
25    public String getName() {
26        return name;
27    }
28
29    public void setName(String name) {
30        this.name = name;
31    }
32
33    public String getSurname() {
34        return surname;
35    }
36
37    public void setSurname(String surname) {
38        this.surname = surname;
39    }
40
41    @Override
42    public String toString() {
43        return "Person{" + "name=" + name + ", surname=" + surname
44                + ", country_of_origin=" + country_of_origin + '}';
45    }
46
47 }
```

Όπως βλέπουμε, η δημιουργία μιας οντότητας γίνεται πολύ εύκολα καθώς είναι ακριβώς η ίδια διαδικασία με την δημιουργία οποιαδήποτε κλάσης στην Java. Δηλαδή, φτιάχνουμε κατασκευαστές, μεθόδους get – set, toString για κάθε χαρακτηριστικό του αντικειμένου. Η κλάση Person, έχει τρία χαρακτηριστικά :

Όνομα, Επώνυμο και χώρα καταγωγής. Περιγράφει ένα άτομο και θα το χρησιμοποιήσουμε για να προβάλουμε την κληρονομικότητα στο db4o για την υλοποίηση που έχουμε πραγματοποιήσει. Ας δούμε όμως πρώτα πως θα χρησιμοποιούμε το db4o.

### 4.2.1 Πρόσβαση στη βάση δεδομένων

Για να έχουμε πρόσβαση στη βάση δεδομένων του db4o ή για να δημιουργήσουμε μια καινούργια πρέπει να καλέσουμε την μέθοδο Db4oEmbedded.openFile() και να του παρέχουμε ένα Db4oEmbedded.newConfiguration() ως ένα πρότυπο ρυθμίσεων παραμέτρων και τη διαδρομή προς το αρχείο δεδομένων σαν δεύτερη παράμετρο για να δημιουργήσουμε ένα στιγμιότυπο ObjectContainer. Αυτό το στιγμιότυπο αντιπροσωπεύει την βάση δεδομένων και θα είναι η πρωταρχική διασύνδεση για το db4o. Όταν θα κλείσουμε το ObjectContainer με την μέθοδο close(), θα κλείσει το αρχείο της βάσης δεδομένων και θα αποδεσμεύσει όλους τους πόρους που σχετίζονται με αυτό.

```
// accessDb4o
ObjectContainer db = Db4oEmbedded.openFile(Db4oEmbedded
        .newConfiguration(), DB4OFILENAME);
try {
    // do something with db4o
} finally {
    db.close();
}
```

Το DB4OFILENAME είναι μια συμβολοσειρά που αντιπροσωπεύει οποιοδήποτε όνομα του αρχείο που θα επιλέξουμε. Εάν το όνομα αρχείου υπάρχει, τότε θα το ανοίξει σαν αρχείο βάσης δεδομένων του db4o, αλλιώς θα δημιουργήσει μία καινούργια βάση δεδομένων.

### 4.2.2 Αποθήκευση δεδομένων

Για να αποθηκεύσουμε ένα αντικείμενο απλά καλούμε την μέθοδο store() στην βάση δεδομένων μας περνώντας σαν παράμετρο κάποιο αντικείμενο.

```
// storeFirstPerson
Person person1 = new Person("name", "surname", "country_of_origin");
db.store(person1);
System.out.println("Stored " + person1);
```

Ομοίως, για κάθε αντικείμενο τύπου Person που θέλουμε να δημιουργήσουμε και να το αποθηκεύσουμε στη βάση δεδομένων.

### 4.2.3 Ανάκτηση δεδομένων

Ο πιο απλός τρόπος για να δούμε τα περιεχόμενα της βάσης δεδομένων μας είναι το Object Manager Enterprise, που έχουμε αναφερθεί προηγουμένως. Τώρα ας δούμε πώς να χτίζουμε ερωτήματα στο db4o. Το db4o υποστηρίζει αρκετά συστήματα ερωτημάτων όπως, Query by Example (QBE), Native Queries (NQ) και το SODA query API (SODA). Σαν πρώτο παράδειγμα θα δείξουμε την χρήση του QBE, έτσι ώστε να εξοικειωθούμε με την ανάκτηση των δεδομένων και έπειτα θα δούμε πως γίνεται με NQ.

Όταν χρησιμοποιούμε QBE, δημιουργούμε ένα πρωτότυπο αντικείμενο στο db4o για να το χρησιμοποιήσουμε σαν παράδειγμα σαν αυτό το αντικείμενο που θέλουμε να ανακτήσουμε. Το db4o θα ανακτήσει όλα τα αντικείμενα του δεδομένου τύπου τα οποία εμπεριέχουνε τις ίδιες τιμές στα πεδία όπως το παράδειγμα. Τα αποτελέσματα θα επιστρέψουν σε ένα στιγμιότυπο ObjectSet. Επίσης, θα χρησιμοποιήσουμε μια βολική μέθοδο την listResult() για να εμφανίσουμε τα περιεχόμενα του ObjectSet.

```
public static void listResult(List<?> result){
    System.out.println("Results number: " + result.size());
        for (Object o : result) {
            System.out.println(o);
        }
}
```

Για να ανακτήσουμε όλα τα Person από την βάση δεδομένων μας, παρέχουμε ένα κενό πρωτότυπο

```
// retrieveAllPersonQBE
Person proto = new Person(null, null, null);
ObjectSet result = db.queryByExample(proto);
listResult(result);
```

Τα αποτελέσματα που μας δίνει αυτό το ερώτημα είναι αυτά :

```
run:
Saved at C:\Users\michael/Documents/Ptixiaki/db4oDB/db4oTestDb.db4o
Results number: 100
Trainer{Person{name=Alexander, surname=Makarov,
country_of_origin=Russia}Trainercode=1}
Athlete{Person{name=Sergey, surname=Makarov, country_of_origin=Russia}
trainer=Trainer{Person{name=Alexander, surname=Makarov,
country_of_origin=Russia}Trainercode=1}, Athletecode=35, gender=Male,
date_of_birth=19/3/1973, weight=102, height=190,
country_of_participation=Russia}
Trainer{Person{name=Maris, surname=Griva,
country_of_origin=Latvia}Trainercode=4}
Athlete{Person{name=Eriks, surname=Rags, country_of_origin=Latvia}
trainer=Trainer{Person{name=Maris, surname=Griva,
country_of_origin=Latvia}Trainercode=4}, Athletecode=39, gender=Male,
date_of_birth=1/6/1975, weight=95, height=183,
country_of_participation=Latvia}
```

```
Trainer{Person{name=Jukka, surname=Manninen,
country_of_origin=Finland}Trainercode=5}
Athlete{Person{name=Esko, surname=Mikkola, country_of_origin=Finland}
trainer=Trainer{Person{name=Jukka, surname=Manninen,
country_of_origin=Finland}Trainercode=5}, Athletecode=43, gender=Male,
date_of_birth=14/2/1975, weight=95, height=185,
country_of_participation=Finland}
Trainer{Person{name=Don, surname=Babbitt,
country_of_origin=USA}Trainercode=6}
Athlete{Person{name=Breaux, surname=Greer, country_of_origin=USA}
trainer=Trainer{Person{name=Don, surname=Babbitt,
country_of_origin=USA}Trainercode=6}, Athletecode=44, gender=Male,
date_of_birth=19/10/1976, weight=92, height=190,
country_of_participation=USA}
Athlete{Person{name=Gerhardus, surname=Pienaar, country_of_origin=South
Africa} trainer=Trainer{Person{name=Don, surname=Babbitt,
country_of_origin=USA}Trainercode=6}, Athletecode=45, gender=Male,
date_of_birth=10/8/1981, weight=104, height=194,
country_of_participation=South Africa}
Trainer{Person{name=Terseus, surname=Liebenberg, country_of_origin=South
Africa}Trainercode=7}
BUILD SUCCESSFUL (total time: 2 seconds)
```

Βέβαια, στο αποτέλεσμα βλέπουμε ένα μικρό πλήθος των δεδομένων της βάσης καθώς υπάρχουν πολλές εγγραφές.

Το db4o παρέχει και έναν απευθείας τρόπο για να ανακτήσουμε όλα τα στιγμιότυπα μιας κλάσης.

```
// retrieveAllPersons
ObjectSet result = db.queryByExample(Person.class);
listResult(result);
```

Ας δούμε όμως τα αποτελέσματα που μας δίνει αυτός ο τρόπος :

```
run:
Saved at C:\Users\michael/Documents/Ptixiaki/db4oDB/db4oTestDb.db4o
Results number: 100
Trainer{Person{name=Alexander, surname=Makarov,
country_of_origin=Russia}Trainercode=1}
Athlete{Person{name=Sergey, surname=Makarov, country_of_origin=Russia}
trainer=Trainer{Person{name=Alexander, surname=Makarov,
country_of_origin=Russia}Trainercode=1}, Athletecode=35, gender=Male,
date_of_birth=19/3/1973, weight=102, height=190,
country_of_participation=Russia}
Trainer{Person{name=Maris, surname=Griva,
country_of_origin=Latvia}Trainercode=4}
Athlete{Person{name=Eriks, surname=Rags, country_of_origin=Latvia}
trainer=Trainer{Person{name=Maris, surname=Griva,
country_of_origin=Latvia}Trainercode=4}, Athletecode=39, gender=Male,
date_of_birth=1/6/1975, weight=95, height=183,
country_of_participation=Latvia}
Trainer{Person{name=Jukka, surname=Manninen,
country_of_origin=Finland}Trainercode=5}
Athlete{Person{name=Esko, surname=Mikkola, country_of_origin=Finland}
trainer=Trainer{Person{name=Jukka, surname=Manninen,
country_of_origin=Finland}Trainercode=5}, Athletecode=43, gender=Male,
```

```
date_of_birth=14/2/1975, weight=95, height=185,
country_of_participation=Finland}
Trainer{Person{name=Don, surname=Babbitt,
country_of_origin=USA}Trainercode=6}
Athlete{Person{name=Breaux, surname=Greer, country_of_origin=USA}
trainer=Trainer{Person{name=Don, surname=Babbitt,
country_of_origin=USA}Trainercode=6}, Athletecode=44, gender=Male,
date_of_birth=19/10/1976, weight=92, height=190,
country_of_participation=USA}
Athlete{Person{name=Gerhardus, surname=Pienaar, country_of_origin=South
Africa} trainer=Trainer{Person{name=Don, surname=Babbitt,
country_of_origin=USA}Trainercode=6}, Athletecode=45, gender=Male,
date_of_birth=10/8/1981, weight=104, height=194,
country_of_participation=South Africa}
Trainer{Person{name=Terseus, surname=Liebenberg, country_of_origin=South
Africa}Trainercode=7}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παρατηρούμε ότι για το ίδιο πλήθος εγγραφών το αποτέλεσμα μας το εμφάνισε πιο γρήγορα στη δεύτερη περίπτωση.

Δημιουργία ερωτήματος για ανάκτηση με βάση το όνομα

```
// retrievePersonByName
Person proto = new Person("Michael", null, null);
ObjectSet result = db.queryByExample(proto);
listResult(result);
```

Αποτέλεσμα της εκτέλεσης αυτής της εντολής είναι :

```
run:
Saved at C:\Users\michael/Documents/Ptixiaki/db4oDB/db4oTestDb.db4o
Results number: 1
Athlete{Person{name=Michael, surname=Frater, country_of_origin=Jamaica}
trainer=null, Athletecode=3, gender=Male, date_of_birth=6/10/1982,
weight=73, height=175, country_of_participation=Jamaica}
BUILD SUCCESSFUL (total time: 0 seconds)
```

### 4.2.4 Ενημέρωση αντικειμένων

Ας δούμε τώρα πως γίνεται η ενημέρωση των αντικειμένων, καθώς είναι μια χρήσιμη λειτουργία στις βάσεις δεδομένων. Χρησιμοποιούμε την ίδια ακριβώς μέθοδο store() και για την ενημέρωση τους.

```
// updatePerson
ObjectSet result = db
        .queryByExample(new Person("Andrew", "Jordan", "USA"));
Person found = (Person) result.next();
found.setName("Michael");
db.store(found);
retrieveAllPersons(db);
```

Βλέποντας ένα – ένα τα βήματα που χρειάζεται να ακολουθήσουμε για να ενημερώσουμε μια εγγραφή ή ένα πλήθος εγγραφών, παρατηρούμε πως πρέπει πρώτα να γεμίσουμε το αντικείμενο result που είναι μία τύπου ObjectSet και αυτό γίνεται με την χρήση ενός ερωτήματος QBE που του δίνουμε ένα πρότυπο αντικείμενο που θέλουμε να ενημερώσουμε και μας τα επιστρέφει στο result. Έπειτα, δημιουργούμε ένα αντικείμενο τύπου Person το found και εκχωρούμε τη τιμή του result σε αυτό. Οπότε, από τη στιγμή που έχουμε το αντικείμενο που θέλουμε να επεξεργαστούμε στη μεταβλητή found, μπορούμε να χρησιμοποιήσουμε τις μεθόδους του για να μεταβάλουμε τα χαρακτηριστικά του. Όπως παρατηρούμε στο κώδικα, χρησιμοποιούμε τη μέθοδο setName() για να αλλάξουμε το όνομα του Person. Έπειτα, καλούμε τη μέθοδο store() με παράμετρο το αντικείμενο found για να καταχωρηθεί στη βάση δεδομένων μας. Και στο τέλος, καλούμε μια μέθοδο για να μας εμφανίσει τα αποτελέσματα.

### 4.2.5 Διαγραφή αντικειμένων

Τα αντικείμενα αφαιρούνται από τη βάση δεδομένων μας κάνοντας χρήση της μεθόδου delete().

```
// deleteFirstPersonByName
ObjectSet result = db
        .queryByExample(new Person("Michael" "Jordan", "USA"));
Person found = (Person) result.next();
db.delete(found);
System.out.println("Deleted " + found);
retrieveAllPersons(db);
```

Όπως και στην ενημέρωση έτσι και στη διαγραφή των αντικειμένων, αυτά πρέπει να είναι γνωστά από το σύστημα. Δεν είναι επαρκής τρόπος να παρέχουμε ένα πρωτότυπο αντικείμενο με τις ίδιες τιμές των πεδίων τους.

### 4.2.6 Σύνοψη

Όπως είδαμε, η δημιουργία μιας αντικειμενοστρεφούς βάσης δεδομένων είναι πολύ εύκολη και έτσι ο προγραμματιστής μπορεί πολύ εύκολα να δημιουργήσει αντικείμενα, να τα ενημερώσει, να τα διαγράψει και να τα ανακτήσει. Στη συνέχεια του κεφαλαίου θα αρχίσουμε να μπαίνουμε σε ποιο σύνθετες λειτουργίες του db4o καθώς και στην γλώσσα ερωτημάτων NQ.

## 4.3 Γλώσσες ερωτημάτων

Το db4o παρέχει τρία συστήματα δημιουργίας ερωτημάτων, QBE, NQ, SODA API. Στο προηγούμενο κεφάλαιο αναφερθήκαμε στο QBE που είναι ιδανικό για μια γρήγορη εκκίνηση για τους χρήστες που δεν έχουνε ακόμα εγκλιματιστεί στη φιλοσοφία για την αποθήκευση και ανάκτηση στο db4o. Το NQ είναι η κύρια διασύνδεση του συστήματος ερωτημάτων στο db4o και χρησιμοποιείτε για γενική χρήση. Το SODA είναι ένα βασικό εσωτερικό API. Παρέχεται για συμβατότητα με προηγούμενες εκδόσεις και μπορεί να είναι χρήσιμη για δυναμική παραγωγή ερωτημάτων, όπου το NQ είναι έντονα διατυπωμένο.

### 4.3.1 Query by Example - QBE

Όποτε χρησιμοποιούμε QBE παρέχουμε στο db4o με ένα πρότυπο αντικείμενο. Αυτό θα μας επιστρέψει όλα τα αντικείμενα που ταιριάζουν με όλες τις μη προκαθορισμένες τιμές των πεδίων.

Αυτός ο τρόπος δημιουργίας ερωτημάτων έχει κάποιους περιορισμούς :

- Το db4o πρέπει να προβάλει όλα τα μέλη του πρότυπου αντικειμένου
- Δεν μπορούμε να εκτελέσουμε προηγμένες εκφράσεις ερωτημάτων (AND, OR, NOT κτλ)
- Δεν μπορούμε να περιορίσουμε σε τιμές όπως 0 (ακέραιος), “” κενή συμβολοσειρά) ή nulls (αναφορές τύπων) γιατί θα μεταγλωττίζονταν χωρίς περιορισμούς.
- Πρέπει να είμαστε σε θέση να δημιουργούμε αντικείμενα χωρίς να αρχικοποιούμε τα πεδία τους. Αυτό σημαίνει πως δεν μπορούμε να αρχικοποιήσουμε τα πεδία εκεί που ορίστηκαν. Δεν μπορούμε να υποχρεώσουμε τις συμβάσεις στα αντικείμενα μιας κλάσης που επιτρέπεται μόνο σε μια καθορισμένη κατάσταση αρχικοποίησης.
- Χρειαζόμαστε έναν κατασκευαστή να δημιουργεί τα αντικείμενα χωρίς να αρχικοποιεί τα πεδία.

Για να ξεπεράσουμε όλα αυτά τα εμπόδια, το db4o παρέχει το σύστημα εγγενών ερωτημάτων NQ.

### 4.3.2 Native Queries – NQ

Τα NQ είναι η κύρια διασύνδεση στο db4o για την διενέργεια ερωτημάτων και συστήνετε για την χρησιμοποίηση του στη βάση δεδομένων της εφαρμογής μας. Επειδή τα NQ απλά χρησιμοποιούν τη σημασιολογία της γλώσσας

προγραμματισμού, είναι τέλεια τυποποιημένη και μια ασφαλής επιλογή για το μέλλον.

Ας δούμε όμως πως λειτουργεί

```
// retrieve trainers country of origin = Russia
 List <Trainer> trainers =db.query(new Predicate<Trainer>() {
          @Override
      public boolean match(Trainer trainer) {
           return trainer.getCountry_of_origin().equals("Russia");
       }
  });
 listResult(trainers);
```

Τα αποτελέσματα τις εκτέλεσης αυτού του ερωτήματος είναι :

```
run:
Saved at C:\Users\michael/Documents/Ptixiaki/db4oDB/db4oTestDb.db4o
Results number: 1
Trainer{Person{name=Alexander, surname=Makarov,
country_of_origin=Russia}Trainercode=1}
BUILD SUCCESSFUL (total time: 1 second)
```

Τα NQ λειτουργούν με μία σύμβαση. Μια κλάση που επεκτείνει την κλάση com.db4o.Predicate αναμένεται να έχει μια λογική (Boolean) μέθοδο match() με μία παράμετρο που να περιγράφει την κλάση της. Ας τα πάρουμε όμως από την αρχή. Δημιουργούμε μια λίστα αντικειμένων trainers όπου θέλουμε να εισάγουμε τα αντικείμενα τύπου Trainer που θα μας επιστρέψει η μέθοδος query(). Για να γίνει αυτό χρησιμοποιούμε το αντικείμενο Predicate που περιέχει τη μέθοδο match. Αυτή η μέθοδος, ελέγχει τα αντικείμενα ένα – ένα μέχρι να βρει ένα αντικείμενο που να πληροί αυτό που θέλουμε, στη συγκεκριμένη περίπτωση θέλουμε χώρα προέλευσης Ρωσία. Τη στιγμή που θα βρει μια τέτοια εγγραφή επιστρέφει το αντικείμενο που βρήκε στη κλάση Predicate. Όταν τελειώσει η αναζήτηση τα αντικείμενα που βρέθηκαν εκχωρούνται στη λίστα μας. Και έπειτα μπορούμε να τα εμφανίσουμε στην έξοδο.

Όταν χρησιμοποιούμε NQ μην ξεχνάμε ότι στα σύγχρονα Integrated Development Environments (IDEs) μπορούμε να τα αναθέσουμε να κάνουνε όλους τους ελέγχους για τους τύπους σχετικά με τις εκφράσεις των NQ, εάν χρησιμοποιούμε πρότυπα και αυτόματη συμπλήρωση.

### 4.3.3 SODA API

Το SODA API είναι ένα χαμηλού επιπέδου API δημιουργίας ερωτημάτων που μας επιτρέπει να έχουμε απευθείας πρόσβαση στους κόμβους των γράφων των ερωτημάτων. Αφού το SODA χρησιμοποιεί συμβολοσειρές για να αναγνωρίσει τα πεδία, δεν είναι ούτε ασφαλής για λάθη ούτε μπορεί να μεταγλωττιστεί και

χρειάζεται αρκετές λεπτομέρειες για να συνταχθεί.

## 4.4 Δομημένα αντικείμενα

Σε αυτό το κεφάλαιο θα προσπαθήσουμε να δείξουμε τον τρόπο με τον οποίο μπορούμε να κατασκευάσουμε σύνθετα αντικείμενα καθώς και πως μπορούνε αυτά να συσχετιστούν μεταξύ τους. Σαν παράδειγμα θα πάρουμε τις σχέσεις που υπάρχουνε μεταξύ ενός αθλητή με ένα προπονητή. Ο κάθε προπονητής έχει πολλούς αθλητές, ενώ ο κάθε αθλητής έχει έναν προπονητή. Άρα υπάρχει σχέση ένα – πολλά. Για να δούμε όμως τις κλάσεις μας. Έχουμε λοιπόν την κλάση αθλητής.

```
1 // Athlete Class
2 import java.util.ArrayList;
3 import java.util.List;
4
5 public class Athlete extends Person {
6
7 // features
8    private Trainer trainer;
9    private int Athletecode;
10    private String gender;
11    private String date_of_birth;
12    private int weight;
13    private int height;
14    private String country_of_participation;
15    public List partic;
16
17 // default constructor
18    public Athlete() {
19    }
20 // constructor
21    public Athlete(int Athletecode, String name, String surname, String
22 gender,
23            String date_of_birth, int weight, int height,
24            String country_of_participation, String country_of_origin)
25 {
26        super(name, surname,country_of_origin);
27        this.Athletecode = Athletecode;
28        this.gender = gender;
29       this.date_of_birth = date_of_birth;
30        this.weight = weight;
31        this.height = height;
32        this.country_of_participation = country_of_participation;
33        this.partic = new ArrayList();
34    }
35
36 //getter - setter
37
38    public int getAthletecode() {
39        return Athletecode;
40    }
41
42   public void setAthletecode(int Athletecode) {
43        this.Athletecode = Athletecode;
44    }
```

```
45
46    public String getCountry_of_participation() {
47        return country_of_participation;
48    }
49
50    public void setCountry_of_participation(String
51country_of_participation) {
52        this.country_of_participation = country_of_participation;
53    }
54
55    public String getDate_of_birth() {
56        return date_of_birth;
57    }
58
59    public void setDate_of_birth(String date_of_birth) {
60        this.date_of_birth = date_of_birth;
61    }
62
63    public String getGender() {
64        return gender;
65    }
66
67    public void setGender(String gender) {
68        this.gender = gender;
69    }
70
71    public int getHeight() {
72        return height;
73    }
74
75    public void setHeight(int height) {
76        this.height = height;
77    }
78
79    public Trainer getTrainer() {
80        return trainer;
81    }
82
83    public void setTrainer(Trainer trainer) {
84        this.trainer = trainer;
85    }
86
87    public int getWeight() {
88        return weight;
89    }
90
91    public void setWeight(int weight) {
92        this.weight = weight;
93    }
94
95    public List getPartic() {
96        return partic;
97    }
98
99
100    // a method to add paricipations to athlete
101    public void addPartic(Participates participation){
102        partic.add(participation);
103    }
104
105       //to string method
106    @Override
```

```
107  public String toString() {
108        return "Athlete{" + super.toString() +  " trainer=" + trainer
109 +
110                 ", Athletecode=" + Athletecode + ", gender=" + gender
111                 + ", date_of_birth=" + date_of_birth
112                + ", weight=" + weight + ", height=" + height
113                 + ", country_of_participation=" +
114country_of_participation +'}';
115     }
116
117
118
119 }
```

Παρατηρούμε πως υπάρχει ένα πεδίο τύπου Trainer που θα πάρει ένα αντικείμενο τύπου Trainer, δηλαδή τον προπονητή με όλα τα στοιχεία του. Στη γραμμή 8 που βλέπου με το πεδίο Trainer θα εισάγουμε τη τιμή του προπονητή. Επειδή έχουνε σχέση 1 – πολλά, σημαίνει πως κάθε αθλητής έχει ένα και μόνο προπονητή. Ας δούμε τώρα και την κλάση Προπονητής.

```
1 // Trainer Class
2 import java.util.ArrayList;
3 import java.util.List;
4 public class Trainer extends Person {
5    //features
6    private int Trainercode;
7    public List athletes;
8     //default constructor
9    public Trainer() {
10    }
11    //constructor
12    public Trainer(int Trainercode, String name, String surname,
13            String country_of_origin) {
14        super(name,surname,country_of_origin);
15        this.Trainercode = Trainercode;
16        this.athletes=new ArrayList();
17    }
18    //getter - setter
19    public int getTrainercode() {
20        return Trainercode;
21    }
22
23    public void setTrainercode(int Trainercode) {
24        this.Trainercode = Trainercode;
25    }
26
27    public void addAthlete(Athlete ath){
28        athletes.add(ath);
29    }
30    //to string
31    @Override
32    public String toString() {
33        return "Trainer{" + super.toString() + "Trainercode=" +
34Trainercode
35                + /*", athletes=" + athletes +*/ '}';
36    }
37
38 }
```

Τώρα παρατηρούμε την ύπαρξη μιας λίστας αντικειμένων που θα εκχωρήσουμε τους αθλητές που έχει ο κάθε προπονητής. Εδώ καταλαβαίνουμε ότι είναι το πολλά τις συσχέτισης μας. Δηλαδή στη γραμμή 7 βλέπουμε μια λίστα αθλητών. Οπότε κάθε προπονητής έχει 1 μέχρι και πολλούς αθλητές. Ας μην ξεχνάμε όμως ότι και οι δύο αυτές κλάσεις κληρονομούνε χαρακτηριστικά από την κλάση Person

```
//Person Class
package db4o;

public class Person {

    private String name;
    private String surname;
    private String country_of_origin;

    public Person() {
    }

    public Person(String name, String surname, String country_of_origin)
{
        this.name = name;
        this.surname = surname;
        this.country_of_origin = country_of_origin;
    }


    public String getCountry_of_origin() {
        return country_of_origin;
    }

    public void setCountry_of_origin(String country_of_origin) {
        this.country_of_origin = country_of_origin;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }


    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    @Override
    public String toString() {
        return "Person{" + "name=" + name + ", surname=" + surname
                + ", country_of_origin=" + country_of_origin + '}';
    }



}
```

### 4.4.1 Αποθήκευση δομημένων αντικειμένων

Ας δημιουργήσουμε λοιπόν έναν προπονητή

```
Trainer tr1 = new Trainer(1, "Alexander", "Makarov", "Russia");
```

Ας δημιουργήσουμε και έναν αθλητή

```
Athlete ath35 = new Athlete(35, "Sergey", "Makarov", "Male", "19/3/1973",
102, 190, "Russia", "Russia");
```

Αλλά ο αθλητής πρέπει να έχει και προπονητή οπότε

```
ath35.setTrainer(tr1);
```

Και ο προπονητής θα έχει τον αθλητή

```
tr1.addAthlete(ath35);
```

Για να αποθηκευτούνε στην βάση δεδομένων μας απλώς κάνουμε την χρήση της μεθόδου store()

```
db.store(tr1);
db.store(ath35);
```

### 4.4.2 Ανάκτηση δεδομένων

Ας δούμε τώρα πως γίνεται η ανάκτηση των δεδομένων στο db4o με χρήση NQ

```
// retrieve athletes where trainers country of origin = USA
List <Athlete> athletes = db.query(new Predicate<Athlete>(){
    @Override
    public boolean match(Athlete athlete){
        return athlete.getTrainer().getCountry_of_origin().equals("USA");
    }
});
listResult(athletes);
```

Ας δούμε τώρα τα αποτελέσματα του ερωτήματος :

```
run:
Saved at C:\Users\michael/Documents/Ptixiaki/db4oDB/db4oTestDb.db4o
Results number: 2
Athlete{Person{name=Breaux, surname=Greer, country_of_origin=USA}
trainer=Trainer{Person{name=Don, surname=Babbitt,
country_of_origin=USA}Trainercode=6}, Athletecode=44, gender=Male,
date_of_birth=19/10/1976, weight=92, height=190,
country_of_participation=USA}
```

```
Athlete{Person{name=Gerhardus, surname=Pienaar, country_of_origin=South
Africa} trainer=Trainer{Person{name=Don, surname=Babbitt,
country_of_origin=USA}Trainercode=6}, Athletecode=45, gender=Male,
date_of_birth=10/8/1981, weight=104, height=194,
country_of_participation=South Africa}
BUILD SUCCESSFUL (total time: 2 seconds)
```

Τα αποτελέσματα εκχωρούνται σε μια λίστα από τη μέθοδο match που συγκρίνει τις εγγραφές που υπάρχουνε αν όντως πληρούν τις προϋποθέσεις. Όπως είχαμε αναφέρει και σε προηγούμενο κεφάλαιο φτιάχνουμε μια λίστα που θα επιστρέψουν οι αθλητές και μετά την εμφανίζουμε.

## 4.5 Μεταδεδομένα στο db4o

Τα μεταδεδομένα που παρέχει το db4o παρέχουνε πρόσβαση στη δομή του αρχείου της βάσης δεδομένων. Η κύρια χρήση του είναι για τη λειτουργία του refactoring. Μπορούμε να έχουμε πρόσβαση σε αυτά από το extended object container. Μπορούμε να ζητήσουμε από αυτό να μας επιστρέψει για όλες τις κλάσης ή και μεμονωμένα. Για να πάρουμε τις πληροφορίες για τα μεταδεδομένα από μία μεμονωμένη κλάση μπορούμε να δώσουμε το πλήρες όνομά της, την ίδια την κλάση, ή ακόμα και ένα στιγμιότυπο ενός συγκεκριμένου τύπου. Σημειώνουμε πως το db4o μπορεί να επιστρέψει πληροφορίες σχετικά με εσωτερικά στιγμιότυπα του db4o που είναι αποθηκευμένα.

```
// Get the information about all stored classes.
StoredClass[] classesInDB = container.ext().storedClasses();
for (StoredClass storedClass : classesInDB) {
System.out.println(storedClass.getName());
}
// Information for a certain class
StoredClass metaInfo = db.ext().storedClass(Person.class);
System.out.println(metaInfo);
```

Η εκτέλεση των παραπάνω εντολών μας δίνουν τα εξής αποτελέσματα :

```
run:
Saved at C:\Users\michael/Documents/Ptixiaki/db4oDB/db4oTestDb.db4o
com.db4o.ext.Db4oDatabase
com.db4o.StaticField
com.db4o.StaticClass
db4o.Person
db4o.Trainer
java.util.AbstractCollection
java.util.AbstractList
java.util.ArrayList
db4o.Athlete
db4o.Sport
db4o.Games
db4o.Stadium
db4o.Participates
```

```
BUILD SUCCESSFUL (total time: 0 seconds)
```

Και :

```
run:
Saved at C:\Users\michael/Documents/Ptixiaki/db4oDB/db4oTestDb.db4o
StoredClass(db4o.Person)
BUILD SUCCESSFUL (total time: 0 seconds)
```

Η διασύνδεση StoredClass μας παρέχει όλα τα μεταδεδομένα που γνωρίζει το db4o. Μπορούμε να πάρουμε το όνομα της κλάσης, να ρωτήσουμε για το πλήθος των στιγμιότυπων, μιας λίστας με τις ταυτότητες των αντικειμένων και να πάρουμε τα μεταδεδομένα από τις υπερκλάσης τους. Η πιο χρήσιμη πληροφορία σχετικά με τις αποθηκευμένες κλάσης είναι τα μεταδεδομένα που μας δίνουν μια λίστα με τα πεδία που είναι αποθηκευμένα. Μπορούμε να πάρουμε μια λίστα με όλα τα πεδία ή να ζητήσουμε για συγκεκριμένα. Σημειώνουμε πως τα μεταδεδομένα μπορεί να μας επιστρέψουν πεδία που δεν υπάρχουν πια. Αυτό είναι χρήσιμο για το refactoring.

```
StoredClass metaInfoForPerson =
container.ext().storedClass(Person.class);
// Access all existing fields
for (StoredField field : metaInfoForPerson.getStoredFields()) {
System.out.println("Field: "+field.getName());
}
// Accessing the field 'name' of any type.
StoredField nameField = metaInfoForPerson.storedField("name", null);
// Accessing the string field 'name'. Important if this field had another
time in previous
// versions of the class model
StoredField ageField = metaInfoForPerson.storedField("age",int.class);
// Check if the field is indexed
boolean isAgeFieldIndexed = ageField.hasIndex();
// Get the type of the field
String fieldType = ageField.getStoredType().getName();
```

Ο παραπάνω κώδικας μας δίνει αποτελέσματα όπως :

```
run:
Field: name
Field: surname
Field: country_of_origin
BUILD SUCCESSFUL (total time: 0 seconds)
```

Στα μεταδεδομένα για το πεδίο μπορούμε να μάθουμε το όνομα, το τύπο και αν το πεδίο έχει κάποιο ευρετήριο (index). Μπορούμε επίσης να έχουμε πρόσβαση στις τιμές κάποιου αντικειμένου μέσω των αποθηκευμένων πεδίων. Αυτό μας επιτρέπει να έχουμε πρόσβαση σε πληροφορίες που είναι αποθηκευμένες στη βάση δεδομένων μας, αλλά έχουνε αφαιρεθεί από το μοντέλο της κλάσης, πράγμα που το καθιστά χρήσιμο για το refactoring.

```
StoredClass metaForPerson = container.ext().storedClass(Person.class);
StoredField metaNameField = metaForPerson.storedField("name", null);
ObjectSet<Person> persons = container.query(Person.class);
for (Person person : persons) {
String name = (String)metaNameField.get(person);
System.out.println("Name is "+name);
}
```

Στο παραπάνω απόσπασμα κώδικα θα σας δείξουμε ένα τρόπο που μπορούμε να χρησιμοποιήσουμε τα μεταδεδομένα. Τα αποτελέσματα του παραπάνω κώδικα είναι :

```
run:
Saved at C:\Users\michael/Documents/Ptixiaki/db4oDB/db4oTestDb.db4o
Name is Alexander
Name is Sergey
Name is Maris
Name is Eriks
Name is Jukka
Name is Esko
Name is Don
Name is Breaux
Name is Gerhardus
Name is Terseus
Name is Christian
Name is Christian
Name is Janis
Name is Voldemars
Name is Rudolph
Name is William
Name is Jaroslev
Name is John
Name is Steve
Name is Nick
Name is Heino
Name is Andrus
Name is Steve
Name is Oliver
Name is Karin
Name is Peter
Name is Jaroslav
Name is Jan
Name is Miroslav
Name is Klaus
Name is Boris
Name is Dionisio
Name is Osleidys
Name is Helge
Name is Mirela
Name is Vassilis
Name is Steffi
Name is Yannis
Name is Savva
Name is Kari
Name is Taina
Name is Henryk
Name is Barbara
Name is Dana
Name is Andreas
Name is Vadims
```

```
Name is Alexander
Name is Tero
Name is Matti
Name is Isbel
Name is Rongxlang
Name is Yukifumi
Name is Sergey
Name is Stuart
Name is Jae Myong
Name is Edi
Name is Peter
Name is Gergely
Name is Ronny
Name is Manuel
Name is David
Name is Marian
Name is Sonia
Name is Lavern
Name is Noraida
Name is Tetyana
Name is Felicia
Name is Shawn
Name is Obadele
Name is Vicente
Name is Matic
Name is Deji
Name is Nicolas
Name is Gennadiy
Name is Idrissa
Name is Justin
Name is Jason
Name is Uchenna
Name is Nobuharu
Name is Yeoryios
Name is Ronald
Name is Nicconnor
Name is Eddy
Name is Kim
Name is Michael
Name is Frank
Name is Joshua
Name is Alexander
Name is Andrey
Name is Jaysuma
Name is Maurice
Name is Asafa
Name is Leonard
Name is Lukasz
Name is Kareem
Name is Simone
Name is Jarbas
Name is Eric Pacome
Name is Aziz
Name is Francis
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Επίλογος

Γενικά, από ότι έχουμε παρατηρήσει για να δημιουργήσουμε μια απλή βάση δεδομένων το μόνο που χρειαζόμαστε να γνωρίζουμε καλά είναι η γλώσσα προγραμματισμού Java. Κατά τα άλλα, χρησιμοποιούμε τέσσερις βιβλιοθήκες μόνο και ακόμα δυο για να εξάγουμε τα μεταδεδομένα μας :

```
import com.db4o.Db4oEmbedded
import com.db4o.ObjectContainer
import com.db4o.ObjectSet
import com.db4o.query.Predicate
import com.db4o.ext.StoredClass;
import com.db4o.ext.StoredField;
```

# ΚΕΦΑΛΑΙΟ 5

# Εξαγωγή – Εισαγωγή XML από αντικειμενοστρεφή βάση δεδομένων

## Εισαγωγή

Η κυριότερη πλατφόρμα για τη μεταφορά των δεδομένων και πιο διαδεδομένη είναι το XML. Δυστυχώς, το db4o δεν παρέχει κάποιο API για να χρησιμοποιηθεί για την εισαγωγή – εξαγωγή XML εγγράφων, αλλά υπάρχουνε πάρα πολλά εργαλεία σειριοποίησης σε XML και έτσι δεν χρειάζεται να υπάρξει ένα τέτοιο. Ότι χρειαζόμαστε για να εξάγουμε τη βάση δεδομένων μας ή τα αποτελέσματα κάποιου ερωτήματος είναι :

1. Ανάκτηση δεδομένων από τη βάση δεδομένων
2. Σειριοποίηση τους σε διαμόρφωση XML
3. Αποθήκευση της δέσμης XML σε έγγραφο στο δίσκο, στη μνήμη ή σε άλλη βάση δεδομένων

Η διαδικασία της εισαγωγής είναι ακριβώς το αντίθετο

1. Διαβάζουμε τη δέσμη XML
2. Δημιουργούμε τα αντικείμενα από το XML
3. Αποθηκεύουμε τα αντικείμενα στο db4o

## 5.1 XStream API

Για τα παραδείγματα μας θα χρησιμοποιήσουμε το XStream API από την codehaus [39]. Αυτό είναι μια βιβλιοθήκη για την σειριοποιησιμότητα των αντικειμένων σε XML και αντίστροφα. Είναι εύκολο στη χρήση, δεν χρειάζεται να έχουμε mapping, η απόδοση του είναι πολύ γρήγορη με μικρό αποτύπωμα μνήμης που είναι καθοριστικό μέρος του σχεδιασμού του, καθιστώντας το κατάλληλο για μεγάλους γράφους αντικειμένων ή για συστήματα που έχουνε μεγάλη διακίνηση μηνυμάτων. Αποδίδει καθαρό έγγραφο XML, καθώς δεν διπλασιάζεται καμία πληροφορία από αντανάκλαση. Δεν χρειάζεται να γίνει κάποια αλλαγή στα αντικείμενα καθώς σειριοποιεί και τα private αλλά και τα final πεδία. Υποστηρίζει non – public και εσωτερικές κλάσεις και δεν χρειάζονται να έχουνε default κατασκευαστή.

## 5.2 Εισαγωγή στο XStream API

Ας θεωρήσουμε την κλάση Person, που περιγράφει μία οντότητα, όπως έχουμε δει προγενέστερα. Θα χρησιμοποιήσουμε αυτή για να μπορέσουμε να δώσουμε απλά και σαφή παραδείγματα για τον τρόπο λειτουργίας του API. Για να χρησιμοποιήσουμε το API απλά αρχικοποιούμε την XStream κλάση :

```
XStream xstream = new XStream();
```

Χρειαζόμαστε βέβαια τις βιβλιοθήκες xstream-[version].jar και kxml2-min-[version].jar στο εκάστοτε classpath. Το kXML είναι μία πολύ γρήγορη υλοποίηση του XML pull – parser. Εάν δεν θέλουμε να συμπεριλάβουμε αυτή την εξάρτηση μπορούμε κάλλιστα να χρησιμοποιήσουμε το JAXP DOM parser ή αφού βρισκόμαστε στην έκδοση 6 της JAVA να χρησιμοποιήσουμε τον ενσωματωμένο Stax parser.

```
XStream xstream = new XStream(new DomDriver()); // does not require XPP3
library
XStream xstream = new XStream(new StaxDriver()); // does not require XPP3
library
```

Σημειώστε πως αυτή η κλάση είναι μια όψη που σχεδιάστηκε για απλές λειτουργίες. Για μεγαλύτερη ευελιξία, μπορείτε να επιλέξετε να δημιουργήσετε τη δικιά σας όψη που θα συμπεριφέρεται διαφορετικά.

Τώρα, για να κάνουμε το XML να εξαχθεί από το XStream ποιο σαφή, μπορούμε να δημιουργήσουμε ψευδώνυμα (alias) για κάθε κλάση σε στοιχεία του XML. Αυτό είναι το μόνο mapping που χρειάζεται να κάνουμε στο XStream και είναι και προαιρετικό.

```
xstream.alias("person", Person.class);
```

Τώρα ας δούμε πως θα γίνει η εξαγωγή των αντικειμένων σε XML :

```
//export to xml
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;
import com.thoughtworks.xstream.XStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

public class XmlMain {
    public static void main(String [] args)throws IOException{
        ObjectContainer db = Db4oEmbedded.openFile("database.db4o");
     try {
        Person[] persons = db.query(Person.class).toArray(new Person[0]);
        System.out.println(persons[0].toString());
        XStream xstream = new XStream();
        xstream.alias("Person",Person.class);
        OutputStream stream = new FileOutputStream("person.xml");
```

```
      try {
            xstream.toXML(persons, stream);
        } finally {
            stream.close();
    }
} finally {
        db.close();
}

    }
}
```

Που δίνει έξοδο :

```
<Person-array>
  <db4o.Trainer>
    <name>Alexander</name>
    <surname>Makarov</surname>
    <country__of__origin>Russia</country__of__origin>
    <Trainercode>1</Trainercode>
    <athletes>
      <db4o.Athlete>
        <name>Sergey</name>
        <surname>Makarov</surname>
        <country__of__origin>Russia</country__of__origin>
        <trainer reference="../../.."/>
        <Athletecode>35</Athletecode>
        <gender>Male</gender>
        <date__of__birth>19/3/1973</date__of__birth>
        <weight>102</weight>
        <height>190</height>
        <country__of__participation>Russia</country__of__participation>
        <partic/>
      </db4o.Athlete>
    </athletes>
  </db4o.Trainer>
</Person-array>
```

Βλέπουμε πως μετά την εκτέλεση της μεθόδου όλα τα αντικείμενα από τη βάση δεδομένων έχουνε αποθηκευτεί σε ένα XML έγγραφο σαν ένα πίνακα. Εδώ παρουσιάζουμε μόνο ένα πεδίο αυτού του πίνακα για συντομία. Αντίστοιχα η διαδικασία εισαγωγής δεδομένων στη βάση δεδομένων μας γίνετε ως εξής :

```
//import xml
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;
import com.thoughtworks.xstream.XStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

public class ImportXml {
    public static void main(String args []) throws IOException{
        ObjectContainer container =
Db4oEmbedded.openFile("database.db4o");
try {
    XStream xstream = new XStream();
    InputStream stream = new FileInputStream("person.xml");
    try {
        Person[] p = (Person[]) xstream.fromXML(stream);
```

```
            for (Person person : p) {
                container.store(person);
            }
        } finally {
            stream.close();
        }
    } finally {
        container.close();
    }
        }
    }
```

Φυσικά δεν είναι μόνο αυτά που μπορούμε να δούμε γύρω από την εξαγωγή - εισαγωγή δεδομένων από XML αρχεία. Μπορούμε να δούμε μετονομασία πεδίων, αποθήκευση συλλογών, σχέσεων κτλ. Εδώ περιγράφοντας την βασική λειτουργία είδαμε πως για την εξαγωγή χρησιμοποιούμε την `xstream.toXML(Object obj);`

Και για την εισαγωγή την `xstream.fromXML(String xml);`

## 5.3 Μεταδεδομένα σε XML

Φυσικά, είναι πολύ εύκολο να κατανοήσουμε πως μπορούμε να εξάγουμε τα μεταδεδομένα μας σε ένα XML αρχείο. Έτσι, μπορούμε να πάρουμε πληροφορίες για την δομή των κλάσεων μας, το όνομα μιας κλάσης, τα πεδία του, τους τύπους των πεδίων, καθώς και τους τύπους των μεθόδων που υπάρχουνε σε κάθε κλάση καθώς και τις υπογραφές τους. Δυστυχώς, δεν είναι δυνατό να μπορέσουμε να εξάγουμε το σώμα της μεθόδου. Φυσικά, από την έρευνά μας δεν βρέθηκε κάποιος σίγουρος τρόπος για να γίνει αυτή η εξαγωγή καθώς APIs που βρέθηκαν για την μεταφορά του σώματος μιας μεθόδου από το byte code της Java, δεν φάνηκαν αξιόπιστα για αυτή τη λειτουργία. Ας δούμε όμως ένα παράδειγμα για το πώς γίνεται αυτό.

```
//example of metadata to XML
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;
import com.thoughtworks.xstream.XStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;

public class XmlMain {
    public static void main(String [] args)throws IOException{
        ObjectContainer db = Db4oEmbedded.openFile("db4oTestDb.db4o");
     try {
        List p = new ArrayList();
        p.add(Person.class.getMethods());
        XStream xstream = new XStream();
        xstream.setMode(XStream.XPATH_ABSOLUTE_REFERENCES);
        xstream.alias("Person",Person.class);
```

```
        xstream.alias("Athlete",Athlete.class);
        xstream.alias("Trainer",Trainer.class);
        OutputStream stream = new FileOutputStream("hopefully.xml");
      try {
            xstream.toXML(p, stream);
        } finally {
            stream.close();
    }
} finally {
        db.close();
}
```

Έτσι, έχουμε σαν αποτέλεσμα ένα αρχείο που μας δίνει μια λίστα με τις μεθόδους τις κλάσης και τους τύπους τους, καθώς και τους τύπους των παραμέτρων τους.

```
<list>
  <method-array>
    <method>
      <class>db4o.Person</class>
      <name>getCountry_of_origin</name>
      <parameter-types/>
    </method>
    <method>
      <class>db4o.Person</class>
      <name>setCountry_of_origin</name>
      <parameter-types>
        <class>java.lang.String</class>
      </parameter-types>
    </method>
    <method>
      <class>db4o.Person</class>
      <name>getSurname</name>
      <parameter-types/>
    </method>
    <method>
      <class>db4o.Person</class>
      <name>setSurname</name>
      <parameter-types>
        <class>java.lang.String</class>
      </parameter-types>
    </method>
    <method>
      <class>db4o.Person</class>
      <name>toString</name>
      <parameter-types/>
    </method>
    <method>
      <class>db4o.Person</class>
      <name>getName</name>
      <parameter-types/>
    </method>
    <method>
      <class>db4o.Person</class>
      <name>setName</name>
      <parameter-types>
        <class>java.lang.String</class>
      </parameter-types>
    </method>
    <method>
      <class>java.lang.Object</class>
```

```
      <name>wait</name>
      <parameter-types>
        <class>long</class>
      </parameter-types>
    </method>
    <method>
      <class>java.lang.Object</class>
      <name>wait</name>
      <parameter-types/>
    </method>
    <method>
      <class>java.lang.Object</class>
      <name>wait</name>
      <parameter-types>
        <class>long</class>
        <class>int</class>
      </parameter-types>
    </method>
    <method>
      <class>java.lang.Object</class>
      <name>equals</name>
      <parameter-types>
        <class>java.lang.Object</class>
      </parameter-types>
    </method>
    <method>
      <class>java.lang.Object</class>
      <name>hashCode</name>
      <parameter-types/>
    </method>
    <method>
      <class>java.lang.Object</class>
      <name>getClass</name>
      <parameter-types/>
    </method>
    <method>
      <class>java.lang.Object</class>
      <name>notify</name>
      <parameter-types/>
    </method>
    <method>
      <class>java.lang.Object</class>
      <name>notifyAll</name>
      <parameter-types/>
    </method>
  </method-array>
</list>
```

## Επίλογος

Από ότι παρατηρήσαμε, για να εξάγουμε τα στοιχεία της βάσης δεδομένων μας δεν χρειαζόμαστε κάποια εξειδικευμένη γνώση. Βέβαια θα ήταν χρήσιμο να μπορούσαμε να εξάγουμε αυτόματα από κάποιο API απευθείας τη βάση δεδομένων μας αντί να χρησιμοποιήσουμε αυτό τον τρόπο και να εξάγουμε τα δεδομένα από τις κλάσεις μία προς μία.

# ΣΥΜΠΕΡΑΣΜΑΤΑ

Από ότι έχουμε αντιληφθεί τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων έχουνε μπει στην αγορά λογισμικού γιατί μπορούν και μας παρέχουνε την λειτουργικότητα, την φορητότητα, την αποτελεσματικότητα και μια τεράστια γκάμα λειτουργιών που μπορούμε να επιτύχουμε τα και να καλύψουμε τις ανάγκες του λογισμικού που κατασκευάζουμε. Ένα πρόβλημα που αντιμετωπίσαμε στην υλοποίηση της βάσης δεδομένων ήταν η δυσκολία εξαγωγής των μεταδεδομένων καθώς η κύρια χρήση των μεταδεδομένων είναι για λειτουργίες refactoring. Φυσικά μπορούμε να πάρουμε τα πεδία μιας κλάσης, τους τύπους, καθώς και το όνομα τους. Φυσικά, λόγο της ενθυλάκωσης, δεν είναι εφικτό να έχουμε πρόσβαση στις μεθόδους που έχει υλοποιημένες κάθε οντότητα, αλλά θα ήτανε πολύ χρήσιμο να μπορούσαμε διότι θα μπορούσαμε να τις εξάγουμε σε ένα XML έγγραφο και θα μπορούσαμε να μεταφέρουμε τη βάση ανά πάσα στιγμή. Μια άποψη, για το πώς θα μπορούσαμε να πάρουμε το σώμα των μεθόδων κάθε οντοτήτων είναι μέσα από το κώδικα byte που παράγει η Java, αλλά δεν βρέθηκε κάποιο API που να μπορεί να μας επιστρέφει το σώμα της εκάστοτε μεθόδου. Γενικά, τα αντικειμενοστρεφή συστήματα διαχείρισης βάσεων δεδομένων έχουνε φτάσει ένα πολύ αξιόλογο επίπεδο και μπορούν να παρέχουνε υπηρεσίες από πολύ μικρές βάσεις μέχρι και τις πιο μεγάλες και σύνθετες.

## ΑΝΑΦΟΡΕΣ

*[1]http://ece.ut.ac.ir/classpages/f84/AdvancedDatabase/Paper/Hot_Topic_Paper/http__www-2.cs.cmu.edu_People_clamen_OODBMS_Manifesto_htManifesto_Manifesto.pdf*

*[2] Hall et al. 76*
*P. Hall, J. Owlett, S. Todd, ``Relations and Entities'', in ``Modeling in Data Base Management*
*Systems'', G.M. Nijssen (ed.), pp 201-220, North-Holland, 1976.*
*[3] Maier and Price 84*
*D. Maier and D. Price, ``Data model requirements for engineering applications'', Proceedings of*
*the First International Workshop on Expert Database Systems, IEEE, 1984, pp 759-765*
*[4] Khoshafian and Copeland 86*
*S. Khoshafian and G. Copeland, ``Object identity'', Proceedings of the 1st ACM OOPSLA conference, Portland, Oregon, September 1986*
*[5] Abiteboul and Kanellakis 89*
*S. Abiteboul and P. Kanellakis, ``Object identity as a query language primitive'', Proceedings of the 1989 ACM SIGMOD, Portland, Oregon, June 89*
*[6] Goldberg and Robson 83*
*A. Goldberg and D. Robson, ``Smalltalk-80: the language and its implementation'', Addison-*
*Wesley, 1983.*
*[7] Maier, et al. 84*
*D. Maier, J. Stein, A. Otis, A. Purdy, ``Development of an object-oriented DBMS'' Report CS/E-*
*86-005, Oregon Graduate Center, April 86*
*[8] Caruso and Sciore 87*
*``The VISION Object-Oriented Database Management System'', Proceedings of the Workshop on*
*Database Programming Languages, Roscoff, France, September 1987*
*[9] Banerjee et al. 87*
*J. Banerjee, H.T. Chou, J. Garza, W. Kim, D. Woelk, N. Ballou and H.J. Kim, ``Data model issues for object-oriented applications'', ACM TOIS, January 1987.*
*[10] Bobrow and Steifik 81*
*D. Bobrow and M. Steifik, `` The Loops Manual'', Technical Report LB-VLSI-81-13, Knowledge*
*Systems Area, Xerox Palo Alto Research Center, 1981.*
*[11] G-Base 88*
*``G-Base version 3, Introductory guide'', Graphael, 1988.*
*[12] Caseau 89*
*``A model for a reflective object-oriented language'', Sigplan Notices, Special issue on Concurrent Object-Oriented Programming, March 1989.*
*[13] Stroustrup 86*
*B. Stroustrup, ``The C++ programming language'', Addison-Wesley, 1986.*
*[14] Simula 67*
*``Simula 67 Reference Manual''*

*[15] Schaffert, et al. 86*
*C. Schaffert, T. Cooper, B. Bullis, M. Kilian and C. Wilpolt, ``An introduction to Trellis/Owl'', Proceedings of the 1st OOPSLA Conference, Portland, Oregon, September 1986*
*[16] Atwood 85*
*T. Atwood, ``An object-oriented DBMS for design support applications'', Ontologic Inc. Report.*
*[17] François Bancilhon, Claude Delobel, Paris C. Kanellakis: Building an Object-Oriented Database System, The Story of O2 Morgan Kaufmann 1992*
*[18] Atkinson et al. 83*
*M. Atkinson, P.J. Bayley, K. Chilsom, W. Cockshott and R. Morrison, ``An approach to persistent programming'', Computer Journal, 26(4), 1983, pp 360-365.*
*[19] Bancilhon 86*
*F. Bancilhon, ``A logic programming object oriented cocktail'', ACM SIGMOD Record, 15:3, pp.*
*11-21, 1986.*
*[20] Zaniolo 86*
*C. Zaniolo, ``Object-oriented programming in Prolog '', Proceedings of the first workshop on*
*Expert Database Systems, 1985.*

*[21] Albano et al. 1986*
*A. Albano, G. Gheli, G. Occhiuto and R. Orsini, ``Galileo: a strongly typed interactive conceptual language'', ACM TODS, Vol 10, No. 2, June 1985.*
*[22] Eiffel 87*
*``Eiffel user's manual'', Interactive Software Engineering Inc., TR-EI-5/UM, 1987.*
*[23] Skarra et al. 86*
*A. Skarra, S. Zdonik and S. Reiss, ``An object server for an object oriented database system,''*
*Proceedings of the 1986 International Workshop on Object Oriented Database System, Computer*
*Society Press, IEEE, pp. 196-204, 1986*
*[24] Kim 88*
*W. Kim, ``A foundation for object-oriented databases'', MCC Technical Report, 1988.*
*[25] Dittrich 1986*
*K.R. Dittrich, ``Object-Oriented Database System : The Notions and the issues'', in : Dittrich, K.*
*R. and Dayal, U. (eds): Proceedings of the 1986 International Workshop on Object-Oriented*
*Database Systems, IEEE Computer Science Press*
*[26] Dittrich 1988*
*K. R. Dittrich, ``Preface'', In : Dittrich, K.R. (ed): Advances in Object-Oriented Database Systems, Lecture Notes in Computer Science, Vol, 334, Springer-Verlag, 1988*
*[27] Gemstone (Smalltalk). Web site: http://www.gemstone.com/ , Accessed 2011.*

*[28] Jdoinstruments (objectDB). Web site: http://www.jdoinstruments.org/ , Accessed 2011.*

*[29] Versant Object Database. Web Site: http://www.versant.com/ , Accessed 2011.*

*[30] ObjectStore. Web Site: http://web.progress.com/en/objectstore/ , Accessed 2011.*

*[31] Jade. Web Site: http://www.jadeworld.com/ , Accessed 2011.*

*[32] Matisse. Web Site: http://www.matisse.com/ , Accessed 2011.*

*[33] db4objects. Web Site: www.db4o.com/ ,Accessed 2011.*

*[34] Objectivity. Web Site: http://www.objectivity.com/ , Accessed 2011.*

*[35] Progress Software. Web Site: http://web.progress.com/en/index.html , Accessed 2011.*

*[36] EyeDB Object Oriented Database Management System. Web Site: http://www.eyedb.org/ , Accessed 2011.*

*[37] McObject Perst. Web Site: http://www.mcobject.com/perst , Accessed 2011.*

*[38] Cache. Web Site: http://www.intersystems.com/cache/ , Accessed 2011.*

*[39] ODABA. Web Site: http://odaba.com/content/start/ , Accessed 2011.*

*[40] Eloquera. Web site: http://eloquera.com/page/home.aspx , Accessed 2011.*

*[41] Generic Object Oriented Database System (GOODS). Web site: http://www.garret.ru/goods.html , Accessed 2011.*

*[42] JODB (Java Objects Database). Web site: http://www.java-objects-database.com/ , Accessed 2011.*

*[43] MyOODB.Web site: http://www.myoodb.org/ , Accessed 2011.*

*[44] NeoDatis ODB. Web site: http://www.neodatis.org/ , Accessed 2011.*

*[45] Orient ODBMS. Web site: http://www.orientechnologies.com/cms/ , Accessed 2011.*

*[46] Ozone Database Project. Web site: http://www.ozone-db.org/frames/home/what.html , Accessed 2011.*

*[47] Statice. Web site: http://www.sts.tu-harburg.de/~r.f.moeller/symbolics-info/statice.html , Accessed 2011.*

*[48] VOSS (Virtual Object Storage System). Web site: http://voss.logicarts.com/ , Accessed 2011.*

*[49] Obsidian Dynamics (DTS/S1). Web site: http://obsidiandynamics.com/dts/index.html , Accessed 2011.*

*[50] Cattell R.G., Barry D.K.. The Object Data Standard: ODMG 3.0 (Morgan Kaufmann Publishers) 1999.*

*[51] Strozzi, Carlo: NoSQL - A Relational Database Management System. Web Site: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page, Accessed 2010.*

*[52] LINQ. Web Site: http://msdn.microsoft.com/en-us/netframework/aa904594 , Accessed 2011.*

*[53] Maydene Fisher, Jon Ellis, Jonathan Bruce. JDBC™ API Tutorial and Reference (3rd Edition).The Java Series, Sun, 2003.*

*[54] Daoqi Yang. Java(TM) Persistence with JPA. Outskirts Press, 2010.*

*[55] Oracle Toplink. Website: http://www.oracle.com/technetwork/middleware/toplink/tl-grid-097210.html , Accessed 2011.*

*[56] Boss Hibernate. Web Site: http://www.hibernate.org/ . Accessed 2011.*

*[57] Brian Sam-Bodden. Beginning POJOs From Novice to Professional. Apress Media LLC,2006.*

*[58] M.L. Fussell. Foundations of object relational mapping. White Paper, http://www.chimu.com, 1997*

*[59] Oracle JDO. Web Site: www.oracle.com/technetwork/java/index-jsp-135919.html . Accessed 2011.*

*[60] Java Data Objects. Web Site: http://db.apache.org/jdo/ .Accessed 2011.*

*[61] PolePosition Web Site: http://www.polepos.org*

*[62] Netbeans Web Site : http://www.netbeans.com*

*[63] http://www.trgtd.com.au/licenses/cddl.html*

*[64] http://www.gnu.org/licenses/gpl-2.0.html*

*[65] http://xstream.codehaus.org/index.html*

# ΠΑΡΑΡΤΗΜΑ Α

## Προδιαγραφές του db4o

| Platforms | |
|---|---|
| **Java** | ✓J2EE<br>✓J2SE<br>✓Android<br>✓Compatible Frameworks : Spring, OSGi, DataNucleus, Restlet, Griffon, Guide |
| **.NET** | ✓.NET (3.5, 4.0)<br>✓Compact Framework (2.0, 3.5)<br>✓Windows (XP, Vista)<br>✓Windows Mobile / PocketPC / Windows Phone<br>✓Mono (partial)<br>✓Silverlight<br>✓Compatible Frameworks : Castle, Eiffel, Spring.net |

| Languages | |
|---|---|
| **Java** | ✓JDK 5+<br>✓Scala |
| **.NET** | ✓C#<br>✓Visual Basic |

| Commands | |
|---|---|
| **Sessions** | ✓Start and end |
| **Database files** | ✓Create, open, close and delete |
| **Transactions** | ✓Commit and Rollback |
| **Objects** | ✓Store, retrieve, update (incl. cascaded), replicate, delete (incl. cascaded) |
| **Messaging** | ✓TCP/IP |

| Transparency | |
|---|---|
| **Language construct** | ✓Primitive types<br>✓Strings<br>✓Arrays<br>✓Multi-dimensional arrays<br>✓Inner classes<br>✓Java/C# collections<br>✓Classes without public constructors<br>✓.NET structs<br>✓Blobs (stored outside of DB file) |

| | |
|---|---|
| **Non – Intrusive** | ✓Without deriving from a specific base class<br>✓Without implementing a specific interface<br>✓Without modifications to source code<br>✓Without implementing Serializable |
| **Private Fields** | ✓Storable |
| **File I/O** | ✓Pluggable |
| **Reflector** | ✓Pluggable<br>✓Generic |
| **Aliases** | ✓Class aliasing for class – to – class mappings |
| **Transparent Persistence** | ✓Programmatic or via byte – code instrumentation |

| **Query Languages / API** | |
|---|---|
| **Object oriented** | ✓LINQ<br>✓Native Queries (NQ)<br>✓Query By Example (QBE)<br>✓S.O.D.A |
| **SQL** | ✓Only via replication to many relational databases<br>✓With Third – Party products (e.g. Datanucleus sql4o) |
| **XML** | ✓With Third – Party products (e.g. XStream) |

| **Modes / Concurrency** | |
|---|---|
| **Operation Modes** | ✓Local<br>✓Client / Server |
| **Threads** | ✓Multiple |
| **Transactions** | ✓Multiple / parallel |
| **Semaphores** | ✓Available |
| **Read – Only Mode** | ✓Available |

| **Scalability and Performance** | |
|---|---|
| **Performance benchmark** | ✓Up to 55x faster than Hibernate / MySQL<br>✓See polePosition open – source benchmark [35] |
| **In – Memory Mode** | ✓Available |
| **Client – side** | ✓Single process execution available |
| **Server – side** | ✓Server – side query execution available |
| **DB – aware Collections** | ✓Available |
| **Object Caching** | ✓Available |
| **Pagination** | ✓Server – side cursors (lazy queries) |
| **Indexing** | ✓BTree field indexes |
| **BTree Query Processor** | ✓Big Sets |

| Replication | |
|---|---|
| **db4o Replication Service (dRS)** | ✓100% object oriented : simply replicate objects, not tables<br>✓Uni and bi –directional<br>✓db4o to db4o<br>✓db4o to relational databases (RDBMS), via Hibernate (Java Only)<br>✓db4o to Versant Object Database (VOD, Java Only) |
| **UUID** | ✓Unique Universal Identity over all DB instances |
| **Synchronization** | ✓Querying<br>✓Update<br>✓Delete<br>✓Conflict resolution |

| Reporting | |
|---|---|
| **For .Java Objects** | ✓Actuate BIRT<br>✓Elixir report<br>✓Jaspersoft JasperReports<br>✓Jinfonet JReport |
| **For .NET Objects** | ✓Microsoft Visual Studio ReportView |

| Security and Encryption | |
|---|---|
| **DB File Encryption** | ✓Pluggable File I/O for custom encryption<br>✓With Third – Party products (e.g. XTEA) |

| Availability, Reliability, Zero – Admin | |
|---|---|
| **Transactions** | ✓ACID<br>✓Commit – Recovery on system failures |
| **Thread Safety** | ✓Available |
| **Automatic Recovery** | ✓Available |
| **Online – Backup** | ✓Available |
| **Free Space Management** | ✓Defragmentation API available |

| Internationalization | |
|---|---|
| **Unicode** | ✓Available |

| Refactoring | |
|---|---|
| **Schema Versioning** | ✓Automatic recognition and maintenance |
| **Renaming** | ✓Classes, fields by API<br>✓Access to values of removed fields by API or Object Manager UI |

| Memory and File Size | |
|---|---|
| **DB library footprint** | ✓~1 MB |
| **Minimal RAM footprint** | ✓Application – dependent; deterministic; typically < 3 MB |
| **Maximum DB file size** | ✓254 GB / database file |

# ΠΑΡΑΡΤΗΜΑ Β

## Υλοποίηση Βάσης Δεδομένων

### Κλάση Person

```
package db4o;

/**
 *
 * @author michael
 */
public class Person {

    private String name;
    private String surname;
    private String country_of_origin;

    public Person() {
    }

    public Person(String name, String surname, String country_of_origin)
{
        this.name = name;
        this.surname = surname;
        this.country_of_origin = country_of_origin;
    }


    public String getCountry_of_origin() {
        return country_of_origin;
    }

    public void setCountry_of_origin(String country_of_origin) {
        this.country_of_origin = country_of_origin;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }


    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    @Override
    public String toString() {
        return "Person{" + "name=" + name + ", surname=" + surname
                + ", country_of_origin=" + country_of_origin + '}';
    }}
```

## Κλάση Αθλητής

```
package db4o;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author michael
 */
public class Athlete extends Person {

    private Trainer trainer;
    private int Athletecode;
    private String gender;
    private String date_of_birth;
    private int weight;
    private int height;
    private String country_of_participation;
    public List partic;

    public Athlete() {
    }

    public Athlete(int Athletecode, String name, String surname, String
gender,
            String date_of_birth, int weight, int height,
            String country_of_participation, String country_of_origin) {
        super(name, surname,country_of_origin);
        this.Athletecode = Athletecode;
        this.gender = gender;
        this.date_of_birth = date_of_birth;
        this.weight = weight;
        this.height = height;
        this.country_of_participation = country_of_participation;
        this.partic = new ArrayList();
    }



    public int getAthletecode() {
        return Athletecode;
    }

    public void setAthletecode(int Athletecode) {
        this.Athletecode = Athletecode;
    }

    public String getCountry_of_participation() {
        return country_of_participation;
    }

    public void setCountry_of_participation(String
country_of_participation) {
        this.country_of_participation = country_of_participation;
    }

    public String getDate_of_birth() {
        return date_of_birth;
    }
```

```
    public void setDate_of_birth(String date_of_birth) {
        this.date_of_birth = date_of_birth;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public int getHeight() {
        return height;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public Trainer getTrainer() {
        return trainer;
    }

    public void setTrainer(Trainer trainer) {
        this.trainer = trainer;
    }

    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    public List getPartic() {
        return partic;
    }



    public void addPartic(Participates participation){
        partic.add(participation);
    }

    @Override
    public String toString() {
        return "Athlete{" + super.toString() +  " trainer=" + trainer +
                ", Athletecode=" + Athletecode + ", gender=" + gender
                + ", date_of_birth=" + date_of_birth
                + ", weight=" + weight + ", height=" + height
                + ", country_of_participation=" +
country_of_participation +'}';
    }



}
```

## Κλάση Προπονητής

```
package db4o;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author michael
 */
public class Trainer extends Person {

    private int Trainercode;
    public List athletes;

    public Trainer() {
    }

    public Trainer(int Trainercode, String name, String surname,
            String country_of_origin) {
        super(name,surname,country_of_origin);
        this.Trainercode = Trainercode;
        this.athletes=new ArrayList();
    }

    public int getTrainercode() {
        return Trainercode;
    }

    public void setTrainercode(int Trainercode) {
        this.Trainercode = Trainercode;
    }

    public void addAthlete(Athlete ath){
        athletes.add(ath);
    }

    @Override
    public String toString() {
        return "Trainer{" + super.toString() + "Trainercode=" +
Trainercode
                + /*", athletes=" + athletes +*/ '}';
    }


}
```

## Κλάση Συμμετοχή

```
package db4o;
import java.util.List;
/**
 *
 * @author michael
 */
public class Participates {

    Games game;
    Athlete athlete;
    double performance;
    String cancellation;
    private int Performanceid;

    public List par;

    public Participates() {
    }

    public Participates(Games game, Athlete athlete, double performance,
            String cancellation, int Performanceid) {
        this.game = game;
        this.athlete = athlete;
        this.performance = performance;
        this.cancellation = cancellation;
        this.Performanceid = Performanceid;
    }

    public int getPerformanceid() {
        return Performanceid;
    }

    public void setPerformanceid(int Performanceid) {
        this.Performanceid = Performanceid;
    }

    public Athlete getAthlete() {
        return athlete;
    }

    public void setAthlete(Athlete athlete) {
        this.athlete = athlete;
    }

    public String getCancellation() {
        return cancellation;
    }

    public void setCancellation(String cancellation) {
        this.cancellation = cancellation;
    }

    public Games getGame() {
        return game;
    }

    public void setGame(Games game) {
        this.game = game;
```

```
    }

    public double getPerformance() {
        return performance;
    }

    public void setPerformance(double performance) {
        this.performance = performance;
    }


    public void addGame(Games game){
        par.add(game);
    }


    @Override
    public String toString() {
        return "Participates{" + "game=" + game + //", athlete="
               /* + athlete +*/ ", performance=" + performance + ",
cancellation="
                + cancellation + ", Performanceid=" + Performanceid +
'}';
    }


}
```

## Κλάση Παιχνίδι

```
package db4o;

/**
 *
 * @author michael
 */
public class Games {

    Stadium stadium;

    Sport sport;

    private int Gamecode;
    private String Gameslevel;
    private String gdate;
    private String gtime;

    Participates par;

    public Games() {
    }

    public Games(int Gamecode, String Gameslevel,
            String gdate, String gtime, Sport sport, Stadium stadium) {
        this.stadium = stadium;
        this.sport = sport;
        this.Gamecode = Gamecode;
        this.Gameslevel = Gameslevel;
        this.gdate = gdate;
        this.gtime = gtime;
    }

    public int getGamecode() {
        return Gamecode;
    }

    public void setGamecode(int Gamecode) {
        this.Gamecode = Gamecode;
    }

    public String getGameslevel() {
        return Gameslevel;
    }

    public void setGameslevel(String Gameslevel) {
        this.Gameslevel = Gameslevel;
    }

    public String getGdate() {
        return gdate;
    }

    public void setGdate(String gdate) {
        this.gdate = gdate;
    }

    public String getGtime() {
        return gtime;
```

```
    }

    public void setGtime(String gtime) {
        this.gtime = gtime;
    }

    public Sport getSport() {
        return sport;
    }

    public void setSport(Sport sport) {
        this.sport = sport;
    }

    public Stadium getStadium() {
        return stadium;
    }

    public Participates getPar() {
        return par;
    }

    public void setPar(Participates par) {
        this.par = par;
    }

    public void setStadium(Stadium stadium) {
        this.stadium = stadium;
    }



    @Override
    public String toString() {
        return "Games{" + "stadium=" + stadium + ", sport=" + sport
                + ", Gamecode=" + Gamecode + ", Gameslevel=" + Gameslevel
                + ", gdate=" + gdate + ", gtime=" + gtime + '}';
    }


}
```

## Κλάση Άθλημα

```
package db4o;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author michael
 */
public class Sport {

    private int Sportcode;
    private String Sportname;
    private String gender;
    private double record;
    private String record_date;
    private String recordman_name;
    private String recordnam_surname;
    public List sports;

    public Sport() {
    }

    public Sport(int Sportcode, String Sportname, String gender,
            double record, String record_date, String recordman_name,
            String recordnam_surname) {
        this.Sportcode = Sportcode;
        this.Sportname = Sportname;
        this.gender = gender;
        this.record = record;
        this.record_date = record_date;
        this.recordman_name = recordman_name;
        this.recordnam_surname = recordnam_surname;
        this.sports = new ArrayList();
    }

    public int getSportcode() {
        return Sportcode;
    }

    public void setSportcode(int Sportcode) {
        this.Sportcode = Sportcode;
    }

    public String getSportname() {
        return Sportname;
    }

    public void setSportname(String Sportname) {
        this.Sportname = Sportname;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
```

```
    public double getRecord() {
        return record;
    }

    public void setRecord(double record) {
        this.record = record;
    }

    public String getRecord_date() {
        return record_date;
    }

    public void setRecord_date(String record_date) {
        this.record_date = record_date;
    }

    public String getRecordman_name() {
        return recordman_name;
    }

    public void setRecordman_name(String recordman_name) {
        this.recordman_name = recordman_name;
    }

    public String getRecordnam_surname() {
        return recordnam_surname;
    }

    public void setRecordnam_surname(String recordnam_surname) {
        this.recordnam_surname = recordnam_surname;
    }

    public void addGame(Games game){
        sports.add(game);
    }

    @Override
    public String toString() {
        return "Sport{" + "Sportcode=" + Sportcode + ", Sportname="
                + Sportname + ", gender=" + gender + ", record="
                + record + ", record_date=" + record_date + ",
recordman_name="
                + recordman_name + ", recordnam_surname="
                + recordnam_surname +  ", games=" + sports +'}';
    }


}
```

## Κλάση Γήπεδο

```
package db4o;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author michael
 */
public class Stadium {

    private String Stadiumname;
    private String city;
    private int spectator_capacity;
    private double distance;
    private int press_seats;
    private int photographer_seats;
    public List games;

    public Stadium() {
    }

    public Stadium(String Stadiumname, String city, int
spectator_capacity,
            double distance, int press_seats,
            int photographer_seats) {
        this.Stadiumname = Stadiumname;
        this.city = city;
        this.spectator_capacity = spectator_capacity;
        this.distance = distance;
        this.press_seats = press_seats;
        this.photographer_seats = photographer_seats;
        this.games = new ArrayList();
    }

    public String getStadiumname() {
        return Stadiumname;
    }

    public void setStadiumname(String Stadiumname) {
        this.Stadiumname = Stadiumname;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public double getDistance() {
        return distance;
    }

    public void setDistance(double distance) {
        this.distance = distance;
    }

    public int getPhotographer_seats() {
```

```
        return photographer_seats;
    }

    public void setPhotographer_seats(int photographer_seats) {
        this.photographer_seats = photographer_seats;
    }


    public int getPress_seats() {
        return press_seats;
    }

    public void setPress_seats(int press_seats) {
        this.press_seats = press_seats;
    }

    public int getSpectator_capacity() {
        return spectator_capacity;
    }

    public void setSpectator_capacity(int spectator_capacity) {
        this.spectator_capacity = spectator_capacity;
    }

    public void addGame(Games game){
        games.add(game);
    }

    @Override
    public String toString() {
        return "Stadium{" + "Stadiumname=" + Stadiumname + ", city=" +
city
                + ", spectator_capacity=" + spectator_capacity + ",
distance=" +
                distance + ", press_seats=" + press_seats
                + ", photographer_seats=" + photographer_seats
                + ", games=" + games + '}';
    }


}
```

## Δημιουργία και αποθήκευση Δεδομένων της Βάσης

```
package db4o;
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;
import com.db4o.ext.StoredClass;
import com.db4o.ext.StoredField;
import com.db4o.query.Predicate;
import java.io.File;
import java.util.List;
/**
 *
 * @author michael
 */
public class Main {

    final static String DB4OFILENAME =
System.getProperty("user.home")
+ "/Documents/Ptixiaki/db4oDB/db4oTestDb.db4o";
    public static void main(String [] args){
        System.err.println("Saved at " + DB4OFILENAME);
        //access db4o
    new File(DB4OFILENAME).delete();
    accessDb4o();
    new File(DB4OFILENAME).delete();
    ObjectContainer db = Db4oEmbedded.openFile(Db4oEmbedded
    .newConfiguration(), DB4OFILENAME);

                try{

////////////////////////////// CREATING TRAINERS
////////////////////////////////////

            Trainer tr1 =  new Trainer(1, "Alexander", "Makarov",
"Russia");

            Trainer tr2 =  new Trainer(2, "John", "Trower", "Great
Britain");

            Trainer tr3 =  new Trainer(3, "Heino", "Puuste", "Estonia");

            Trainer tr4 =  new Trainer(4, "Maris", "Griva", "Latvia");

            Trainer tr5 =  new Trainer(5, "Jukka", "Manninen",
"Finland");

            Trainer tr6 =  new Trainer(6, "Don", "Babbitt", "USA");

            Trainer tr7 =  new Trainer(7, "Terseus", "Liebenberg", "South
Africa");

            Trainer tr8 =  new Trainer(8, "Christian", "Nicolay",
"Germany");

            Trainer tr9 =  new Trainer(9, "Janis", "Lusis", "Latvia");

            Trainer tr10 = new Trainer(10, "Rudolph", "Sopko",
"Austria");
```

```
            Trainer tr11 = new Trainer(11, "Jaroslev", "Brabec",
"Czech");

            Trainer tr12 = new Trainer(12, "Steve", "Lemke", "Austria");

            Trainer tr13 = new Trainer(13, "Karin", "Liebhardt",
"Germany");

            Trainer tr14 = new Trainer(14, "Jaroslav", "Brabec",
"Czech");

            Trainer tr15 = new Trainer(15, "Klaus", "Bartonietz",
"Germany");

            Trainer tr16 = new Trainer(16, "Dionisio", "Quintana",
"Cuba");

            Trainer tr17 = new Trainer(17, "Helge", "Zoellkau",
"Germany");

            Trainer tr18 = new Trainer(18, "Vassilis", "Kokkolis",
"Greece");

            Trainer tr19 = new Trainer(19, "Yannis", "Peristeris",
"Greece");

            Trainer tr20 = new Trainer(20, "Kari", "Ihalainen",
"Finland");

            Trainer tr21 = new Trainer(21, "Henryk", "Michalski",
"Poland");

            Trainer tr22 = new Trainer(22, "Dana", "Zatopkova", "Czech");


//////////////////////////// END OF TRAINERS
////////////////////////////////////

//////////////////////// ADDING ATHLETES TO DATABASE
//////////////////////////////
//////////////////////// SETTING TRAINERS TO ATHLETES
//////////////////////////////

            Athlete ath1 =        new Athlete(1, "Aziz", "Zakari",
"Male", "2/9/1976", 85, 175, "Ghana", "Ghana");

            Athlete ath2 =        new Athlete(2, "Kim", "Collins",
"Male", "5/4/1976", 77, 180, "SKN", "SKN");

            Athlete ath3 =        new Athlete(3, "Michael", "Frater",
"Male", "6/10/1982", 73, 175, "Jamaica", "Jamaica");

            Athlete ath4 =        new Athlete(4, "Frank", "Fredericks",
"Male", "2/10/1967", 75, 180, "Namibia", "Namibia");

            Athlete ath5 =        new Athlete(5, "Joshua", "Ross",
"Male", "9/2/1981", 83, 185, "Australia", "Australia");

            Athlete ath6 =        new Athlete(6, "Alexander", "Kosenkow",
"Male", "14/3/1977", 70, 178, "Rusia", "Germany");
```

```
            Athlete ath7 =        new Athlete(7, "Andrey", "Epishin",
"Male", "10/6/1981", 72, 177, "Rusia", "Rusia");

            Athlete ath8 =        new Athlete(8, "Jaysuma", "Saidy
Ndure", "Male", "1/7/1984", 72, 192, "Gam", "Gam");

            Athlete ath9 =        new Athlete(9, "Shawn", "Crawford",
"Male", "14/1/1978", 81, 177, "USA", "USA");

            Athlete ath10 =       new Athlete(10, "Obadele", "Thompson",
"Male", "30/3/1976", 78, 176, "Barbados", "Barbados");

            Athlete ath11 =       new Athlete(11, "Vicente", "Lima",
"Male", "4/6/1977", 65, 166, "Brazil", "Brazil");

            Athlete ath12 =       new Athlete(12, "Matic", "Osovnikar",
"Male", "19/1/1980", 77, 179, "Slovenia", "Slovenia");

            Athlete ath13 =       new Athlete(13, "Deji", "Aliu", "Male",
"22/11/1975", 63, 175, "Nigeria", "Nigeria");

            Athlete ath14 =       new Athlete(14, "Nicolas",
"Macrozonaris", "Male", "22/8/1980", 73, 181, "Canada", "Canada");

            Athlete ath15 =       new Athlete(15, "Gennadiy",
"Chernovol", "Male", "6/6/1976", 85, 187, "Rusia", "Kazakstan");

            Athlete ath16 =       new Athlete(16, "Idrissa", "Sanou",
"Male", "12/6/1977", 88, 188, "Bur", "Bur");

            Athlete ath17 =       new Athlete(17, "Justin", "Gatlin",
"Male", "10/2/1982", 83, 185, "USA", "USA");

            Athlete ath18 =       new Athlete(18, "Jason", "Gardener",
"Male", "18/9/1975", 75, 176, "Great Britain", "Great Britain");

            Athlete ath19 =       new Athlete(19, "Uchenna", "Emedolu",
"Male", "17/9/1976", 68, 185, "Nigeria", "Nigeria");

            Athlete ath20 =       new Athlete(20, "Nobuharu", "Asahara",
"Male", "21/6/1972", 75, 179, "Japan", "Japan");

            Athlete ath21 =       new Athlete(21, "Yeoryios",
"Theodoridis", "Male", "12/12/1972", 80, 181, "Greece", "Greece");

            Athlete ath22 =       new Athlete(22, "Ronald", "Nemeth",
"Male", "19/9/1974", 80, 169, "Hungary", "Hungary");

            Athlete ath23=        new Athlete(23, "Nicconnor",
"Alexander", "Male", "4/2/1977", 70, 188, "Trinidad", "Trinidad");

            Athlete ath24 =       new Athlete(24, "Eddy", "De Lepine",
"Male", "30/3/1984", 74, 175, "France", "France");

            Athlete ath25 =       new Athlete(25, "Maurice", "Greene",
"Male", "23/7/1974", 81, 175, "USA", "USA");

            Athlete ath26 =       new Athlete(26, "Asafa", "Powell",
"Male", "11/11/1982", 88, 190, "Jamaica", "Jamaica");

            Athlete ath27 =       new Athlete(27, "Leonard", "Myles
Mills", "Male", "9/5/1973", 70, 175, "Ghana", "Ghana");
```

```
            Athlete ath28 =       new Athlete(28, "Lukasz", "Chyla",
"Male", "31/3/1981", 86, 188, "Poland", "Poland");

            Athlete ath29 =       new Athlete(29, "Kareem", "Streete
Thompson", "Male", "30/3/1973", 91, 183, "USA", "CAY");

            Athlete ath30 =       new Athlete(30, "Simone", "Collio",
"Male", "27/12/1979", 74, 179, "Italy", "Italy");

            Athlete ath31 =       new Athlete(31, "Jarbas",
"Mascarenhas", "Male", "25/8/1980", 78, 185, "Brazil", "Brazil");

            Athlete ath32 =       new Athlete(32, "Eric Pacome", "N Dri",
"Male", "1/1/1978", 78, 185, "Ivory Coast", "Ivory Coast");

            Athlete ath33 =       new Athlete(33, "Andreas",
"Thorkildsen", "Male", "1/4/1982", 88, 188, "Norway", "Norway");

            Athlete ath34 =       new Athlete(34, "Vadims",
"Vasilevskis", "Male", "5/1/1982", 89, 188, "Latvia", "Latvia");

            Athlete ath35 =       new Athlete(35, "Sergey", "Makarov",
"Male", "19/3/1973", 102, 190, "Russia", "Russia");
            ath35.setTrainer(tr1);

            Athlete ath36 =       new Athlete(36, "Steve", "Backley",
"Male", "12/2/1969", 102, 195, "Great Britain", "Great Britain");
            ath36.setTrainer(tr2);

            Athlete ath37 =       new Athlete(37, "Alexander", "Ivanov",
"Male", "25/5/1982", 95, 189, "Russia", "Russia");

            Athlete ath38 =       new Athlete(38, "Andrus", "Varnic",
"Male", "27/9/1977", 100, 182, "Estonia", "Estonia");
            ath38.setTrainer(tr3);

            Athlete ath39 =       new Athlete(39, "Eriks", "Rags",
"Male", "1/6/1975", 95, 183, "Latvia", "Latvia");
            ath39.setTrainer(tr4);

            Athlete ath40 =       new Athlete(40, "Tero", "Pitkamaki",
"Male", "19/12/1982", 90, 195, "Finland", "Finland");

            Athlete ath41 =       new Athlete(41, "Jan", "Zelezny",
"Male", "16/6/1966", 86, 185, "Czech", " Czech");
            ath41.setTrainer(tr14);

            Athlete ath42 =       new Athlete(42, "Matti", "Narhi",
"Male", "17/8/1975", 100, 188, "Finland", "Finland");

            Athlete ath43 =       new Athlete(43, "Esko", "Mikkola",
"Male", "14/2/1975", 95, 185, "Finland", "Finland");
            ath43.setTrainer(tr5);

            Athlete ath44 =       new Athlete(44, "Breaux", "Greer",
"Male", "19/10/1976", 92, 190, "USA", "USA");
            ath44.setTrainer(tr6);

            Athlete ath45 =       new Athlete(45, "Gerhardus", "Pienaar",
"Male", "10/8/1981", 104, 194, "South Africa", "South Africa");
            ath45.setTrainer(tr6);
```

```
            Athlete ath46 =       new Athlete(46, "Christian", "Nicolay",
"Male", "4/3/1976", 106, 188, "Germany", "Germany");
            ath46.setTrainer(tr8);

            Athlete ath47 =       new Athlete(47, "Voldemars", "Lusis",
"Male", "7/12/1974", 102, 187, "Latvia", "Latvia");
            ath47.setTrainer(tr9);

            Athlete ath48 =       new Athlete(48, "William", "Hamlyn
Harris", "Male", "14/1/1978", 96, 191, "Austria", "Austria");
            ath48.setTrainer(tr10);

            Athlete ath49 =       new Athlete(49, "Peter", "Zupanc",
"Male", "8/1/1982", 90, 184, "Slovenia", "Slovenia");

            Athlete ath50 =       new Athlete(50, "Miroslav", "Guzdek",
"Male", "3/8/1975", 93, 191, "Czech", "Czech");
            ath50.setTrainer(tr14);

            Athlete ath51 =       new Athlete(51, "Gergely", "Horvath",
"Male", "5/6/1975", 100, 186, "Hungary", "Hungary");

            Athlete ath52 =       new Athlete(52, "Ronny", "Nilsen",
"Male", "7/5/1971", 90, 182, "Norway", "Norway");

            Athlete ath53 =       new Athlete(53, "Manuel", "Fuenmayor",
"Male", "3/12/1980", 105, 190, "Venezuela", "Venezuela");

            Athlete ath54 =       new Athlete(54, "David", "Brisseault",
"Male", "7/3/1969", 95, 185, "France", "France");

            Athlete ath55 =       new Athlete(55, "Marian", "Bokor",
"Male", "17/4/1977", 98, 196, "Czech", "Czech");

            Athlete ath56 =       new Athlete(56, "Isbel", "Luaces",
"Male", "20/7/1975", 99, 184, "Cuba", "Cuba");

            Athlete ath57 =       new Athlete(57, "Rongxlang", "Li",
"Male", "18/1/1972", 86, 180, "China", "China");

            Athlete ath58 =       new Athlete(58, "Yukifumi", "Murakami",
"Male", "23/12/1979", 95, 184, "Japan", "Japan");

            Athlete ath59 =       new Athlete(59, "Oliver", "Dziubak",
"Male", "30/3/1982", 101, 197, "Germany", "Austria");
            ath59.setTrainer(tr12);

            Athlete ath60 =       new Athlete(60, "Peter", "Esenwein",
"Male", "7/12/1967", 102, 189, "Germany", "Germany");
            ath60.setTrainer(tr13);

            Athlete ath61 =       new Athlete(61, "Sergey", "Voynov",
"Male", "26/2/1977", 100, 188, "Russia", "Uzbekistan");

            Athlete ath62 =       new Athlete(62, "Stuart", "Farquhar",
"Male", "15/3/1982", 105, 186, "New Zealand", "New Zealand");

            Athlete ath63 =       new Athlete(63, "Nick", "Nieland",
"Male", "31/1/1972", 100, 190, "Great Britain", "Great Britain");
            ath63.setTrainer(tr2);
```

```
            Athlete ath64 =       new Athlete(64, "Jae Myong", "Park",
"Male", "15/12/1981", 92, 181, "Korea", "Korea");

            Athlete ath65 =       new Athlete(65, "Edi", "Ponos", "Male",
"10/4/1976", 118, 196, "Croatia", "Croatia");

            Athlete ath66 =       new Athlete(66, "Boris", "Henry",
"Male", "14/12/1973", 110, 193, "Germany", "Germany");
            ath66.setTrainer(tr15);

            Athlete ath67 =       new Athlete(67, "Osleidys", "Menendez",
"Female", "14/11/1979", 80, 179, "Cuba", "Cuba");
            ath67.setTrainer(tr16);

            Athlete ath68 =       new Athlete(68, "Steffi", "Nerius",
"Female", "1/7/1972", 72, 178, "Germany", "Germany");
            ath68.setTrainer(tr18);

            Athlete ath69 =       new Athlete(69, "Mirela", "Manjani",
"Female", "21/12/1976", 64, 164, "Greece", "Greece");
            ath69.setTrainer(tr17);

            Athlete ath70 =       new Athlete(70, "Sonia", "Bicet Poll",
"Female", "1/4/1971", 83, 171, "Cuba", "Cuba");

            Athlete ath71 =       new Athlete(71, "Lavern", "Eve",
"Female", "16/6/1965", 81, 180, "Bahamas", "Bahamas");

            Athlete ath72 =       new Athlete(72, "Noraida", "Bicet",
"Female", "29/10/1977", 87, 180, "Cuba", "Cuba");

            Athlete ath73 =       new Athlete(73, "Tetyana",
"Lyakhovych", "Female", "20/5/1979", 81, 178, "Ukraine", "Ukraine");

            Athlete ath74 =       new Athlete(74, "Savva", "Lika",
"Female", "27/6/1970", 70, 168, "Greece", "Greece");
            ath74.setTrainer(tr19);

            Athlete ath75 =       new Athlete(75, "Taina", "Kolkkala",
"Female", "24/10/1976", 73, 173, "Finland", "Finland");
            ath75.setTrainer(tr20);

            Athlete ath76 =       new Athlete(76, "Felicia", "Tilea
Moldovan", "Female", "29/9/1967", 74, 167, "Romania", "Romania");

            Athlete ath77 =       new Athlete(77, "Barbara", "Madejczyk",
"Female", "30/9/1976", 80, 180, "Poland", "Poland");
            ath77.setTrainer(tr21);

            Athlete ath78 =       new Athlete(78, "Francis", "Obikwelu",
"Male", "22/11/1978", 74, 191, "Nigeria", "Portugal");

////////////////////////// END OF STORING ATHLETES
//////////////////////////////

////////////////////// ADD ATHLETES TO TRAINER'S LIST
//////////////////////////

            tr1.addAthlete(ath35);
            tr2.addAthlete(ath36);
            tr2.addAthlete(ath63);
            tr3.addAthlete(ath38);
```

```
            tr4.addAthlete(ath39);
            tr14.addAthlete(ath41);
            tr14.addAthlete(ath50);
            tr5.addAthlete(ath43);
            tr6.addAthlete(ath44);
            tr6.addAthlete(ath45);
            tr8.addAthlete(ath46);
            tr9.addAthlete(ath47);
            tr10.addAthlete(ath48);
            tr22.addAthlete(ath50);
            tr12.addAthlete(ath59);
            tr13.addAthlete(ath60);
            tr15.addAthlete(ath66);
            tr16.addAthlete(ath67);
            tr18.addAthlete(ath68);
            tr17.addAthlete(ath69);
            tr19.addAthlete(ath74);
            tr20.addAthlete(ath75);
            tr21.addAthlete(ath77);

//////////////////// END OF FILLING THE LIST
////////////////////////////////////

//////////////////////////// STORING TRAINERS TO DATABASE
////////////////////////

            db.store(tr1);
            db.store(tr4);
            db.store(tr5);
            db.store(tr6);
            db.store(tr7);
            db.store(tr8);
            db.store(tr9);
            db.store(tr10);
            db.store(tr11);
            db.store(tr2);
            db.store(tr3);
            db.store(tr12);
            db.store(tr13);
            db.store(tr14);
            db.store(tr15);
            db.store(tr16);
            db.store(tr17);
            db.store(tr18);
            db.store(tr19);
            db.store(tr20);
            db.store(tr21);
            db.store(tr22);

////////////////////////////// END OF STORING
////////////////////////////////////

////////////////////////////// CREATING STADIUMS
////////////////////////////////

            Stadium st1 = new Stadium("Olympic Stadium", "Athens",
                    72000, 14.5, 600, 120);

            Stadium st2 = new Stadium("Agios Kosmas Sailing Centre",
                    "Agios Kosmas", 1600, 34.7, 150, 25);
```

```
            Stadium st3 = new Stadium("Kaftanzoglio Stadium",
"Thessaloniki",
                    28200, 12, 151, 27);

            Stadium st4 = new Stadium("Pankritio Stadium", "Heraklio",
                    26400, 23, 138, 0);

            Stadium st5 = new Stadium("Panthessaliko Stadium", "Volos",
                    22700, 22, 133, 0);

////////////////////////////////// END OF STADIUMS
//////////////////////////////

//////////////////////////// CREATING SPORTS
////////////////////////////////
            Sport sp1 = new Sport(1, "Javelin", "Female", 71.54,
"1/7/2001", "Hattestad", "Trine");

            Sport sp2 = new Sport(2, "Javelin", "Male", 90.17,
"23/9/2000", "Zelezny", "Jan");

            Sport sp3 = new Sport(3, "100m", "Male", 9.84, "27/7/1996",
"Bailey", "Donovan");

//////////////////////////// END OF SPORTS
////////////////////////////////


//////////////////////////// CREATING GAMES
////////////////////////////////

            Games g1 = new Games(1, "Final", "27/8/2004", "20:55", sp1,
st1);
            Games g2 = new Games(2, "Round Group A", "26/8/2004",
"20:05", sp2, st1);
            Games g3 = new Games(3, "Round Group B", "26/8/2004",
"21:50", sp2, st1);
            Games g4 = new Games(4, "Final", "28/8/2004", "20:40", sp2,
st2);
            Games g5 = new Games(5, "Round1", "21/8/2004", "19:40",
sp3,st2);
            Games g6 = new Games(6, "Round1", "21/8/2004", "19:47",
sp3,st3);
            Games g7 = new Games(7, "Round1", "21/8/2004", "19:54",
sp3,st4);
            Games g8 = new Games(8, "Round1", "21/8/2004", "20:01",
sp3,st4);
            Games g9 = new Games(9, "Round1", "21/8/2004", "20:08",
sp3,st4);
            Games g10 = new Games(10, "Semifinal", "22/8/2004", "20:55",
sp3,st4);
            Games g11 = new Games(11, "Semifinal", "22/8/2004", "21:03",
sp3,st5);
            Games g12 = new Games(12, "Final", "22/8/2004", "23:10",
sp3,st5);

//////////////////////////////// END OF GAMES
////////////////////////////////////

////////////////////// ADDING GAMES TO SPORT LIST
//////////////////////////////
```

```
            sp1.addGame(g1);
            sp2.addGame(g2);
            sp2.addGame(g3);
            sp2.addGame(g4);
            sp3.addGame(g5);
            sp3.addGame(g6);
            sp3.addGame(g7);
            sp3.addGame(g8);
            sp3.addGame(g9);
            sp3.addGame(g10);
            sp3.addGame(g11);
            sp3.addGame(g12);

//////////////////////////// END OF ADDING
//////////////////////////////////////

//////////////////////////// STORING SPORTS
//////////////////////////////////////

            db.store(sp1);
            db.store(sp2);
            db.store(sp3);

//////////////////////////// END OF STORING
//////////////////////////////////////

////////////////////////// ADDING GAMES TO STADIUM LIST
////////////////////////////

            st1.addGame(g1);
            st1.addGame(g2);
            st1.addGame(g3);
            st2.addGame(g4);
            st2.addGame(g5);
            st3.addGame(g6);
            st4.addGame(g7);
            st4.addGame(g8);
            st4.addGame(g9);
            st4.addGame(g10);
            st5.addGame(g11);
            st5.addGame(g12);

////////////////////////// END OF ADDING
//////////////////////////////////////////

//////////////////////////////// STORING STADIUMS
//////////////////////////////////

            db.store(st1);
            db.store(st2);
            db.store(st3);
            db.store(st4);
            db.store(st5);

//////////////////////////// END OF STORING
//////////////////////////////////////

//////////////////////////////// STORING GAMES
//////////////////////////////////

            db.store(g1);
            db.store(g2);
```

```
            db.store(g3);
            db.store(g4);
            db.store(g5);
            db.store(g6);
            db.store(g7);
            db.store(g8);
            db.store(g9);
            db.store(g10);
            db.store(g11);
            db.store(g12);

//////////////////////////// END OF STORING
////////////////////////////////////

//////////////////////////// CREATING PARTICIPATIONS
////////////////////////////

            Participates prt1=  new Participates(g4, ath33, 86.50,
"Q",1);
            Participates prt2= new Participates(g4, ath34, 84.95, "Q",2);
            Participates prt3= new Participates(g4, ath35, 84.84, "Q",3);
            Participates prt4= new Participates(g4, ath36, 84.13, "Q",4);
            Participates prt5= new Participates(g4, ath37, 83.31, "Q",5);
            Participates prt6= new Participates(g4, ath38, 83.25, "Q",6);
            Participates prt7= new Participates(g4, ath39, 83.14, "Q",7);
            Participates prt8= new Participates(g4, ath40, 83.01, "Q",8);
            Participates prt9= new Participates(g4, ath41, 80.59, "Q",9);
            Participates prt10= new Participates(g4, ath42, 80.28,
"Q",10);
            Participates prt11= new Participates(g4, ath43, 79.43,
"Q",11);
            Participates prt12= new Participates(g4, ath44, 74.36,
"Q",12);
            Participates prt13= new Participates(g3, ath34, 84.43,
"Q",13);
            Participates prt14= new Participates(g3, ath37, 82.18,
"Q",14);
            Participates prt15= new Participates(g3, ath40, 82.04,
"Q",15);
            Participates prt16= new Participates(g3, ath33, 81.74,
"Q",16);
            Participates prt17= new Participates(g3, ath41, 81.18,
"Q",17);
            Participates prt18= new Participates(g3, ath39, 80.84,
"Q",18);
            Participates prt19= new Participates(g3, ath56, 80.07,
"Q",19);
            Participates prt20= new Participates(g3, ath57, 79.94,
"Q",20);
            Participates prt21= new Participates(g3, ath58, 78.59,
"Q",21);
            Participates prt22= new Participates(g3, ath59, 78.53,
"Q",22);
            Participates prt23= new Participates(g3, ath60, 78.41,
"Q",23);
            Participates prt24= new Participates(g3, ath61, 74.68,
"Q",24);
            Participates prt25= new Participates(g3, ath62, 74.63,
"Q",25);
            Participates prt26= new Participates(g3, ath63, 72.79,
"Q",26);
```

```
            Participates prt27= new Participates(g3, ath64, 72.70,
"Q",27);
            Participates prt28= new Participates(g3, ath65, 71.43,
"Q",28);
            Participates prt29= new Participates(g3, ath66, 0, "D",29);
            Participates prt30= new Participates(g2, ath44, 87.25,
"Q",30);
            Participates prt31= new Participates(g2, ath35, 86.08,
"Q",31);
            Participates prt32= new Participates(g2, ath43, 83.64,
"Q",32);
            Participates prt33= new Participates(g2, ath38, 83.25,
"Q",33);
            Participates prt34= new Participates(g2, ath42, 81.06,
"Q",34);
            Participates prt35= new Participates(g2, ath36, 80.68,
"Q",35);
            Participates prt36= new Participates(g2, ath45, 79.95,
"Q",36);
            Participates prt37= new Participates(g2, ath46, 79.77,
"Q",37);
            Participates prt38= new Participates(g2, ath47, 79.27,
"Q",38);
            Participates prt39= new Participates(g2, ath48, 77.43,
"Q",39);
            Participates prt40= new Participates(g2, ath49, 77.34,
"Q",40);
            Participates prt41= new Participates(g2, ath50, 76.45,
"Q",41);
            Participates prt42= new Participates(g2, ath51, 73.95,
"Q",42);
            Participates prt43= new Participates(g2, ath52, 73.46,
"Q",43);
            Participates prt44= new Participates(g2, ath53, 72.26,
"Q",44);
            Participates prt45= new Participates(g2, ath54, 71.86,
"Q",45);
            Participates prt46= new Participates(g2, ath55, 71.74,
"Q",46);
            Participates prt47= new Participates(g1, ath67, 71.53,
"Q",47);
            Participates prt48= new Participates(g1, ath68, 65.82,
"Q",48);
            Participates prt49= new Participates(g1, ath70, 63.54,
"Q",49);
            Participates prt50= new Participates(g1, ath71, 62.77,
"Q",50);
            Participates prt51= new Participates(g1, ath72, 62.51,
"Q",51);
            Participates prt52= new Participates(g1, ath73, 61.75,
"Q",52);
            Participates prt53= new Participates(g1, ath74, 60.91,
"Q",53);
            Participates prt54= new Participates(g1, ath75, 60.72,
"Q",54);
            Participates prt55= new Participates(g1, ath76, 59.72,
"Q",55);
            Participates prt56= new Participates(g1, ath77, 58.22,
"Q",56);
            Participates prt57= new Participates(g6, ath9, 9.89, "Q",57);
            Participates prt58= new Participates(g6, ath10, 10.12,
"Q",58);
```

```
            Participates prt59= new Participates(g6, ath11, 10.26,
"Q",59);
            Participates prt60= new Participates(g6, ath12, 10.26,
"Q",60);
            Participates prt61= new Participates(g6, ath13, 10.26,
"Q",61);
            Participates prt62= new Participates(g6, ath14, 10.28,
"Q",62);
            Participates prt63= new Participates(g6, ath15, 10.42,
"Q",63);
            Participates prt64= new Participates(g6, ath16, 10.43,
"Q",64);
            Participates prt65= new Participates(g7, ath17, 9.96,
"Q",65);
            Participates prt66= new Participates(g7, ath18, 10.15,
"Q",66);
            Participates prt67= new Participates(g7, ath19, 10.15,
"Q",67);
            Participates prt68= new Participates(g7, ath20, 10.24,
"Q",68);
            Participates prt69= new Participates(g7, ath21, 10.36,
"Q",69);
            Participates prt70= new Participates(g7, ath22, 10.38,
"Q",70);
            Participates prt71= new Participates(g7, ath23, 10.48,
"Q",71);
            Participates prt72= new Participates(g7, ath24, 0,  "D",72);
            Participates prt73= new Participates(g8, ath2, 10.05,
"Q",73);
            Participates prt74= new Participates(g8, ath3, 10.11,
"Q",74);
            Participates prt75= new Participates(g8, ath4, 10.17,
"Q",75);
            Participates prt76= new Participates(g8, ath5, 10.22,
"Q",76);
            Participates prt77= new Participates(g8, ath6, 10.24,
"Q",77);
            Participates prt78= new Participates(g8, ath7, 10.29,
"Q",78);
            Participates prt79= new Participates(g8, ath8, 10.39,
"Q",79);
            Participates prt80= new Participates(g9, ath25, 9.93,
"Q",80);
            Participates prt81= new Participates(g9, ath26, 9.99,
"Q",81);
            Participates prt82= new Participates(g9, ath27, 10.18,
"Q",82);
            Participates prt83= new Participates(g9, ath28, 10.23,
"Q",83);
            Participates prt84= new Participates(g9, ath29, 10.24,
"Q",84);
            Participates prt85= new Participates(g9, ath30, 10.29,
"Q",85);
            Participates prt86= new Participates(g9, ath31, 10.30,
"Q",86);
            Participates prt87= new Participates(g9, ath32, 10.32,
"Q",87);
            Participates prt88= new Participates(g10, ath9, 10.07,
"Q",88);
            Participates prt89= new Participates(g10, ath17, 10.09,
"Q",89);
```

```
            Participates prt90= new Participates(g10, ath1, 10.11,
"Q",90);
            Participates prt91= new Participates(g10, ath10, 10.22,
"Q",91);
            Participates prt92= new Participates(g10, ath3, 10.29,
"Q",92);
            Participates prt93= new Participates(g10, ath19, 10.35,
"Q",93);
            Participates prt94= new Participates(g12, ath17, 9.85,
"Q",94);
            Participates prt95= new Participates(g12, ath25, 9.87,
"Q",95);
            Participates prt96= new Participates(g12, ath9, 9.89,
"Q",96);
            Participates prt97= new Participates(g12, ath26, 9.94,
"Q",97);
            Participates prt98= new Participates(g12, ath2, 10.00,
"Q",98);
            Participates prt99= new Participates(g12, ath10, 10.10,
"Q",99);
            Participates prt100= new Participates(g12, ath1, 0, "D",100);
            Participates prt101= new Participates(g12, ath78, 9.86,
"Q",101);

////////////////////////////// END OF CREATING
//////////////////////////////////

////////////////////////////// STORING PARTICIPATIONS
//////////////////////////

            db.store(prt1);
            db.store(prt2);
            db.store(prt3);
            db.store(prt4);
            db.store(prt5);
            db.store(prt6);
            db.store(prt7);
            db.store(prt8);
            db.store(prt9);
            db.store(prt10);
            db.store(prt11);
            db.store(prt12);
            db.store(prt13);
            db.store(prt14);
            db.store(prt15);
            db.store(prt16);
            db.store(prt17);
            db.store(prt18);
            db.store(prt19);
            db.store(prt20);
            db.store(prt21);
            db.store(prt22);
            db.store(prt23);
            db.store(prt24);
            db.store(prt25);
            db.store(prt26);
            db.store(prt27);
            db.store(prt28);
            db.store(prt29);
            db.store(prt30);
            db.store(prt31);
            db.store(prt32);
```

```
db.store(prt33);
db.store(prt34);
db.store(prt35);
db.store(prt36);
db.store(prt37);
db.store(prt38);
db.store(prt39);
db.store(prt40);
db.store(prt41);
db.store(prt42);
db.store(prt43);
db.store(prt44);
db.store(prt45);
db.store(prt46);
db.store(prt47);
db.store(prt48);
db.store(prt49);
db.store(prt50);
db.store(prt51);
db.store(prt52);
db.store(prt53);
db.store(prt54);
db.store(prt55);
db.store(prt56);
db.store(prt57);
db.store(prt58);
db.store(prt59);
db.store(prt60);
db.store(prt61);
db.store(prt62);
db.store(prt63);
db.store(prt64);
db.store(prt65);
db.store(prt66);
db.store(prt67);
db.store(prt68);
db.store(prt69);
db.store(prt70);
db.store(prt71);
db.store(prt72);
db.store(prt73);
db.store(prt74);
db.store(prt75);
db.store(prt76);
db.store(prt77);
db.store(prt78);
db.store(prt79);
db.store(prt80);
db.store(prt81);
db.store(prt82);
db.store(prt83);
db.store(prt84);
db.store(prt85);
db.store(prt86);
db.store(prt87);
db.store(prt88);
db.store(prt89);
db.store(prt90);
db.store(prt91);
db.store(prt92);
db.store(prt93);
db.store(prt94);
```

```
            db.store(prt95);
            db.store(prt96);
            db.store(prt97);
            db.store(prt98);
            db.store(prt99);
            db.store(prt100);
            db.store(prt101);

//////////////////////////// END OF STORING
////////////////////////////////////

/////////////////// ADDING PARTICIPATIONS TO ATHLETE LIST
//////////////////////

            ath33.addPartic(prt1);
            ath33.addPartic(prt2);
            ath34.addPartic(prt3);
            ath34.addPartic(prt4);
            ath35.addPartic(prt5);
            ath35.addPartic(prt6);
            ath36.addPartic(prt7);
            ath36.addPartic(prt8);
            ath37.addPartic(prt9);
            ath37.addPartic(prt10);
            ath38.addPartic(prt11);
            ath38.addPartic(prt12);
            ath39.addPartic(prt13);
            ath39.addPartic(prt14);
            ath40.addPartic(prt15);
            ath40.addPartic(prt16);
            ath41.addPartic(prt17);
            ath41.addPartic(prt18);
            ath42.addPartic(prt19);
            ath42.addPartic(prt20);
            ath43.addPartic(prt21);
            ath43.addPartic(prt22);
            ath44.addPartic(prt23);
            ath44.addPartic(prt24);
            ath56.addPartic(prt25);
            ath57.addPartic(prt26);
            ath58.addPartic(prt27);
            ath59.addPartic(prt28);
            ath60.addPartic(prt29);
            ath61.addPartic(prt30);
            ath62.addPartic(prt31);
            ath63.addPartic(prt32);
            ath64.addPartic(prt33);
            ath65.addPartic(prt34);
            ath66.addPartic(prt35);
            ath45.addPartic(prt36);
            ath46.addPartic(prt37);
            ath47.addPartic(prt38);
            ath48.addPartic(prt39);
            ath49.addPartic(prt40);
            ath50.addPartic(prt41);
            ath51.addPartic(prt42);
            ath52.addPartic(prt43);
            ath53.addPartic(prt44);
            ath54.addPartic(prt45);
            ath55.addPartic(prt46);
            ath67.addPartic(prt47);
            ath68.addPartic(prt48);
```

```
            ath70.addPartic(prt49);
            ath71.addPartic(prt50);
            ath72.addPartic(prt51);
            ath73.addPartic(prt52);
            ath74.addPartic(prt53);
            ath75.addPartic(prt54);
            ath76.addPartic(prt55);
            ath77.addPartic(prt56);
            ath9.addPartic(prt57);
            ath9.addPartic(prt58);
            ath9.addPartic(prt59);
            ath10.addPartic(prt60);
            ath10.addPartic(prt61);
            ath10.addPartic(prt62);
            ath11.addPartic(prt63);
            ath12.addPartic(prt64);
            ath13.addPartic(prt65);
            ath14.addPartic(prt66);
            ath15.addPartic(prt67);
            ath16.addPartic(prt68);
            ath17.addPartic(prt69);
            ath17.addPartic(prt70);
            ath17.addPartic(prt71);
            ath18.addPartic(prt72);
            ath19.addPartic(prt73);
            ath19.addPartic(prt74);
            ath20.addPartic(prt75);
            ath21.addPartic(prt76);
            ath22.addPartic(prt77);
            ath23.addPartic(prt78);
            ath24.addPartic(prt79);
            ath2.addPartic(prt80);
            ath2.addPartic(prt81);
            ath3.addPartic(prt82);
            ath3.addPartic(prt83);
            ath4.addPartic(prt84);
            ath5.addPartic(prt85);
            ath6.addPartic(prt86);
            ath7.addPartic(prt87);
            ath8.addPartic(prt88);
            ath25.addPartic(prt89);
            ath25.addPartic(prt90);
            ath26.addPartic(prt91);
            ath26.addPartic(prt92);
            ath27.addPartic(prt93);
            ath28.addPartic(prt94);
            ath29.addPartic(prt95);
            ath30.addPartic(prt96);
            ath31.addPartic(prt97);
            ath32.addPartic(prt98);
            ath1.addPartic(prt99);
            ath1.addPartic(prt100);
            ath78.addPartic(prt101);

//////////////////////// END OF ADDING
//////////////////////////////////////

//////////////////////// STORING ATHLETES
//////////////////////////////////////

      db.store(ath1);
      db.store(ath2);
```

```
db.store(ath3);
db.store(ath4);
db.store(ath5);
db.store(ath6);
db.store(ath7);
db.store(ath8);
db.store(ath9);
db.store(ath10);
db.store(ath11);
db.store(ath12);
db.store(ath13);
db.store(ath14);
db.store(ath15);
db.store(ath16);
db.store(ath17);
db.store(ath18);
db.store(ath19);
db.store(ath20);
db.store(ath21);
db.store(ath22);
db.store(ath23);
db.store(ath24);
db.store(ath25);
db.store(ath26);
db.store(ath27);
db.store(ath28);
db.store(ath29);
db.store(ath30);
db.store(ath31);
db.store(ath32);
db.store(ath33);
db.store(ath34);
db.store(ath35);
db.store(ath36);
db.store(ath37);
db.store(ath38);
db.store(ath39);
db.store(ath40);
db.store(ath41);
db.store(ath42);
db.store(ath43);
db.store(ath44);
db.store(ath45);
db.store(ath46);
db.store(ath47);
db.store(ath48);
db.store(ath49);
db.store(ath50);
db.store(ath51);
db.store(ath52);
db.store(ath53);
db.store(ath54);
db.store(ath55);
db.store(ath56);
db.store(ath57);
db.store(ath58);
db.store(ath59);
db.store(ath60);
db.store(ath61);
db.store(ath62);
db.store(ath63);
db.store(ath64);
```

```
        db.store(ath65);
        db.store(ath66);
        db.store(ath67);
        db.store(ath68);
        db.store(ath69);
        db.store(ath70);
        db.store(ath71);
        db.store(ath72);
        db.store(ath73);
        db.store(ath74);
        db.store(ath75);
        db.store(ath76);
        db.store(ath77);
        db.store(ath78);

//////////////////////////// END OF STORING
//////////////////////////////////////
                } finally{
                    db.close();
                }
    }
        public static void accessDb4o() {
            ObjectContainer db = Db4oEmbedded.openFile(Db4oEmbedded
            .newConfiguration(), DB4OFILENAME);
                try {
// do something with db4o
        } finally {
            db.close();
        }
    }


       public static void listResult(List<?> result){
            System.out.println("Results number: " + result.size());
            for (Object o : result) {
                System.out.println(o);
    }
}
}
```

## Ερωτήματα στο db4o

```
       // retrieve all trainers -query by example-
        ObjectSet result = db.queryByExample(Trainer.class);
        listResult(result);

        // retrieve athletes where trainers country of origin = USA
        List <Athlete> athletes = db.query(new Predicate<Athlete>(){
                @Override
             public boolean match(Athlete athlete){
                return
athlete.getTrainer().getCountry_of_origin().equals("USA");
             }
         });
         listResult(athletes);

        // retrieve athletes where  participated in a Final
         List<Athlete> athletes2 = db.query(new Predicate<Athlete>(){
                @Override
            public boolean match(Athlete athlete){
                List lista = athlete.getPartic();
                for(Object obj : lista){
                    Participates P = (Participates)obj;
                    if(P.game.getGameslevel().equals("Final"))
                        return true;
                }
                    return false;
            }
        });
        listResult(athletes2);


         // retrieveAllPersonQBE
         Person proto = new Person(null, null, null);
             ObjectSet result = db.queryByExample(proto);
         listResult(result);

         // retrieveAllPersons
         ObjectSet result = db.queryByExample(Person.class);
         listResult(result);

         // retrievePersonByName
         Person proto = new Person("Michael", null, null);
             ObjectSet result = db.queryByExample(proto);
         listResult(result);
```

## Εξαγωγή δεδομένων σε XML

```
package db4o;
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;
import com.thoughtworks.xstream.XStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author michael
 */
public class XmlMain {
    public static void main(String [] args)throws IOException{
        ObjectContainer db = Db4oEmbedded.openFile("db4oTestDb.db4o");
     try {
        List p = new ArrayList();
        Person[] toArray = db.query(Person.class).toArray(new Person[0]);
        XStream xstream = new XStream();
        xstream.setMode(XStream.XPATH_ABSOLUTE_REFERENCES);
        xstream.alias("Person",Person.class);
        xstream.alias("Athlete",Athlete.class);
        xstream.alias("Trainer",Trainer.class);
        OutputStream stream = new FileOutputStream("hopefully.xml");
      try {
            xstream.toXML(toArray, stream);
        } finally {
            stream.close();
    }
} finally {
        db.close();
}

    }
}
```

## Αποτελέσματα σε XML

```
<Person-array>
  <Trainer>
    <name>Alexander</name>
    <surname>Makarov</surname>
    <country__of__origin>Russia</country__of__origin>
    <Trainercode>1</Trainercode>
    <athletes>
      <Athlete>
        <name>Sergey</name>
        <surname>Makarov</surname>
        <country__of__origin>Russia</country__of__origin>
        <trainer reference="/Person-array/Trainer"/>
        <Athletecode>35</Athletecode>
        <gender>Male</gender>
        <date__of__birth>19/3/1973</date__of__birth>
        <weight>102</weight>
        <height>190</height>
        <country__of__participation>Russia</country__of__participation>
        <partic/>
```

```
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer/athletes/Athlete"/>
<Trainer>
  <name>Maris</name>
  <surname>Griva</surname>
  <country__of__origin>Latvia</country__of__origin>
  <Trainercode>4</Trainercode>
  <athletes>
    <Athlete>
      <name>Eriks</name>
      <surname>Rags</surname>
      <country__of__origin>Latvia</country__of__origin>
      <trainer reference="/Person-array/Trainer[2]"/>
      <Athletecode>39</Athletecode>
      <gender>Male</gender>
      <date__of__birth>1/6/1975</date__of__birth>
      <weight>95</weight>
      <height>183</height>
      <country__of__participation>Latvia</country__of__participation>
      <partic/>
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer[2]/athletes/Athlete"/>
<Trainer>
  <name>Jukka</name>
  <surname>Manninen</surname>
  <country__of__origin>Finland</country__of__origin>
  <Trainercode>5</Trainercode>
  <athletes>
    <Athlete>
      <name>Esko</name>
      <surname>Mikkola</surname>
      <country__of__origin>Finland</country__of__origin>
      <trainer reference="/Person-array/Trainer[3]"/>
      <Athletecode>43</Athletecode>
      <gender>Male</gender>
      <date__of__birth>14/2/1975</date__of__birth>
      <weight>95</weight>
      <height>185</height>
      <country__of__participation>Finland</country__of__participation>
      <partic/>
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer[3]/athletes/Athlete"/>
<Trainer>
  <name>Don</name>
  <surname>Babbitt</surname>
  <country__of__origin>USA</country__of__origin>
  <Trainercode>6</Trainercode>
  <athletes>
    <Athlete>
      <name>Breaux</name>
      <surname>Greer</surname>
      <country__of__origin>USA</country__of__origin>
      <trainer reference="/Person-array/Trainer[4]"/>
      <Athletecode>44</Athletecode>
      <gender>Male</gender>
      <date__of__birth>19/10/1976</date__of__birth>
```

```
        <weight>92</weight>
        <height>190</height>
        <country__of__participation>USA</country__of__participation>
        <partic/>
      </Athlete>
      <Athlete>
        <name>Gerhardus</name>
        <surname>Pienaar</surname>
        <country__of__origin>South Africa</country__of__origin>
        <trainer reference="/Person-array/Trainer[4]"/>
        <Athletecode>45</Athletecode>
        <gender>Male</gender>
        <date__of__birth>10/8/1981</date__of__birth>
        <weight>104</weight>
        <height>194</height>
        <country__of__participation>South
Africa</country__of__participation>
        <partic/>
      </Athlete>
    </athletes>
  </Trainer>
  <Athlete reference="/Person-array/Trainer[4]/athletes/Athlete"/>
  <Athlete reference="/Person-array/Trainer[4]/athletes/Athlete[2]"/>
  <Trainer>
    <name>Terseus</name>
    <surname>Liebenberg</surname>
    <country__of__origin>South Africa</country__of__origin>
    <Trainercode>7</Trainercode>
    <athletes/>
  </Trainer>
  <Trainer>
    <name>Christian</name>
    <surname>Nicolay</surname>
    <country__of__origin>Germany</country__of__origin>
    <Trainercode>8</Trainercode>
    <athletes>
      <Athlete>
        <name>Christian</name>
        <surname>Nicolay</surname>
        <country__of__origin>Germany</country__of__origin>
        <trainer reference="/Person-array/Trainer[6]"/>
        <Athletecode>46</Athletecode>
        <gender>Male</gender>
        <date__of__birth>4/3/1976</date__of__birth>
        <weight>106</weight>
        <height>188</height>
        <country__of__participation>Germany</country__of__participation>
        <partic/>
      </Athlete>
    </athletes>
  </Trainer>
  <Athlete reference="/Person-array/Trainer[6]/athletes/Athlete"/>
  <Trainer>
    <name>Janis</name>
    <surname>Lusis</surname>
    <country__of__origin>Latvia</country__of__origin>
    <Trainercode>9</Trainercode>
    <athletes>
      <Athlete>
        <name>Voldemars</name>
        <surname>Lusis</surname>
        <country__of__origin>Latvia</country__of__origin>
```

```
        <trainer reference="/Person-array/Trainer[7]"/>
        <Athletecode>47</Athletecode>
        <gender>Male</gender>
        <date__of__birth>7/12/1974</date__of__birth>
        <weight>102</weight>
        <height>187</height>
        <country__of__participation>Latvia</country__of__participation>
        <partic/>
      </Athlete>
    </athletes>
  </Trainer>
  <Athlete reference="/Person-array/Trainer[7]/athletes/Athlete"/>
  <Trainer>
    <name>Rudolph</name>
    <surname>Sopko</surname>
    <country__of__origin>Austria</country__of__origin>
    <Trainercode>10</Trainercode>
    <athletes>
      <Athlete>
        <name>William</name>
        <surname>Hamlyn Harris</surname>
        <country__of__origin>Austria</country__of__origin>
        <trainer reference="/Person-array/Trainer[8]"/>
        <Athletecode>48</Athletecode>
        <gender>Male</gender>
        <date__of__birth>14/1/1978</date__of__birth>
        <weight>96</weight>
        <height>191</height>
        <country__of__participation>Austria</country__of__participation>
        <partic/>
      </Athlete>
    </athletes>
  </Trainer>
  <Athlete reference="/Person-array/Trainer[8]/athletes/Athlete"/>
  <Trainer>
    <name>Jaroslev</name>
    <surname>Brabec</surname>
    <country__of__origin>Czech</country__of__origin>
    <Trainercode>11</Trainercode>
    <athletes/>
  </Trainer>
  <Trainer>
    <name>John</name>
    <surname>Trower</surname>
    <country__of__origin>Great Britain</country__of__origin>
    <Trainercode>2</Trainercode>
    <athletes>
      <Athlete>
        <name>Steve</name>
        <surname>Backley</surname>
        <country__of__origin>Great Britain</country__of__origin>
        <trainer reference="/Person-array/Trainer[10]"/>
        <Athletecode>36</Athletecode>
        <gender>Male</gender>
        <date__of__birth>12/2/1969</date__of__birth>
        <weight>102</weight>
        <height>195</height>
        <country__of__participation>Great
Britain</country__of__participation>
        <partic/>
      </Athlete>
      <Athlete>
```

```
        <name>Nick</name>
        <surname>Nieland</surname>
        <country__of__origin>Great Britain</country__of__origin>
        <trainer reference="/Person-array/Trainer[10]"/>
        <Athletecode>63</Athletecode>
        <gender>Male</gender>
        <date__of__birth>31/1/1972</date__of__birth>
        <weight>100</weight>
        <height>190</height>
        <country__of__participation>Great
Britain</country__of__participation>
        <partic/>
      </Athlete>
    </athletes>
  </Trainer>
  <Athlete reference="/Person-array/Trainer[10]/athletes/Athlete"/>
  <Athlete reference="/Person-array/Trainer[10]/athletes/Athlete[2]"/>
  <Trainer>
    <name>Heino</name>
    <surname>Puuste</surname>
    <country__of__origin>Estonia</country__of__origin>
    <Trainercode>3</Trainercode>
    <athletes>
      <Athlete>
        <name>Andrus</name>
        <surname>Varnic</surname>
        <country__of__origin>Estonia</country__of__origin>
        <trainer reference="/Person-array/Trainer[11]"/>
        <Athletecode>38</Athletecode>
        <gender>Male</gender>
        <date__of__birth>27/9/1977</date__of__birth>
        <weight>100</weight>
        <height>182</height>
        <country__of__participation>Estonia</country__of__participation>
        <partic/>
      </Athlete>
    </athletes>
  </Trainer>
  <Athlete reference="/Person-array/Trainer[11]/athletes/Athlete"/>
  <Trainer>
    <name>Steve</name>
    <surname>Lemke</surname>
    <country__of__origin>Austria</country__of__origin>
    <Trainercode>12</Trainercode>
    <athletes>
      <Athlete>
        <name>Oliver</name>
        <surname>Dziubak</surname>
        <country__of__origin>Austria</country__of__origin>
        <trainer reference="/Person-array/Trainer[12]"/>
        <Athletecode>59</Athletecode>
        <gender>Male</gender>
        <date__of__birth>30/3/1982</date__of__birth>
        <weight>101</weight>
        <height>197</height>
        <country__of__participation>Germany</country__of__participation>
        <partic/>
      </Athlete>
    </athletes>
  </Trainer>
  <Athlete reference="/Person-array/Trainer[12]/athletes/Athlete"/>
  <Trainer>
```

```
  <name>Karin</name>
  <surname>Liebhardt</surname>
  <country__of__origin>Germany</country__of__origin>
  <Trainercode>13</Trainercode>
  <athletes>
    <Athlete>
      <name>Peter</name>
      <surname>Esenwein</surname>
      <country__of__origin>Germany</country__of__origin>
      <trainer reference="/Person-array/Trainer[13]"/>
      <Athletecode>60</Athletecode>
      <gender>Male</gender>
      <date__of__birth>7/12/1967</date__of__birth>
      <weight>102</weight>
      <height>189</height>
      <country__of__participation>Germany</country__of__participation>
      <partic/>
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer[13]/athletes/Athlete"/>
<Trainer>
  <name>Jaroslav</name>
  <surname>Brabec</surname>
  <country__of__origin>Czech</country__of__origin>
  <Trainercode>14</Trainercode>
  <athletes>
    <Athlete>
      <name>Jan</name>
      <surname>Zelezny</surname>
      <country__of__origin> Czech</country__of__origin>
      <trainer reference="/Person-array/Trainer[14]"/>
      <Athletecode>41</Athletecode>
      <gender>Male</gender>
      <date__of__birth>16/6/1966</date__of__birth>
      <weight>86</weight>
      <height>185</height>
      <country__of__participation>Czech</country__of__participation>
      <partic/>
    </Athlete>
    <Athlete>
      <name>Miroslav</name>
      <surname>Guzdek</surname>
      <country__of__origin>Czech</country__of__origin>
      <trainer reference="/Person-array/Trainer[14]"/>
      <Athletecode>50</Athletecode>
      <gender>Male</gender>
      <date__of__birth>3/8/1975</date__of__birth>
      <weight>93</weight>
      <height>191</height>
      <country__of__participation>Czech</country__of__participation>
      <partic/>
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer[14]/athletes/Athlete"/>
<Athlete reference="/Person-array/Trainer[14]/athletes/Athlete[2]"/>
<Trainer>
  <name>Klaus</name>
  <surname>Bartonietz</surname>
  <country__of__origin>Germany</country__of__origin>
  <Trainercode>15</Trainercode>
```

```
  <athletes>
    <Athlete>
      <name>Boris</name>
      <surname>Henry</surname>
      <country__of__origin>Germany</country__of__origin>
      <trainer reference="/Person-array/Trainer[15]"/>
      <Athletecode>66</Athletecode>
      <gender>Male</gender>
      <date__of__birth>14/12/1973</date__of__birth>
      <weight>110</weight>
      <height>193</height>
      <country__of__participation>Germany</country__of__participation>
      <partic/>
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer[15]/athletes/Athlete"/>
<Trainer>
  <name>Dionisio</name>
  <surname>Quintana</surname>
  <country__of__origin>Cuba</country__of__origin>
  <Trainercode>16</Trainercode>
  <athletes>
    <Athlete>
      <name>Osleidys</name>
      <surname>Menendez</surname>
      <country__of__origin>Cuba</country__of__origin>
      <trainer reference="/Person-array/Trainer[16]"/>
      <Athletecode>67</Athletecode>
      <gender>Female</gender>
      <date__of__birth>14/11/1979</date__of__birth>
      <weight>80</weight>
      <height>179</height>
      <country__of__participation>Cuba</country__of__participation>
      <partic/>
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer[16]/athletes/Athlete"/>
<Trainer>
  <name>Helge</name>
  <surname>Zoellkau</surname>
  <country__of__origin>Germany</country__of__origin>
  <Trainercode>17</Trainercode>
  <athletes>
    <Athlete>
      <name>Mirela</name>
      <surname>Manjani</surname>
      <country__of__origin>Greece</country__of__origin>
      <trainer reference="/Person-array/Trainer[17]"/>
      <Athletecode>69</Athletecode>
      <gender>Female</gender>
      <date__of__birth>21/12/1976</date__of__birth>
      <weight>64</weight>
      <height>164</height>
      <country__of__participation>Greece</country__of__participation>
      <partic/>
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer[17]/athletes/Athlete"/>
<Trainer>
```

```
  <name>Vassilis</name>
  <surname>Kokkolis</surname>
  <country__of__origin>Greece</country__of__origin>
  <Trainercode>18</Trainercode>
  <athletes>
    <Athlete>
      <name>Steffi</name>
      <surname>Nerius</surname>
      <country__of__origin>Germany</country__of__origin>
      <trainer reference="/Person-array/Trainer[18]"/>
      <Athletecode>68</Athletecode>
      <gender>Female</gender>
      <date__of__birth>1/7/1972</date__of__birth>
      <weight>72</weight>
      <height>178</height>
      <country__of__participation>Germany</country__of__participation>
      <partic/>
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer[18]/athletes/Athlete"/>
<Trainer>
  <name>Yannis</name>
  <surname>Peristeris</surname>
  <country__of__origin>Greece</country__of__origin>
  <Trainercode>19</Trainercode>
  <athletes>
    <Athlete>
      <name>Savva</name>
      <surname>Lika</surname>
      <country__of__origin>Greece</country__of__origin>
      <trainer reference="/Person-array/Trainer[19]"/>
      <Athletecode>74</Athletecode>
      <gender>Female</gender>
      <date__of__birth>27/6/1970</date__of__birth>
      <weight>70</weight>
      <height>168</height>
      <country__of__participation>Greece</country__of__participation>
      <partic/>
    </Athlete>
  </athletes>
</Trainer>
<Athlete reference="/Person-array/Trainer[19]/athletes/Athlete"/>
<Trainer>
  <name>Kari</name>
  <surname>Ihalainen</surname>
  <country__of__origin>Finland</country__of__origin>
  <Trainercode>20</Trainercode>
  <athletes>
    <Athlete>
      <name>Taina</name>
      <surname>Kolkkala</surname>
      <country__of__origin>Finland</country__of__origin>
      <trainer reference="/Person-array/Trainer[20]"/>
      <Athletecode>75</Athletecode>
      <gender>Female</gender>
      <date__of__birth>24/10/1976</date__of__birth>
      <weight>73</weight>
      <height>173</height>
      <country__of__participation>Finland</country__of__participation>
      <partic/>
    </Athlete>
```

```
    </athletes>
  </Trainer>
  <Athlete reference="/Person-array/Trainer[20]/athletes/Athlete"/>
  <Trainer>
    <name>Henryk</name>
    <surname>Michalski</surname>
    <country__of__origin>Poland</country__of__origin>
    <Trainercode>21</Trainercode>
    <athletes>
      <Athlete>
        <name>Barbara</name>
        <surname>Madejczyk</surname>
        <country__of__origin>Poland</country__of__origin>
        <trainer reference="/Person-array/Trainer[21]"/>
        <Athletecode>77</Athletecode>
        <gender>Female</gender>
        <date__of__birth>30/9/1976</date__of__birth>
        <weight>80</weight>
        <height>180</height>
        <country__of__participation>Poland</country__of__participation>
        <partic/>
      </Athlete>
    </athletes>
  </Trainer>
  <Athlete reference="/Person-array/Trainer[21]/athletes/Athlete"/>
  <Trainer>
    <name>Dana</name>
    <surname>Zatopkova</surname>
    <country__of__origin>Czech</country__of__origin>
    <Trainercode>22</Trainercode>
    <athletes>
      <Athlete reference="/Person-
array/Trainer[14]/athletes/Athlete[2]"/>
    </athletes>
  </Trainer>
  <Athlete>
    <name>Andreas</name>
    <surname>Thorkildsen</surname>
    <country__of__origin>Norway</country__of__origin>
    <Athletecode>33</Athletecode>
    <gender>Male</gender>
    <date__of__birth>1/4/1982</date__of__birth>
    <weight>88</weight>
    <height>188</height>
    <country__of__participation>Norway</country__of__participation>
    <partic/>
  </Athlete>
  <Athlete>
    <name>Vadims</name>
    <surname>Vasilevskis</surname>
    <country__of__origin>Latvia</country__of__origin>
    <Athletecode>34</Athletecode>
    <gender>Male</gender>
    <date__of__birth>5/1/1982</date__of__birth>
    <weight>89</weight>
    <height>188</height>
    <country__of__participation>Latvia</country__of__participation>
    <partic/>
  </Athlete>
  <Athlete>
    <name>Alexander</name>
    <surname>Ivanov</surname>
```

```
  <country__of__origin>Russia</country__of__origin>
  <Athletecode>37</Athletecode>
  <gender>Male</gender>
  <date__of__birth>25/5/1982</date__of__birth>
  <weight>95</weight>
  <height>189</height>
  <country__of__participation>Russia</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Tero</name>
  <surname>Pitkamaki</surname>
  <country__of__origin>Finland</country__of__origin>
  <Athletecode>40</Athletecode>
  <gender>Male</gender>
  <date__of__birth>19/12/1982</date__of__birth>
  <weight>90</weight>
  <height>195</height>
  <country__of__participation>Finland</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Matti</name>
  <surname>Narhi</surname>
  <country__of__origin>Finland</country__of__origin>
  <Athletecode>42</Athletecode>
  <gender>Male</gender>
  <date__of__birth>17/8/1975</date__of__birth>
  <weight>100</weight>
  <height>188</height>
  <country__of__participation>Finland</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Isbel</name>
  <surname>Luaces</surname>
  <country__of__origin>Cuba</country__of__origin>
  <Athletecode>56</Athletecode>
  <gender>Male</gender>
  <date__of__birth>20/7/1975</date__of__birth>
  <weight>99</weight>
  <height>184</height>
  <country__of__participation>Cuba</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Rongxlang</name>
  <surname>Li</surname>
  <country__of__origin>China</country__of__origin>
  <Athletecode>57</Athletecode>
  <gender>Male</gender>
  <date__of__birth>18/1/1972</date__of__birth>
  <weight>86</weight>
  <height>180</height>
  <country__of__participation>China</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Yukifumi</name>
  <surname>Murakami</surname>
  <country__of__origin>Japan</country__of__origin>
  <Athletecode>58</Athletecode>
```

```
  <gender>Male</gender>
  <date__of__birth>23/12/1979</date__of__birth>
  <weight>95</weight>
  <height>184</height>
  <country__of__participation>Japan</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Sergey</name>
  <surname>Voynov</surname>
  <country__of__origin>Uzbekistan</country__of__origin>
  <Athletecode>61</Athletecode>
  <gender>Male</gender>
  <date__of__birth>26/2/1977</date__of__birth>
  <weight>100</weight>
  <height>188</height>
  <country__of__participation>Russia</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Stuart</name>
  <surname>Farquhar</surname>
  <country__of__origin>New Zealand</country__of__origin>
  <Athletecode>62</Athletecode>
  <gender>Male</gender>
  <date__of__birth>15/3/1982</date__of__birth>
  <weight>105</weight>
  <height>186</height>
  <country__of__participation>New Zealand</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Jae Myong</name>
  <surname>Park</surname>
  <country__of__origin>Korea</country__of__origin>
  <Athletecode>64</Athletecode>
  <gender>Male</gender>
  <date__of__birth>15/12/1981</date__of__birth>
  <weight>92</weight>
  <height>181</height>
  <country__of__participation>Korea</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Edi</name>
  <surname>Ponos</surname>
  <country__of__origin>Croatia</country__of__origin>
  <Athletecode>65</Athletecode>
  <gender>Male</gender>
  <date__of__birth>10/4/1976</date__of__birth>
  <weight>118</weight>
  <height>196</height>
  <country__of__participation>Croatia</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Peter</name>
  <surname>Zupanc</surname>
  <country__of__origin>Slovenia</country__of__origin>
  <Athletecode>49</Athletecode>
  <gender>Male</gender>
  <date__of__birth>8/1/1982</date__of__birth>
```

```
  <weight>90</weight>
  <height>184</height>
  <country__of__participation>Slovenia</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Gergely</name>
  <surname>Horvath</surname>
  <country__of__origin>Hungary</country__of__origin>
  <Athletecode>51</Athletecode>
  <gender>Male</gender>
  <date__of__birth>5/6/1975</date__of__birth>
  <weight>100</weight>
  <height>186</height>
  <country__of__participation>Hungary</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Ronny</name>
  <surname>Nilsen</surname>
  <country__of__origin>Norway</country__of__origin>
  <Athletecode>52</Athletecode>
  <gender>Male</gender>
  <date__of__birth>7/5/1971</date__of__birth>
  <weight>90</weight>
  <height>182</height>
  <country__of__participation>Norway</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Manuel</name>
  <surname>Fuenmayor</surname>
  <country__of__origin>Venezuela</country__of__origin>
  <Athletecode>53</Athletecode>
  <gender>Male</gender>
  <date__of__birth>3/12/1980</date__of__birth>
  <weight>105</weight>
  <height>190</height>
  <country__of__participation>Venezuela</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>David</name>
  <surname>Brisseault</surname>
  <country__of__origin>France</country__of__origin>
  <Athletecode>54</Athletecode>
  <gender>Male</gender>
  <date__of__birth>7/3/1969</date__of__birth>
  <weight>95</weight>
  <height>185</height>
  <country__of__participation>France</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Marian</name>
  <surname>Bokor</surname>
  <country__of__origin>Czech</country__of__origin>
  <Athletecode>55</Athletecode>
  <gender>Male</gender>
  <date__of__birth>17/4/1977</date__of__birth>
  <weight>98</weight>
  <height>196</height>
```

```
  <country__of__participation>Czech</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Sonia</name>
  <surname>Bicet Poll</surname>
  <country__of__origin>Cuba</country__of__origin>
  <Athletecode>70</Athletecode>
  <gender>Female</gender>
  <date__of__birth>1/4/1971</date__of__birth>
  <weight>83</weight>
  <height>171</height>
  <country__of__participation>Cuba</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Lavern</name>
  <surname>Eve</surname>
  <country__of__origin>Bahamas</country__of__origin>
  <Athletecode>71</Athletecode>
  <gender>Female</gender>
  <date__of__birth>16/6/1965</date__of__birth>
  <weight>81</weight>
  <height>180</height>
  <country__of__participation>Bahamas</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Noraida</name>
  <surname>Bicet</surname>
  <country__of__origin>Cuba</country__of__origin>
  <Athletecode>72</Athletecode>
  <gender>Female</gender>
  <date__of__birth>29/10/1977</date__of__birth>
  <weight>87</weight>
  <height>180</height>
  <country__of__participation>Cuba</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Tetyana</name>
  <surname>Lyakhovych</surname>
  <country__of__origin>Ukraine</country__of__origin>
  <Athletecode>73</Athletecode>
  <gender>Female</gender>
  <date__of__birth>20/5/1979</date__of__birth>
  <weight>81</weight>
  <height>178</height>
  <country__of__participation>Ukraine</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Felicia</name>
  <surname>Tilea Moldovan</surname>
  <country__of__origin>Romania</country__of__origin>
  <Athletecode>76</Athletecode>
  <gender>Female</gender>
  <date__of__birth>29/9/1967</date__of__birth>
  <weight>74</weight>
  <height>167</height>
  <country__of__participation>Romania</country__of__participation>
  <partic/>
```

```
</Athlete>
<Athlete>
  <name>Shawn</name>
  <surname>Crawford</surname>
  <country__of__origin>USA</country__of__origin>
  <Athletecode>9</Athletecode>
  <gender>Male</gender>
  <date__of__birth>14/1/1978</date__of__birth>
  <weight>81</weight>
  <height>177</height>
  <country__of__participation>USA</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Obadele</name>
  <surname>Thompson</surname>
  <country__of__origin>Barbados</country__of__origin>
  <Athletecode>10</Athletecode>
  <gender>Male</gender>
  <date__of__birth>30/3/1976</date__of__birth>
  <weight>78</weight>
  <height>176</height>
  <country__of__participation>Barbados</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Vicente</name>
  <surname>Lima</surname>
  <country__of__origin>Brazil</country__of__origin>
  <Athletecode>11</Athletecode>
  <gender>Male</gender>
  <date__of__birth>4/6/1977</date__of__birth>
  <weight>65</weight>
  <height>166</height>
  <country__of__participation>Brazil</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Matic</name>
  <surname>Osovnikar</surname>
  <country__of__origin>Slovenia</country__of__origin>
  <Athletecode>12</Athletecode>
  <gender>Male</gender>
  <date__of__birth>19/1/1980</date__of__birth>
  <weight>77</weight>
  <height>179</height>
  <country__of__participation>Slovenia</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Deji</name>
  <surname>Aliu</surname>
  <country__of__origin>Nigeria</country__of__origin>
  <Athletecode>13</Athletecode>
  <gender>Male</gender>
  <date__of__birth>22/11/1975</date__of__birth>
  <weight>63</weight>
  <height>175</height>
  <country__of__participation>Nigeria</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
```

```
    <name>Nicolas</name>
    <surname>Macrozonaris</surname>
    <country__of__origin>Canada</country__of__origin>
    <Athletecode>14</Athletecode>
    <gender>Male</gender>
    <date__of__birth>22/8/1980</date__of__birth>
    <weight>73</weight>
    <height>181</height>
    <country__of__participation>Canada</country__of__participation>
    <partic/>
  </Athlete>
  <Athlete>
    <name>Gennadiy</name>
    <surname>Chernovol</surname>
    <country__of__origin>Kazakstan</country__of__origin>
    <Athletecode>15</Athletecode>
    <gender>Male</gender>
    <date__of__birth>6/6/1976</date__of__birth>
    <weight>85</weight>
    <height>187</height>
    <country__of__participation>Rusia</country__of__participation>
    <partic/>
  </Athlete>
  <Athlete>
    <name>Idrissa</name>
    <surname>Sanou</surname>
    <country__of__origin>Bur</country__of__origin>
    <Athletecode>16</Athletecode>
    <gender>Male</gender>
    <date__of__birth>12/6/1977</date__of__birth>
    <weight>88</weight>
    <height>188</height>
    <country__of__participation>Bur</country__of__participation>
    <partic/>
  </Athlete>
  <Athlete>
    <name>Justin</name>
    <surname>Gatlin</surname>
    <country__of__origin>USA</country__of__origin>
    <Athletecode>17</Athletecode>
    <gender>Male</gender>
    <date__of__birth>10/2/1982</date__of__birth>
    <weight>83</weight>
    <height>185</height>
    <country__of__participation>USA</country__of__participation>
    <partic/>
  </Athlete>
  <Athlete>
    <name>Jason</name>
    <surname>Gardener</surname>
    <country__of__origin>Great Britain</country__of__origin>
    <Athletecode>18</Athletecode>
    <gender>Male</gender>
    <date__of__birth>18/9/1975</date__of__birth>
    <weight>75</weight>
    <height>176</height>
    <country__of__participation>Great
Britain</country__of__participation>
    <partic/>
  </Athlete>
  <Athlete>
    <name>Uchenna</name>
```

```
  <surname>Emedolu</surname>
  <country__of__origin>Nigeria</country__of__origin>
  <Athletecode>19</Athletecode>
  <gender>Male</gender>
  <date__of__birth>17/9/1976</date__of__birth>
  <weight>68</weight>
  <height>185</height>
  <country__of__participation>Nigeria</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Nobuharu</name>
  <surname>Asahara</surname>
  <country__of__origin>Japan</country__of__origin>
  <Athletecode>20</Athletecode>
  <gender>Male</gender>
  <date__of__birth>21/6/1972</date__of__birth>
  <weight>75</weight>
  <height>179</height>
  <country__of__participation>Japan</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Yeoryios</name>
  <surname>Theodoridis</surname>
  <country__of__origin>Greece</country__of__origin>
  <Athletecode>21</Athletecode>
  <gender>Male</gender>
  <date__of__birth>12/12/1972</date__of__birth>
  <weight>80</weight>
  <height>181</height>
  <country__of__participation>Greece</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Ronald</name>
  <surname>Nemeth</surname>
  <country__of__origin>Hungary</country__of__origin>
  <Athletecode>22</Athletecode>
  <gender>Male</gender>
  <date__of__birth>19/9/1974</date__of__birth>
  <weight>80</weight>
  <height>169</height>
  <country__of__participation>Hungary</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Nicconnor</name>
  <surname>Alexander</surname>
  <country__of__origin>Trinidad</country__of__origin>
  <Athletecode>23</Athletecode>
  <gender>Male</gender>
  <date__of__birth>4/2/1977</date__of__birth>
  <weight>70</weight>
  <height>188</height>
  <country__of__participation>Trinidad</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Eddy</name>
  <surname>De Lepine</surname>
  <country__of__origin>France</country__of__origin>
```

```
  <Athletecode>24</Athletecode>
  <gender>Male</gender>
  <date__of__birth>30/3/1984</date__of__birth>
  <weight>74</weight>
  <height>175</height>
  <country__of__participation>France</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Kim</name>
  <surname>Collins</surname>
  <country__of__origin>SKN</country__of__origin>
  <Athletecode>2</Athletecode>
  <gender>Male</gender>
  <date__of__birth>5/4/1976</date__of__birth>
  <weight>77</weight>
  <height>180</height>
  <country__of__participation>SKN</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Michael</name>
  <surname>Frater</surname>
  <country__of__origin>Jamaica</country__of__origin>
  <Athletecode>3</Athletecode>
  <gender>Male</gender>
  <date__of__birth>6/10/1982</date__of__birth>
  <weight>73</weight>
  <height>175</height>
  <country__of__participation>Jamaica</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Frank</name>
  <surname>Fredericks</surname>
  <country__of__origin>Namibia</country__of__origin>
  <Athletecode>4</Athletecode>
  <gender>Male</gender>
  <date__of__birth>2/10/1967</date__of__birth>
  <weight>75</weight>
  <height>180</height>
  <country__of__participation>Namibia</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Joshua</name>
  <surname>Ross</surname>
  <country__of__origin>Australia</country__of__origin>
  <Athletecode>5</Athletecode>
  <gender>Male</gender>
  <date__of__birth>9/2/1981</date__of__birth>
  <weight>83</weight>
  <height>185</height>
  <country__of__participation>Australia</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Alexander</name>
  <surname>Kosenkow</surname>
  <country__of__origin>Germany</country__of__origin>
  <Athletecode>6</Athletecode>
  <gender>Male</gender>
```

```
  <date__of__birth>14/3/1977</date__of__birth>
  <weight>70</weight>
  <height>178</height>
  <country__of__participation>Rusia</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Andrey</name>
  <surname>Epishin</surname>
  <country__of__origin>Rusia</country__of__origin>
  <Athletecode>7</Athletecode>
  <gender>Male</gender>
  <date__of__birth>10/6/1981</date__of__birth>
  <weight>72</weight>
  <height>177</height>
  <country__of__participation>Rusia</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Jaysuma</name>
  <surname>Saidy Ndure</surname>
  <country__of__origin>Gam</country__of__origin>
  <Athletecode>8</Athletecode>
  <gender>Male</gender>
  <date__of__birth>1/7/1984</date__of__birth>
  <weight>72</weight>
  <height>192</height>
  <country__of__participation>Gam</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Maurice</name>
  <surname>Greene</surname>
  <country__of__origin>USA</country__of__origin>
  <Athletecode>25</Athletecode>
  <gender>Male</gender>
  <date__of__birth>23/7/1974</date__of__birth>
  <weight>81</weight>
  <height>175</height>
  <country__of__participation>USA</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Asafa</name>
  <surname>Powell</surname>
  <country__of__origin>Jamaica</country__of__origin>
  <Athletecode>26</Athletecode>
  <gender>Male</gender>
  <date__of__birth>11/11/1982</date__of__birth>
  <weight>88</weight>
  <height>190</height>
  <country__of__participation>Jamaica</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Leonard</name>
  <surname>Myles Mills</surname>
  <country__of__origin>Ghana</country__of__origin>
  <Athletecode>27</Athletecode>
  <gender>Male</gender>
  <date__of__birth>9/5/1973</date__of__birth>
  <weight>70</weight>
```

```
  <height>175</height>
  <country__of__participation>Ghana</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Lukasz</name>
  <surname>Chyla</surname>
  <country__of__origin>Poland</country__of__origin>
  <Athletecode>28</Athletecode>
  <gender>Male</gender>
  <date__of__birth>31/3/1981</date__of__birth>
  <weight>86</weight>
  <height>188</height>
  <country__of__participation>Poland</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Kareem</name>
  <surname>Streete Thompson</surname>
  <country__of__origin>CAY</country__of__origin>
  <Athletecode>29</Athletecode>
  <gender>Male</gender>
  <date__of__birth>30/3/1973</date__of__birth>
  <weight>91</weight>
  <height>183</height>
  <country__of__participation>USA</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Simone</name>
  <surname>Collio</surname>
  <country__of__origin>Italy</country__of__origin>
  <Athletecode>30</Athletecode>
  <gender>Male</gender>
  <date__of__birth>27/12/1979</date__of__birth>
  <weight>74</weight>
  <height>179</height>
  <country__of__participation>Italy</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Jarbas</name>
  <surname>Mascarenhas</surname>
  <country__of__origin>Brazil</country__of__origin>
  <Athletecode>31</Athletecode>
  <gender>Male</gender>
  <date__of__birth>25/8/1980</date__of__birth>
  <weight>78</weight>
  <height>185</height>
  <country__of__participation>Brazil</country__of__participation>
  <partic/>
</Athlete>
<Athlete>
  <name>Eric Pacome</name>
  <surname>N Dri</surname>
  <country__of__origin>Ivory Coast</country__of__origin>
  <Athletecode>32</Athletecode>
  <gender>Male</gender>
  <date__of__birth>1/1/1978</date__of__birth>
  <weight>78</weight>
  <height>185</height>
  <country__of__participation>Ivory Coast</country__of__participation>
```

```
    <partic/>
  </Athlete>
  <Athlete>
    <name>Aziz</name>
    <surname>Zakari</surname>
    <country__of__origin>Ghana</country__of__origin>
    <Athletecode>1</Athletecode>
    <gender>Male</gender>
    <date__of__birth>2/9/1976</date__of__birth>
    <weight>85</weight>
    <height>175</height>
    <country__of__participation>Ghana</country__of__participation>
    <partic/>
  </Athlete>
  <Athlete>
    <name>Francis</name>
    <surname>Obikwelu</surname>
    <country__of__origin>Portugal</country__of__origin>
    <Athletecode>78</Athletecode>
    <gender>Male</gender>
    <date__of__birth>22/11/1978</date__of__birth>
    <weight>74</weight>
    <height>191</height>
    <country__of__participation>Nigeria</country__of__participation>
    <partic/>
  </Athlete>
</Person-array>
```