ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΝΑΣ OBJECT-ORIENTED ΣΧΕΔΙΑΣΜΟΣ ΓΙΑ HYPERMEDIA ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

πτυχιακή εργασία

ΚΩΣΤΑΣ ΒΑΣΙΛΑΚΗΣ
ΔΗΜΗΤΡΗΣ ΓΚΟΥΣΚΟΣ

επίβλεψη: Μ. Χατζόπουλος

ΑΘΗΝΑ
ΟΚΤΩΒΡΙΟΣ 1990

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Οι *βάσεις δεδομένων* έχουν ήδη πίσω τους μια πολύχρονη ιστορία. Στο διάστημα από την εμφάνισή τους μέχρι σήμερα ο σχεδιασμός τους έγινε συστηματικός, η απόδοσή τους βελτιώθηκε αισθητά και οι εφαρμογές τους επεκτάθηκαν απροσδόκητα. Ενώ αρχικά οι βάσεις δεδομένων αντιμετωπίζονταν απλά ως ένα καλύτερο εργαλείο μηχανογράφησης, σε σχέση με τις συμβατικές μεθόδους με χρήση αρχείων, και με περιορισμένη, κατά συνέπεια, εμβέλεια, σήμερα τις συναντάμε να χρησιμοποιούνται σε τράπεζες πληροφοριών, σε κάθε μορφής εταιρείες και οργανισμούς του τριτογενούς τομέα, σε μικρές και μεγάλες εμπορικές επιχειρήσεις, αλλά και από απλούς ακόμα χρήστες. Οι σύγχρονες βάσεις δεδομένων μπορούν να ενσωματώνονται σε *έμπειρα συστήματα* (expert systems, συντομ. ES) ή σε συστήματα *υποστήριξης αποφάσεων* (decision support systems, συντομ. DSS) και να κατανέμονται, μέσω δικτύων, σε μικρές ή μεγάλες αποστάσεις.

Η εξάπλωση των εφαρμογών των βάσεων δεδομένων συνοδεύτηκε και από την αντίστοιχη αύξηση των απαιτήσεων που παρουσιάζονταν γι' αυτές και, συνακόλουθα, από την εμφάνιση νέων τάσεων για την ανάπτυξή τους. Δύο ουσιώδεις διαπιστώσεις που μπορούν να γίνουν για τις βάσεις δεδομένων σήμερα είναι οι εξής:

(1) Επανέρχεται όλο και πιο πιεστικό το αίτημα να έχουμε βάσεις δεδομένων στις οποίες να μην αποθηκεύεται μόνο κείμενο και δομημένα δεδομένα αλλά και δεδομένα οποιασδήποτε άλλης μορφής, όπως, για παράδειγμα, εικόνα ή video.

(2) Έχουν αρχίσει να γίνονται αισθητοί οι περιορισμοί που επιβάλλει το σχεσιακό μοντέλο, καθώς υπάρχει ανάγκη για λεπτότερη και πολυπλοκότερη διασύνδεση των αποθηκευμένων δεδομένων, για δημιουργία ευέλικτων και εύχρηστων δομών και για δήλωση σημασιολογικών στοιχείων.

Τα δύο αυτά ζητούμενα οδήγησαν, όπως θα φανεί στη συνέχεια, στην εμφάνιση των hypermedia βάσεων δεδομένων . Οι βάσεις αυτές στοχεύουν από τη μια πλευρά να αποθηκεύουν δεδομένα διαφορετικής μορφής και δόμησης και, από την άλλη, να τα συνδέουν ισχυρά και αποδοτικά μεταξύ τους.

Οι hypermedia βάσεις δεδομένων άρχισαν να συζητιούνται σε μια εποχή που και μια άλλη καινούρια τάση εμφανιζόταν στο χώρο της ανάπτυξης λογισμικού, ο *object-oriented προγραμματισμός* (object-oriented programming, συντομ. OOP). Αυτή η προγραμματιστική μεθοδολογία αναπτύχθηκε κυρίως για να αντιμετωπίσει τα προβλήματα που είχαν εντοπιστεί στο χώρο της *τεχνολογίας λογισμικού* (software engineering, συντομ. SE), και τα οποία αφορούσαν την ποιότητα του παραγόμενου λογισμικού (με την ευρεία έννοια του όρου "ποιότητα λογισμικού"), την ικανότητα του λογισμικού να συντηρηθεί εύκολα και να χρησιμοποιηθεί σε καινούριες εφαρμογές και γενικά την αύξηση της παραγωγικότητας κατά την ανάπτυξη λογισμικού. Η object-oriented φιλοσοφία βρήκε εφαρμογή σε μια σειρά από προβλήματα ανάλυσης και σχεδίασης από το χώρο της Πληροφορικής και χρησιμοποιήθηκε, όπως ήταν επόμενο, και στο σχεδιασμό βάσεων δεδομένων. Τα αποτελέσματα των προσπαθειών που έγιναν ήταν ενθαρρυντικά και, σήμερα πλέον, πολλές εργασίες γίνονται σχετικά με object-oriented σχεδιασμό βάσεων δεδομένων. Οι πρώτες εμπορικές object-oriented βάσεις δεδομένων είναι ήδη γεγονός.

Η παρούσα εργασία αναφέρεται στον object-oriented σχεδιασμό hypermedia βάσεων δεδομένων και είναι χωρισμένη σε πέντε μέρη:

Στο πρώτο μέρος (κεφάλαια 2 και 3) αναλύονται και οριοθετούνται οι έννοιες του object-oriented προγραμματισμού αφενός, και των συστημάτων hypertext, multimedia και hypermedia αφετέρου.

Στο δεύτερο μέρος (κεφάλαια 4 ως 9) αιτιολογείται η επιλογή του object-oriented μοντέλου για το σχεδιασμό μιας hypermedia βάσης δεδομένων, παρουσιάζεται ο σχεδιασμός αυτός και αναλύονται επιμέρους ζητήματα όπως η υποστήριξη όψεων (views) και εκδοχών (versions), καθώς και η διαχείριση μιας τέτοιας βάσης. Επίσης προτείνονται λύσεις για το ζήτημα του user interface για βάσεις hypermedia και περιγράφεται μια δηλωτική γλώσσα χειρισμού δεδομένων (data manipulation language, συντομ. DML) για τη σχεδιασμένη βάση.

Στο τρίτο μέρος, (κεφάλαιο 10) αξιολογείται ο προτεινόμενος σχεδιασμός και αναφέρονται οι δυνατότητές του για επέκταση.

Στο τέταρτο μέρος (παραρτήματα Α, Β και Γ) παρουσιάζεται η εργασία που έχει γίνει για να υλοποιηθεί μελλοντικά μια πρωτότυπη hypermedia βάση δεδομένων στηριγμένη στη σχεδίαση που

αναπτύχθηκε ήδη· συγκεκριμένα παρουσιάζονται (i) το σχήμα της object-oriented βάσης δεδομένων (παράρτημα Α) (ii) ο ορισμός σε BNF μιας δηλωτικής γλώσσας για το χειρισμό των δεδομένων (παράρτημα Β) και (iii) ο κώδικας που γράφτηκε σε C++ για τη δημιουργία μιας πρωτότυπης βάσης (παράρτημα Γ).

Τέλος, στο πέμπτο και τελευταίο μέρος (παραρτήματα Δ και Ε) παρατίθεται ένα γλωσσάρι τεχνικών όρων και βιβλιογραφία. Το γλωσσάρι περιέχει τους αγγλικούς όρους που χρειάστηκε να μεταφραστούν, καθώς και έναν κατάλογο με όρους που παρέμειναν αμετάφραστοι, ελλείψει δόκιμων ελληνικών αντίστοιχων. Κατά την απόδοση των αγγλικών όρων έγινε προσπάθεια να αποδοθεί το νόημα του πρωτότυπου και όχι απλώς να δοθεί η αντίστοιχη λέξη· το αποτέλεσμα ίσως μερικές φορές να ξενίζει, ελπίζουμε όμως ότι τελικά θα διευκολύνει την κατανόηση του κειμένου. Τέλος, θα πρέπει να σημειωθεί το ότι ένα μεγάλο μέρος από την εργασία αυτή στηρίζεται σε αποτελέσματα που προέκυψαν από την ερευνητική δραστηριότητα της ελληνικής ομάδας για το task HYPERATE του project SAFE την οποία αποτελούν, εκτός από τους γράφοντες, ο αναπληρωτής καθηγητής του Τμήματος Πληροφορικής Μιχάλης Χατζόπουλος, οι μεταπτυχιακοί φοιτητές Μαρία Σπηλιοπούλου και Μιχάλης Βαζιργιάννης, και η προπτυχιακή φοιτήτρια Κατερίνα Δημοπούλου. Θέλουμε να ευχαριστήσουμε όλους αυτούς για τη βοήθειά τους.

Αθήνα, Σεπτέμβριος 1990

Κώστας Βασιλάκης
Δημήτρης Γκούσκος

1. OBJECT-ORIENTED ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΚΑΙ ΜΕΘΟΔΟΛΟΓΙΕΣ

1.1. Οι αρχές του object-oriented προγραμματισμού

Η έννοια του object-oriented προγραμματισμού άρχισε να μορφοποιείται στα μέσα της δεκαετίας του '70. Ο object-oriented προγραμματισμός έχει ως στόχο την αύξηση της παραγωγικότητας των προγραμματιστών και της ικανότητας των προγραμμάτων να *συντηρούνται εύκολα* (program maintainability) και θεωρείται το τρίτο σημαντικό βήμα προς την κατεύθυνση αυτή μετά την εισαγωγή των *γλωσσών προγραμματισμού υψηλού επιπέδου* (high-level programming languages) στις αρχές της δεκαετίας του '50 και του *δομημένου προγραμματισμού* (structured programming) στα μέσα της δεκαετίας του '60. Αν και ως τώρα δεν έχουν διατυπωθεί σαφή κριτήρια σύμφωνα με τα οποία οι γλώσσες προγραμματισμού να διαχωρίζονται σε object-oriented και μη, φαίνεται ότι τέσσερα χαρακτηριστικά που θα πρέπει να υποστηρίζει μια γλώσσα προγραμματισμού προκειμένου να θεωρηθεί object-oriented είναι η *αφαίρεση των δεδομένων* (data abstraction), η *περιχαράκωση των δεδομένων* (data encapsulation), η *κληρονομικότητα των ιδιοτήτων* (property inheritance) και ο *πολυμορφισμός* (polymorphism) ([PIN88]). Άλλα κριτήρια που έχουν προταθεί είναι η δυνατότητα για ορισμό *αφηρημένων τύπων δεδομένων* (abstract data types), και η απόδοση διαφορετικής ταυτότητας για κάθε αντικείμενο με χρήση *προσδιοριστών αντικειμένων* (object identifiers) ([PAR89]).

Κεντρική έννοια για τον object-oriented προγραμματισμό είναι τα *αντικείμενα* (objects). Χωρίς αυτό να αποτελεί και ορισμό, μπορούμε να πούμε ότι ως αντικείμενο μέσα στα πλαίσια της object-oriented φιλοσοφίας θεωρείται κάθε δεδομένο και κάθε λογική οντότητα από δεδομένα. Μέσα σε ένα γνήσιο object-oriented περιβάλλον, όπως, για παράδειγμα, το περιβάλλον της Smalltalk-80, αντικείμενα θεωρούνται και τα κομμάτια κώδικα που υπάρχουν σε αντίστοιχα αρχεία. Στη γενική, ωστόσο, περίπτωση, μιλώντας για μια object-oriented γλώσσα μπορούμε να θεωρούμε ότι ως αντικείμενα νοούνται μόνο τα δεδομένα.

Με τον όρο *αφαίρεση* εννοούμε την αναπαράσταση ενός πολύπλοκου αντικειμένου κατά τρόπο τέτοιο ώστε να μην είναι απαραίτητη για την κατανόηση της λειτουργικότητας ή της χρησιμότητάς του η γνώση της εσωτερικής του δομής . Αυτό βρίσκεται σε πλήρη αναλογία με τον τρόπο που οι άνθρωποι σκέπτονται και ενεργούν: όλοι, για παράδειγμα, ξέρουν τη χρησιμότητα ενός αεροπλάνου, χωρίς, όμως, να γνωρίζουν παράλληλα και τους φυσικούς νόμους που επιτρέπουν στο αεροπλάνο να πετάει, ή τις λεπτομέρειες της κατασκευής του.

Η αφαίρεση έχει σκοπό να απαλλάξει τους χρήστες από την ανάγκη να ξέρουν τις εσωτερικές δομές αναπαράστασης των αντικειμένων. Από την άλλη πλευρά, η αρχή της *περιχαράκωσης* στοχεύει στην προστασία των δεδομένων από την παρέμβαση των χρηστών. Σύμφωνα με την αρχή αυτή ένας χρήστης έχει δικαίωμα προσπέλασης στα δεδομένα ενός αντικειμένου μόνο μέσω προκαθορισμένων διαδικασιών που ονομάζονται *μέθοδοι* (methods), ή, αντίστροφα, το κάθε αντικείμενο παρέχει στους χρήστες δικαίωμα να προσπελάσουν τα δεδομένα του όχι με οποιεσδήποτε λειτουργίες, αλλά μόνο μέσω των δικών του *λειτουργιών προσπέλασης* (accessor functions). Με τον τρόπο αυτό προστατεύεται η *ακεραιότητα* (integrity) και η *συνέπεια* (consistency) των δεδομένων, ενώ παράλληλα προστατεύονται και τα προγράμματα των χρηστών από αλλαγές που μπορεί να γίνουν στην εσωτερική δομή των αντικειμένων. Με άλλα λόγια, δεδομένα και μέθοδοι παρουσιάζονται στον χρήστη σαν ένα ενιαίο πακέτο , το οποίο περιέχει στο αόρατο εσωτερικό του προστατευμένα τα δεδομένα, και στο ορατό του περίβλημα τις μεθόδους, οι οποίες αποτελούν και το μόνο τρόπο επικοινωνίας του πακέτου αυτού με το προγραμματιστικό περιβάλλον. Αυτό το πακέτο αποτελεί ένα αντικείμενο .

Τα αντικείμενα που περιέχουν εσωτερικά τα ίδια δεδομένα και διαθέτουν εξωτερικά τις ίδιες μεθόδους για τη χρήση των δεδομένων αυτών ομαδοποιούνται σε μια *κλάση* (class). Τα αντικείμενα που ανήκουν σε κάποια κλάση ονομάζονται *στιγμιότυπα* της κλάσης αυτής (class instances). Όλα τα στιγμιότυπα μιας κλάσης έχουν τις ίδιες *χαρακτηριστικές* (attributes), οι οποίες όμως μπορούν να παίρνουν διαφορετικές τιμές για διαφορετικά στιγμιότυπα. Οι χαρακτηριστικές μιας κλάσης απεικονίζονται στις *μεταβλητές* (data variables) των αντικειμένων της κλάσης αυτής. Είναι δυνατό κάποια κλάση να μην έχει στιγμιότυπα, αλλά

να υπάρχει μόνο και μόνο για να ομαδοποιούνται, κάτω από τον ορισμό της, συναφείς μέθοδοι και χαρακτηριστικές. Μια τέτοια κλάση θα ονομάζεται *αφηρημένη κλάση* (abstract class). Οι μέθοδοι μιας κλάσης αποτελούν το *δημόσιο τμήμα* (public part) της κλάσης αυτής, με την έννοια ότι μέθοδοι από όλες τις άλλες κλάσεις έχουν δικαίωμα να τις προσπελάσουν και να τις χρησιμοποιήσουν, ενώ τα δεδομένα μιας κλάσης αποτελούν το *ιδιωτικό τμήμα* (private part) της κλάσης αυτής, με την έννοια ότι μόνο οι μέθοδοι της συγκεκριμένης κλάσης μπορούν να τα προσπελάσουν και να τα χρησιμοποιήσουν. Οι γλώσσες προγραμματισμού και τα διάφορα προϊόντα λογισμικού που υποστηρίζουν αντικείμενα, δεν υποστηρίζουν αναγκαστικά και κλάσεις αντικειμένων. Για την πρώτη κατηγορία προϊόντων έχει προταθεί ο χαρακτηρισμός *βασισμένα στα αντικείμενα* (object-based), ενώ για τη δεύτερη κατηγορία ο χαρακτηρισμός *προσανατολισμένα στα αντικείμενα* (object-oriented) ([WEG87]).

Οι κλάσεις οργανώνονται σε *ιεραρχίες κλάσεων* (class hierarchies) στις οποίες κάθε κλάση που είναι παιδί μιας άλλης κλάσης ονομάζεται *υποκλάση* (subclass) της κλάσης αυτής, και κάθε κλάση που είναι πατέρας κάποιων άλλων κλάσεων ονομάζεται *υπερκλάση* (superclass) των κλάσεων αυτών. Κάθε κλάση-παιδί (child class) σε μια τέτοια ιεραρχία κληρονομεί τόσο τα δεδομένα όσο και τις μεθόδους των *κλάσεων-γονέων* της (parent classes). Η ιδιότητα αυτή ονομάζεται *κληρονομικότητα* . Θα πρέπει να σημειωθεί ότι οι έννοιες των κλάσεων, των ιεραρχιών από κλάσεις και της κληρονομικότητας δεν εμφανίστηκαν για πρώτη φορά με την object-orinted φιλοσοφία, αλλά εισήχθησαν κατά τη δεκαετία του '60 στη Simula67, μια γλώσσα φτιαγμένη ειδικά για να επιλύει προβλήματα προσομοίωσης. Υπάρχει και η περίπτωση μια κλάση-παιδί να κληρονομεί δεδομένα κλάσεων άλλων από τις κλάσεις-προγόνους της (πράγμα που κατά κάποιο τρόπο παραβιάζει την ιεραρχία των κλάσεων). Στην περίπτωση αυτή λέμε ότι έχουμε *πολλαπλή κληρονομικότητα* (multiple inheritance). Οι περισσότερες object-oriented γλώσσες προγραμματισμού υποστηρίζουν σε μικρό βαθμό την πολλαπλή κληρονομικότητα. Ανάλογα με την υλοποίηση της κάθε object-oriented γλώσσας προγραμματισμού, μία κλάση-παιδί μπορεί να έχει ή να μην έχει άμεση προσπέλαση, μέσα από τις μεθόδους της, στα δεδομένα που κληρονομεί από τις υπερκλάσεις της. Είναι δυνατό μια κλάση να επανακαθορίσει τη λειτουργία μιας μεθόδου που κληρονόμησε από έναν πρόγονό της. το φαινόμενο αυτό ονομάζεται *πολυμορφικός επανορισμός* (polymorphic redefinition). Γενικότερα, είναι δυνατό μια μέθοδος να εμφανίζεται σε περισσότερες από μία κλάσεις με διαφορετικούς ορισμούς˙ το φαινόμενο αυτό ονομάζεται *πολυμορφισμός* (polymorphism). Όταν ο πολυμορφισμός εφαρμόζεται σε τελεστές, τότε ονομάζεται *επιφόρτωση τελεστών* (operator overloading). Εδώ θα πρέπει να σημειωθεί ότι πολυμορφικά χαρακτηριστικά υπάρχουν και στις συμβατικές, μη object-oriented γλώσσες (ο ίδιος τελεστής, για παράδειγμα, χρησιμοποιείται κατά κανόνα για πρόσθεση πραγματικών και ακέραιων, και η ίδια εντολή χρησιμοποιείται συνήθως για ανάγνωση αριθμών και χαρακτήρων). Στις object-oriented γλώσσες, ωστόσο, ο πολυμορφισμός αναδεικνύεται σε βασικό χαρακτηριστικό του προγραμματισμού και γενικεύεται για όλα τα είδη δεδομένων και για όλες τις λειτουργίες.

Κάθε εντολή ενός προγράμματος γραμμένου σε μια object-oriented γλώσσα υλοποιείται με *αποστολή μηνυμάτων* (messages) σε αντικείμενα. Τα μηνύματα έχουν ως αποστολή να ενεργοποιήσουν καθορισμένες μεθόδους ώστε οι τελευταίες να δράσουν στα αντικείμενα ή αλλιώς, όταν τα αντικείμενα λαμβάνουν μηνύματα, ενεργοποιούν κατάλληλες μεθόδους που δρουν πάνω στα πρώτα. Πιο συγκεκριμένα, όταν ένα μήνυμα φθάνει σε ένα αντικείμενο, το μήνυμα ψάχνει, στη λίστα των λειτουργιών της κλάσης όπου το αντικείμενο ανήκει, για μια λειτουργία που θα ενεργοποιηθεί. Αν η λειτουργία αυτή δε βρεθεί εκεί αναζητείται στη λίστα των λειτουργιών της γονικής κλάσης, κοκ., μέχρι να βρεθεί σε κάποια κλάση. Η αλληλεπίδραση αντικειμένων και μηνυμάτων μας κάνει να λέμε ότι τα αντικείμενα σε μία object-oriented γλώσσα προγραμματισμού είναι *ενεργητικά*, σε αντιδιαστολή με τα *παθητικά* αντικείμενα των κλασικών αλγοριθμικών γλωσσών προγραμματισμού. Θα πρέπει εδώ να σημειώσουμε πως όταν λέγεται ότι "τα αντικείμενα λαμβάνουν μηνύματα και ενεργοποιούν κατάλληλες μεθόδους" δεν υπονοείται κανένα είδος παράλληλης επεξεργασίας: οι object-oriented γλώσσες προγραμματισμού δεν έχουν σχεδιαστεί για να ανακαλύπτουν παραλληλία μέσα στα προγράμματα, αλλά για να διευκολύνουν την ανάπτυξη πιο καλών προγραμμάτων. Το ότι η ίδια η φύση του μηχανισμού εκτέλεσης ενός προγράμματος με *πέρασμα μηνυμάτων* (message passing) επιδέχεται παράλληλη υλοποίηση, δε σημαίνει υποχρεωτικά ότι οι object-oriented γλώσσες είναι και παράλληλες. Η εκτέλεση των προγραμμάτων στις object-oriented γλώσσες προγραμματισμού οι οποίες έχουν αναπτυχθεί ως τώρα είναι σειριακή.

1.2. 2. Object-oriented γλώσσες προγραμματισμού

Την τελευταία δεκαετία έχουν αναπτυχθεί αρκετές γλώσσες προγραμματισμού που κινούνται στην τροχιά του προσανατολισμού στα αντικείμενα. Θα πρέπει να πούμε εδώ ότι στοιχεία object-orientation συναντάμε και σε παλιότερες γλώσσες προγραμματισμού˙ την έννοια της περιχαράκωσης δεδομένων, για παράδειγμα, τη συναντάμε στα *πακέτα* (packages) της Ada και στα *αντικείμενα* (objects) της Simula, οι οποίες αναπτύχθηκαν το 1960 και το 1970 αντίστοιχα. Δεν μπορούμε, ωστόσο, να πούμε ότι αυτές οι γλώσσες προγραμματισμού είναι object-oriented αφού δεν ικανοποιούν όλες τις προϋποθέσεις που αναφέρθηκαν πιο πάνω.

Υπάρχουν δύο προσεγγίσεις για τη δημιουργία μιας object-oriented γλώσσας προγραμματισμού. Μπορεί κανείς να ξεκινήσει από πολύ χαμηλό επίπεδο και να χτίσει μια εντελώς νέα γλώσσα, ή να πάρει μία από τις ήδη αναπτυγμένες γλώσσες προγραμματισμού (π.χ. Pascal ή C) και να την επεκτείνει έτσι ώστε να υποστηρίζει και χαρακτηριστικά object-orientation. Και οι δύο προσεγγίσεις έχουν τα υπέρ και τα κατά τους, τα οποία θα εξεταστούν παρακάτω.

1.2.1. Smalltalk-80

Είναι γενικά παραδεκτό ότι η γλώσσα προγραμματισμού που ακολουθεί πιο πιστά τις αρχές του object-oriented προγραμματισμού είναι η Smalltalk-80 ([PIN88] και [KAE86]). Η Smalltalk-80 αναπτύχθηκε στα εργαστήρια της Xerox και κυκλοφόρησε στο εμπόριο στις αρχές της δεκαετίας του '80. Σήμερα υπάρχουν εκδόσεις της Smalltalk-80 που τρέχουν σε υπολογιστές Apple MacIntosh, σε σταθμούς εργασίας Sun, σε IBM PC συμβατούς υπολογιστές καθώς και σε άλλες μηχανές.

Η ανάπτυξη της Smalltalk-80 έγινε σύμφωνα με την πρώτη από τις δύο προσεγγίσεις που αναφέρθηκαν πιο πάνω, δηλαδή άρχισε από πολύ χαμηλά. Η Smalltalk-80 έχει πολύ απλό συντακτικό, καθώς όλα τα αντικείμενα έχουν ομοιόμορφη αντιμετώπιση˙ αυτό το γεγονός ισχύει ακόμη και για τα κομμάτια του κώδικα που αποτελούν το περιβάλλον προγραμματισμού ή που αναπτύσσονται μέσα στο τελευταίο. Έτσι, η σύνταξη της Smalltalk-80 ορίζει μόνο τις παρακάτω λειτουργίες (που είναι αρκετές για την ανάπτυξη οποιουδήποτε προγράμματος στη γλώσσα αυτή): δήλωση μεταβλητών, εκχώρηση τιμών, αποστολή μηνυμάτων σε αντικείμενα, ορισμό καινούριων κλάσεων και ορισμό καινούριων μεθόδων. Οι δύο τελευταίες λειτουργίες γίνονται με τη βοήθεια του browser του συστήματος. Κατά τη διάρκεια της εκτέλεσης ενός προγράμματος Smalltalk-80, είναι δυνατό να εκχωρηθούν στην ίδια μεταβλητή αντικείμενα διαφορετικών τύπων, καθώς η Smalltalk είναι γλώσσα προγραμματισμού *ασθενών τύπων* (weakly-typed programming language). Κατά συνέπεια, εφόσον για κάθε τύπο αντικειμένων έχουμε, γενικά, διαφορετικές μεθόδους, η επιλογή της μεθόδου που αντιστοιχεί στην αποστολή ενός μηνύματος σε μια μεταβλητή της Smalltalk-80 γίνεται αποκλειστικά κατά την διάρκεια της εκτέλεσης ενός προγράμματος. Η ιδιότητα αυτή είναι γνωστή ως *καθυστερημένη δέσμευση* (late binding) των μεταβλητών με κλάσεις αντικειμένων.

Η Smalltalk-80, όμως, είναι κάτι περισσότερο από μία απλή γλώσσα προγραμματισμού: αποτελεί ένα ολόκληρο περιβάλλον ανάπτυξης προγραμμάτων, το οποίο αφενός περιλαμβάνει όλα τα εργαλεία που χρειάζονται για τον προγραμματισμό, όπως text editors, debuggers κλπ., και αφετέρου παρέχει στους χρήστες τη δυνατότητα να κινηθούν μέσα σε ολόκληρο το σύστημα μέσω του browser που διαθέτει, και να κάνουν αλλαγές όπου αυτοί κρίνουν σκόπιμο. Ο browser παρέχει επίσης εργαλεία για την επικοινωνία όσων προγραμματιστών εργάζονται πάνω στο ίδιο πρόγραμμα.

Η ομοιομορφία αυτή της Smalltalk-80 μπορεί να χαρακτηριστεί ωφέλιμη ή βλαπτική. Από την μία πλευρά, το περιβάλλον της Smalltalk είναι πολύ φιλικό και εύχρηστο (είναι αυτό, μάλιστα, που κατά γενική ομολογία έχει επηρεάσει την φιλοσοφία αρκετών συστημάτων μεταξύ των οποίων συγκαταλέγονται το MacIntosh και το Lisa της Apple) και παρέχει όλα τα εργαλεία για την ανάπτυξη προγραμμάτων. Από την άλλη πλευρά, ωστόσο, αναγκάζει τους προγραμματιστές να εγκαταλείψουν τα προγραμματιστικά περιβάλλοντα με τα οποία έχουν εξοικειωθεί. Στο μειονέκτημα αυτό θα πρέπει να προστεθεί το ότι δεν υπάρχει, μέχρι στιγμής τουλάχιστον, *μεταγλωττιστής* (compiler) της Smalltalk, παρά μόνο *διερμηνευτές* (interpreters). Άλλο ένα σημείο της Smalltalk που έχει συζητηθεί είναι το ότι επιτρέπει σε όλες τις κλάσεις-παιδιά να έχουν προσπέλαση στα δεδομένα των γονικών κλάσεων. Το γεγονός αυτό παραβιάζει την αρχή της περιχαράκωσης, αφού μία αλλαγή στη δομή των δεδομένων μιας γονικής κλάσης συνεπάγεται αλλαγές στις μεθόδους όλων των κλάσεων-απογόνων της.

1.2.2. Objective-C

Σε αντίθεση με τη Smalltalk, η Objective-C ([COX86]) δε φτιάχτηκε από την αρχή, αλλά με την εμφύτευση object-oriented χαρακτηριστικών σε μια γνωστή και πολύ διαδεδομένη γλώσσα προγραμματισμού, τη C. Η Objective-C αναπτύχθηκε από τον Brad Cox στα μέσα της δεκαετίας του 1980. Αυτός είναι και ο λόγος για τον οποίο δεν υπάρχουν μεταγλωττιστές της Objective-C, αλλά μόνο *μεταφραστές* (translators) που μεταφράζουν τον πηγαίο κώδικα της Objective-C σε πηγαίο κώδικα C, και από εκεί και πέρα δίνουν τον κώδικα αυτόν σε ένα μεταγλωττιστή για C.

Η σύνταξη της Objective-C δίνει τη δυνατότητα για ορισμό νέων κλάσεων και για αποστολή μηνυμάτων σε αντικείμενα. Τα αντικείμενα που θα λάβουν τα μηνύματα αυτά πρέπει να έχουν δηλωθεί με τύπο *id* , που είναι και ο μοναδικός τύπος δεδομένων της Objective-C ο οποίος δεν υπάρχει στη C. Ο τύπος αυτός χρησιμοποιείται για όλες τις κλάσεις που φτιάχνουν οι προγραμματιστές. Για τις μεταβλητές αυτού του τύπου ισχύει η καθυστερημένη δέσμευση, ενώ για όλες τις άλλες μεταβλητές η δέσμευση είναι *πρόωρη* (early binding) (δηλαδή η επιλογή της μεθόδου που αντιστοιχεί στην αποστολή ενός μηνύματος σε μια μεταβλητή γίνεται κατά τη διάρκεια της μεταγλώττισης (compile time)). Οι μεταβλητές που ορίζονται σε κάποια κλάση της Objective-C δεν προσπελάζονται άμεσα από τις μεθόδους των υποκλάσεών της. Τα αρχεία που περιέχουν τις περιγραφές κλάσεων είναι δυνατό να μεταγλωττίζονται ξεχωριστά και να συγκροτούν μια βιβλιοθήκη για τον ορισμό νέων κλάσεων ή την ανάπτυξη προγραμμάτων. Τα αρχεία που προκύπτουν από τη διαδικασία αυτή της μετάφρασης ονομάζονται *software ICs* , κατ' αναλογία προς τα ηλεκτρονικά ολοκληρωμένα κυκλώματα.

Ένας προγραμματιστής της Objective-C μπορεί να χρησιμοποιήσει την καινούρια σύνταξη που προσφέρει η γλώσσα και να τηρήσει όλες τις αρχές του object-oriented προγραμματισμού, ή να εξακολουθήσει να χρησιμοποιεί τις χαμηλού επιπέδου τεχνικές που χρησιμοποιούνται στη C όπως είναι η απευθείας προσπέλαση στη μνήμη, η χρήση bit-fields (μεταβλητών των οποίων το μέγεθος δηλώνεται απευθείας σε πλήθος δυαδικών ψηφίων), κλπ. Παρατηρούμε, επομένως, ότι η ευθύνη για την τήρηση των αρχών του object-oriented προγραμματισμού μετατοπίζεται από την ίδια τη γλώσσα προγραμματισμού στον προγραμματιστή. Είναι γενικά αμφίβολο αν κάτι τέτοιο είναι επιθυμητό, έστω και αν πολλοί προγραμματιστές το αντιμετωπίζουν θετικά, με το επιχείρημα ότι τους παρέχεται η δυνατότητα να χρησιμοποιήσουν αποτελεσματικές (από άποψη χρόνου) και δοκιμασμένες τεχνικές.

1.2.3. C++

Η C++ ([BER88]) αναπτύχθηκε στα εργαστήρια της AT&T, όπου είχε αναπτυχθεί και η C, και κυκλοφόρησε στο εμπόριο περίπου το 1985. 'Οπως και η Objective-C, έτσι και η C++ δε χτίστηκε από την αρχή, αλλά με την επέκταση της C με object-oriented χαρακτηριστικά. Η διαδικασία δημιουργίας εκτελέσιμων αρχείων είναι επίσης παρόμοια με αυτή της Objective-C, δηλαδή πρώτα ένας μεταφραστής μεταφράζει τον κώδικα της C++ σε κώδικα C, και κατόπιν ένας μεταγλωττιστής C παράγει το εκτελέσιμο πρόγραμμα.

Η C++ έχει επεκτείνει το συντακτικό της C έτσι ώστε να περιλάβει δυνατότητες για ορισμό νέων κλάσεων και αποστολή μηνυμάτων σε αντικείμενα. Υπάρχουν ακόμη και άλλες επεκτάσεις, όπως η εισαγωγή της έννοιας του *πρωτοτύπου μιας συνάρτησης* (function prototype) που αργότερα υιοθετήθηκε από την ANSI C, η δυνατότητα για *κλήση με αναφορά* (call by reference), κ.ά. Η C++ είναι γλώσσα *ισχυρών τύπων* (strongly typed), δηλαδή κάθε μεταβλητή έχει έναν και μόνο τύπο στην *περιοχή ισχύος* της (έναν από τους τύπους που υπάρχουν στην C ή μία από τις κλάσεις ή *δομές* (structures) που έχει ορίσει ο χρήστης). Το γεγονός αυτό είναι αρκετά δεσμευτικό και περιορίζει τη χρησιμότητα του πολυμορφισμού. Η C++ χρησιμοποιεί πρόωρη δέσμευση για τις μεταβλητές της εκτός αν ο προγραμματιστής ζητήσει με ειδική δήλωση κάποιας μεθόδου να ισχύσει καθυστερημένη δέσμευση. Η χρήση της καθυστερημένης δέσμευσης είναι επίσης αρκετά περιορισμένη και μπορεί να χρησιμοποιηθεί μόνο για κλάσεις που η μία να είναι άμεση ή έμμεση πρόγονος της άλλης. Ο προγραμματιστής μπορεί επίσης να ορίσει αν κάποια μεταβλητή ή μέθοδος μιας κλάσης θα είναι *δημόσια* (public), δηλαδή προσπελάσιμη από οποιοδήποτε τμήμα κώδικα χρησιμοποιεί το αντικείμενο, *προστατευμένη* (protected), δηλαδή προσπελάσιμη μόνο από τις μεθόδους της κλάσης όπου ορίζεται και των απογόνων της, ή *ιδιωτική* (private), δηλαδή προσπελάσιμη μόνο από τις μεθόδους της κλάσης όπου ορίζεται. Μπορούν ακόμη να οριστούν *φίλες συναρτήσεις* (friend functions) στις οποίες επιτρέπεται η προσπέλαση ακόμη και στα ιδιωτικά δεδομένα κάποιας κλάσης.

'Οπως στην Objective-C, έτσι και στη C++ η ευθύνη για την εφαρμογή των αρχών του object-oriented προγραμματισμού βαραίνει τον προγραμματιστή και όχι το μεταγλωττιστή, πράγμα που παρουσιάζει τα πλεονεκτήματα και τα μειονεκτήματα που αναφέρθηκαν πιο πάνω. Ειδικά στη C++, η παραβίαση της object-oriented φιλοσοφίας διευκολύνεται ακόμα περισσότερο, καθώς ο προγραμματιστής είναι δυνατό να παραβιάσει την αρχή της περιχαράκωσης ορίζοντας κλάσεις με δημόσια ή προστατευμένα δεδομένα, ή ακόμα και ιδιωτικά δεδομένα τα οποία προσπελάζονται με φίλες συναρτήσεις.

Υπάρχουν βέβαια και άλλες object-oriented γλώσσες προγραμματισμού που έχουν δημιουργηθεί σύμφωνα με τις μεθόδους που περιγράψαμε. Παραδείγματα object-oriented γλωσσών που αναπτύχθηκαν από την αρχή είναι η Eiffel και η Trellis/Owl, ενώ με επέκταση συμβατικών γλωσσών δημιουργήθηκαν η Neon (βασισμένη στη Forth), οι ExperCommonLisp, CommonObjects και Flavors (βασισμένες στη Lisp), η Object Pascal (βασισμένη στην Pascal) και άλλες.

1.3. Object-oriented μεθοδολογίες σχεδιασμού

Η φιλοσοφία του προσανατολισμού στα αντικείμενα δεν εξαντλείται στην ανάπτυξη κάποιων γλωσσών προγραμματισμού με μερικά συγκεκριμένα χαρακτηριστικά. Θα πρέπει μάλλον να μιλάμε για object-oriented μεθοδολογίες, παρά απλά για object-oriented προγραμματισμό. Οι μεθοδολογίες αυτές διέπουν την ανάπτυξη του λογισμικού από το στάδιο του σχεδιασμού μέχρι το στάδιο της υλοποίησης και της συντήρησης. Ειδικά οι object-oriented μεθοδολογίες για σχεδιασμό είναι ιδιαίτερα δημοφιλείς, καθώς,

σε αντίθεση με τις άλλες μεθόδους, ο object-oriented σχεδιασμός (object-oriented design, συντομ. OOD), συνδέει τα δεδομένα και τις λειτουργίες με τέτοιο τρόπο, ώστε να οργανώνεται σε καλά συνεργαζόμενες ενότητες όχι μόνο η επεξεργασία αλλά και η πληροφορία ([PRE82])

Ο object-oriented σχεδιασμός αποτελείται, σύμφωνα με το Λόρενσεν (Lorensen, [LOR86]), από τα εξής έξη βήματα :

(1) Αναγνώριση των αντικειμένων της συγκεκριμένης εφαρμογής και των κλάσεων στις οποίες ομαδοποιούνται αυτά. Οι κλάσεις αυτές πρέπει να αναγνωρίζονται από τις *πιο γενικευμένες* προς τις *πιο εξειδικευμένες* (προσέγγιση top-down), εκτός από τις περιπτώσεις των κλάσεων που ορίζονται σαφώς από την ανάλυση του προβλήματος˙ όσον αφορά αυτές τις κλάσεις, η διαδικασία σχεδιασμού πρέπει να προχωρά από κάτω προς τα πάνω (προσέγγιση bottom-up).

(2) Αναγνώριση των χαρακτηριστικών κάθε κλάσης αντικειμένων. Εδώ καθορίζουμε μόνο το *ποιες* θα είναι οι χαρακτηριστικές αυτές και όχι το *πώς* θα απεικονιστούν σε μεταβλητές.

(3) Αναγνώριση των λειτουργιών που τα αντικείμενα κάθε κλάσης θα πρέπει να εκτελούν. Δεν καθορίζουμε το *πώς* θα γίνουν αυτές οι λειτουργίες αλλά μόνο το *ποιες* θα είναι.

(4) Αναγνώριση των περιπτώσεων στις οποίες επικοινωνούν τα αντικείμενα μεταξύ τους. Ουσιαστικά εδώ καθορίζουμε ποια μηνύματα μπορούν να λάβουν τα αντικείμενα κάθε κλάσης και ποιες λειτουργίες θα ενεργοποιούν κάθε φορά.

Θα πρέπει να σημειωθεί εδώ ότι οι κλάσεις αντικειμένων , οι χαρακτηριστικές , οι *λειτουργίες* και τα *μηνύματα* που αναγνωρίζονται μπορούν να προκύψουν κατά τρόπο τυπικό από μια πλήρη και ακριβή περιγραφή της αντίστοιχης εφαρμογής ([PRE82]).

(5) Έλεγχος του σχεδιασμού. Εδώ ελέγχουμε αν οι κλάσεις αντικειμένων, οι χαρακτηριστικές, οι μέθοδοι και τα μηνύματα που αναγνωρίστηκαν επαρκούν για τις ανάγκες που έχει καταγράψει η ανάλυση. Προσπαθούμε να εντοπίσουμε λειτουργίες που δε γίνονται καλά ή καθόλου με το μοντέλο που έχουμε σχηματίσει, και ελέγχουμε αν το μοντέλο μας ανταποκρίνεται στους σημασιολογικούς περιορισμούς της εφαρμογής.

(6) Εφαρμογή της κληρονομικότητας, όπου αυτό κρίνεται σκόπιμο και εφικτό. Αν κατά το βήμα της αναγνώρισης κλάσεων ακολουθήσαμε για όλες τις κλάσεις την προσέγγιση top-down, δηλαδή όλες οι κλάσεις αναγνωρίστηκαν αφού είχαν αναγνωριστεί οι γονικές τους, τότε εδώ θα έχουμε μια έτοιμη ιεραρχία κλάσεων. Αν, όμως, κατά το πρώτο βήμα κάποιες κλάσεις αναγνωρίστηκαν με προσέγγιση bottom-up, δηλαδή χωρίς να έχουν προηγουμένως αναγνωριστεί οι γονικές τους, τότε εδώ θα έχουμε μια ιεραρχία κλάσεων και κάποιες ακόμα κλάσεις που δε θα συνδέονται με την ιεραρχία αυτή. Έτσι, κάνουμε το έκτο βήμα του οποίου σκοπός είναι να αναγνωρίσει ομοιότητες μεταξύ των κλάσεων που έχουν οριστεί. Σε κάθε περίπτωση που εντοπίζονται τέτοιες ομοιότητες επιλέγεται από την ιεραρχία ή σχηματίζεται μια νέα κλάση-γονέας των κλάσεων που βρέθηκαν να μοιάζουν, και τα κοινά στοιχεία

των κλάσεων αυτών (χαρακτηριστικά ή μέθοδοι) μεταφέρονται στην κοινή τους πρόγονο. Η επεξεργασία αυτή γίνεται συνέχεια, την πρώτη φορά για τις αρχικές κλάσεις, τη δεύτερη φορά για τις κλάσεις αυτές και τις γονικές τους που σχηματίστηκαν στην προηγούμενη επανάληψη, κοκ. Γενικός σκοπός του βήματος αυτού είναι το να ελεγχθεί πώς, με εφαρμογή της κληρονομικότητας, μπορούν να οριστούν χαρακτηριστικές και λειτουργίες μία φορά σε ανώτερες κλάσεις, ώστε να αποφευχθεί ο επαναλαμβανόμενος ορισμός τους σε κατώτερες.

Τα βήματα αυτά επαναλαμβάνονται μέχρι το μοντέλο που θα προκύψει από το σχεδιασμό να καλύπτει όλες τις ανάγκες που προέκυψαν από το στάδιο της ανάλυσης.

Και άλλες περιγραφές για την object-oriented μεθοδολογία σχεδιασμού μπορούν να βρεθούν στη βιβλιογραφία ([BOO83] και [PIN88]), είναι όμως ισοδύναμες με την προηγούμενη.

Θα πρέπει να σημειωθεί εδώ το ότι η χρήση της object-oriented μεθοδολογίας σχεδιασμού δεν απαιτεί και τη χρήση μιας object-oriented γλώσσας προγραμματισμού κατά τη φάση της υλοποίησης: ο object-oriented σχεδιασμός έχει σκοπό να διευκολύνει τη φυσική απεικόνιση του πραγματικού κόσμου στο σχηματικό περιβάλλον του σχεδιασμού. Μπορούμε έτσι να χρησιμοποιήσουμε την object-oriented μεθοδολογία σχεδιασμού, και κατόπιν να γράψουμε τον απαιτούμενο κώδικα σε Pascal, C ή οποιαδήποτε όχι object-oriented γλώσσα. Από την άλλη πλευρά, η object-oriented μεθοδολογία δεν έρχεται να ανατρέψει τις αρχές σχεδιασμού που ξέραμε ως τώρα (όπως ο *δομημένος σχεδιασμός* (structured design), για παράδειγμα) αλλά να τις συμπληρώσει και να βελτιώσει τα σημεία στα οποία παρουσιάζουν αδυναμίες˙ αυτό φαίνεται από το ότι χρησιμοποιεί αρκετά από τα γνωστά εργαλεία σχεδιασμού, όπως, για παράδειγμα, τα *διαγράμματα ροής δεδομένων* (data flow diagrams). Τέλος, με τη φιλοσοφία του προσανατολισμού στα αντικείμενα αίρεται η διάκριση μεταξύ δεδομένων και διαδικασιών, και έτσι είναι δυνατό να έχουμε καλύτερης ποιότητας κώδικα και σημαντική απλοποίηση του περιβάλλοντος προγραμματισμού.

1.4. Γενικά συμπεράσματα

Ο object-oriented προγραμματισμός εμφανίστηκε σαν ιδέα στα μέσα της δεκαετίας του '70, και έκτοτε η ανάπτυξή του ακολούθησε πολύ γρήγορους ρυθμούς. Καθώς τα πράγματα εξελίσσονται και τα προβλήματα στο χώρο της ανάπτυξης λογισμικού γίνονται όλο και πιο σύνθετα, ο object-oriented προγραμματισμός συγκεντρώνει το ενδιαφέρον των προγραμματιστών λόγω των πολύ καλών χαρακτηριστικών του: αυξάνει τη δυνατότητα *επαναχρησιμοποίησης του κώδικα* (code reusability), επιτρέπει αφαιρέσεις τόσο σε επίπεδο δεδομένων όσο και σε επίπεδο λειτουργιών , διευκολύνει τη *μείωση της αλληλεξάρτησης* μεταξύ των *ενοτήτων* (modules) των προγραμμάτων, βελτιώνει τη *δυνατότητα τροποποίησης του κώδικα* (code modifiability), κλπ. Δε θα πρέπει βέβαια να παραβλέψουμε και κάποια αρνητικά χαρακτηριστικά του object-oriented προγραμματισμού, καθώς και κάποια ανοιχτά, προς το παρόν, σχετικά προβλήματα. Όπως αναφέρεται και στη σχετική βιβλιογραφία ([DYK89], [COX86]), χρειάζεται συχνά περισσότερος χρόνος για το πέρασμα μηνυμάτων σε ένα object-oriented περιβάλλον παρά για την απλή κλήση συναρτήσεων σε ένα συμβατικό σύστημα, και η διαφορά αυτή μειώνεται, προς όφελος της επικοινωνίας με μηνύματα, μόνο για τις σύνθετες εφαρμογές, ενώ το μέγεθος του εκτελέσιμου κώδικα είναι μεγαλύτερο αν η εφαρμογή αναπτυχθεί σε κάποια object-oriented γλώσσα προγραμματισμού. Επιπλέον, από την άποψη των προγραμματιστών είναι πολύ ευκολότερο να μάθει κανείς τις βασικές αρχές μιας συνηθισμένης γλώσσας προγραμματισμού, παρά να εξοικειωθεί με μια ολόκληρη βιβλιοθήκη από λογισμικό που έχει ήδη αναπτυχθεί μέσα σε ένα object-oriented προγραμματιστικό περιβάλλον.

Πέρα από αυτά, δεν έχουν δοθεί ακόμα οριστικές απαντήσεις σε μια σειρά από ερωτήματα όπως το αν είναι καλύτερη η απόδοση τύπου και χώρου μνήμης στα αντικείμενα κατά τη μεταγλώττιση ή κατά τη μετάφραση ενός προγράμματος, το αν είναι προτιμότερη η απλή ή η πολλαπλή κληρονομικότητα, καθώς και το αν η διαχείριση του χώρου που απελευθερώνεται μετά την καταστροφή αντικειμένων θα πρέπει να γίνεται από την object-oriented γλώσσα (π.χ. Smalltalk) ή από τον προγραμματιστή (π.χ. C++) ([DYK89]). Όλα αυτά δείχνουν ότι ο προσανατολισμός στα αντικείμενα, παρόλη τη θεωρητική του θεμελίωση, προσκρούει σε αρκετά σύνθετες τεχνικές δυσκολίες προκειμένου να υλοποιηθεί.

Ο object-oriented προγραμματισμός, ωστόσο, δεν είναι ένα εργαλείο που χρησιμοποιείται αποκομμένο από τις διάφορες τεχνικές ανάπτυξης λογισμικού, αλλά μπορεί να ενταχθεί σε άλλες μεθοδολογίες και να δώσει πολύ καλά αποτελέσματα. Ήδη υπάρχουν object-oriented περιβάλλοντα προγραμματισμού όπως το KEE ([DYK89]) και το LOOPS, συστήματα και γλώσσες διαχείρισης βάσεων δεδομένων με object-oriented χαρακτηριστικά (π.χ. Ontos και Opal αντίστοιχα) και λειτουργικά συστήματα βασισμένα στις

αρχές του object-oriented προγραμματισμού. Ακόμα, έχουν διατυπωθεί και πιο φιλόδοξα σχέδια που μιλούν για την ολοκλήρωση των βάσεων δεδομένων, του object-oriented προγραμματισμού, της τεχνητής νοημοσύνης και των συστημάτων hypermedia ([PAR89]). Αν και είναι πιθανό να περάσει αρκετός καιρός πριν οι προσπάθειες αυτές δώσουν ικανοποιητικά αποτελέσματα, είναι βέβαιο ότι ο object-oriented προγραμματισμός θα χρησιμοποιείται όλο και περισσότερο στο μέλλον, και θα κατακτήσει ένα μεγάλο μέρος της βιομηχανίας λογισμικού.

1.5. Βιβλιογραφία

1. [BER88] John Thomas Berry, "C++ Programming", The Waite Group, 1988.
2. [BOO83] Booch G., "Software Engineering with Ada", Benjamin-Cummings, 1983.
3. [COX86] Brad J. Cox, "Object-Oriented Programming: An Evolutionary Approach", Addison-Wesley, 1986.
4. [DYK89] R.P. Ten Dyke και J.C. Kunz, "Object-Oriented Programming", IBM Systems Journal, τεύχος 28/3, 1989.
5. [KAE86] Ted Kaehler και Dave Patterson, "A Taste of Smalltalk", W.W.Norton, 1986.
6. [LOR86] Lorensen W., "Object-Oriented Design", CRD Software Engineering Guidelines, General Electrics Co., 1986.
7. [PAR89] Kamran Parsaye, M. Chingel, S. Khoshafian και H. Wong, "Intelligent Databases", Wiley, 1989.
8. [PIN88] Lewis J. Pinson και Richard S. Wiener, "An Introduction to Object-Oriented Programming and Smalltalk", Addison-Wesley, 1988.
9. [PRE82] R.S. Pressman, "Software Engineering: A Practitioner's Approach", Mc Graw Hill, 1982.
10. [WEG87] P. Wegner, "Dimensions of Object-Based Language Design", πρακτικά OOPSLA'87, Οκτώβριος 1987.

2. HYPERTEXT ΚΑΙ ΣΥΣΤΗΜΑΤΑ MULTIMEDIA ΚΑΙ HYPERMEDIA - ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ HYPERMEDIA

2.1. 1. Ορισμός και εφαρμογές του hypertext και των συστημάτων multimedia και hypermedia

Οι βάσεις δεδομένων μας επιτρέπουν να χειριστούμε μόνο δομημένα δεδομένα˙ καθώς, ωστόσο, εξαπλώνονται οι εφαρμογές της Πληροφορικής, εμφανίζεται όλο και πιο πιεστική η ανάγκη να χειριστούμε μη αριθμητικά δεδομένα και δεδομένα χωρίς δόμηση. Έτσι, τα τελευταία χρόνια έχει αλλάξει ριζικά η αντίληψή μας για το πώς μπορούμε να αποθηκεύσουμε και να ανακτήσουμε κείμενο και γενικότερα πληροφορίες που δεν έχουν μια συγκεκριμένη δομή. Ένα αποφασιστικό βήμα προς αυτήν την κατεύθυνση ήταν η εμφάνιση της έννοιας του hypertext, η οποία συνέπεσε λίγο ή πολύ με την ανάπτυξη τεχνολογιών για την επεξεργασία δεδομένων διαφορετικών από κείμενο, όπως γραφικά, εικόνες, ήχο, video, κλπ. Η επέκταση της λογικής του hypertext σε δεδομένα multimedia οδήγησε στη δημιουργία των πρώτων συστημάτων hypermedia.

Η έννοια του hypertext μπορεί να περιγραφεί ως εξής:

> *Hypertext είναι ένα κείμενο του οποίου τα κομμάτια μπορούν να συνδεθούν χωρίς όμως να καταστραφεί η φυσική τους διάταξη. Τα κομμάτια ενός hypertext μπορούν να προσπελαστούν είτε σειριακά, είτε ακολουθώντας τις συνδέσεις που έχουν οριστεί μεταξύ τους.*

Στη σχετική βιβλιογραφία, τα συστήματα hypertext και hypermedia ορίζονται σχεδόν πάντα μαζί. Έτσι, διαβάζουμε ότι

> *Hypertext είναι ένα σύνολο από μη σειριακά συνδεδεμένα κομμάτια κειμένου ή άλλης πληροφορίας. Αν το κέντρο βάρους ενός τέτοιου συστήματος πέφτει σε μορφές πληροφοριών διαφορετικές από το κείμενο (π.χ. γραφικά, ήχο, κινούμενες εικόνες από βιντεοδίσκους, εκτελέσιμα προγράμματα) τότε συχνά αντί του όρου hypertext χρησιμοποιείται ο όρος hypermedia ([NIE90]).*

Τα συστήματα hypermedia ορίζονται συχνά με βάση τη δομή των αποθηκευμένων πληροφοριών. Ένας τέτοιος ορισμός είναι και ο ακόλουθος:

Hypermedia είναι ένα είδος από συστήματα για αναπαράσταση και διαχείριση πληροφοριών χτισμένα πάνω σε δίκτυα από multimedia κόμβους οι οποίοι ενώνονται με συνδέσμους καθορισμένων τύπων ([HAL88]).

Τα συστήματα hypertext και hypermedia έχουν γίνει αρκετά δημοφιλή πρόσφατα και χρησιμοποιούνται για την οργάνωση και τη διαχείριση ανομοιόμορφα δομημένων πληροφοριών, σε εφαρμογές που εκτείνονται από τη νομική έρευνα ως την τεχνολογία λογισμικού.

Η εκπαίδευση είναι ένας πολύ καλός χώρος εφαρμογών, γιατί συχνά βασίζεται σε προσομοίωση πραγματικών καταστάσεων, για την οποία οι μέθοδοι που χρησιμοποιούν μόνο γραφικά δεν είναι αρκετά ισχυρές. Η ιατρική και η αεροπορική εκπαίδευση ιδιαίτερα προσφέρουν τέτοιες εφαρμογές. Τέλος, στο χώρο της διδασκαλίας, όπου η συμβολή των οπτικών βοηθημάτων στην διαδικασία της μάθησης εθεωρείτο πάντα πολύτιμη, υπάρχουν ήδη πιλοτικά συστήματα hypermedia σε λειτουργία, όπως για παράδειγμα το Palenque ([RIP89]) και το "Tell Me Why" (FRE89]).

Η δημοτικότητα των συστημάτων hypermedia οφείλεται στην αύξηση της λειτουργικότητας των *ηλεκτρονικών εγγράφων* (electronic documents) που αποθηκεύονται σε αυτά. Τα συμβατικά ηλεκτρονικά έγγραφα αποτελούνται από κείμενα γραμμένα με επεξεργαστές και αναπαριστάνουν στατικές πληροφορίες, οι οποίες εμφανίζονται απομονωμένες από άλλες πληροφορίες, διαφορετικής μορφής (όπως πίνακες, γραφικά, εικόνες, κλπ.), που αναφέρονται στο ίδιο θέμα και μπορεί επίσης να αποθηκεύονται ηλεκτρονικά. Τα έγγραφα multimedia, αντίθετα, μπορούν να περιέχουν πληροφορίες οποιασδήποτε μορφής σχετικά με ένα συγκεκριμένα θέμα. Έτσι, μπορούμε να φανταστούμε ηλεκτρονικά έγγραφα multimedia που να περιέχουν όχι μόνο κείμενο και στατικά γραφικά και εικόνες, αλλά ακόμα κινούμενα γραφικά και εικόνες, video, ανθρώπινη ομιλία και οποιοδήποτε είδος ήχου. Εννοείται, βέβαια, ότι για την παρουσίαση ενός τέτοιου εγγράφου απαιτούνται εξειδικευμένα εργαλεία, τόσο από άποψη υλικού

(περιφερειακά), όσο και από άποψη λογισμικού (προγράμματα παρουσίασης)˙ σε αυτό το θέμα θα επανέλθουμε παρακάτω.

Αν έχουμε ένα σύνολο από ηλεκτρονικά έγγραφα multimedia, ασύνδετα μεταξύ τους, που μπορούν να προσπελαστούν από ένα κοινό user interface, τότε μιλάμε για ένα σύστημα multimedia. Το βασικό πλεονέκτημα ενός τέτοιου συστήματος είναι το ότι μπορούμε να έχουμε θεματικά συγκεντρωμένες πληροφορίες διαφορετικής μορφής. Η σημαντικότερη ιδιότητα, ωστόσο, που μας επιτρέπει να περάσουμε από ένα σύστημα multimedia σε ένα σύστημα hypermedia , είναι η δυνατότητα να συνδέσουμε οποιοδήποτε multimedia έγγραφα μεταξύ τους, ή οποιοδήποτε τμήμα πληροφορίας ενός multimedia εγγράφου με ένα οποιοδήποτε τμήμα πληροφορίας ενός άλλου multimedia εγγράφου. Έτσι, σχηματίζεται ένα δίκτυο από έγγραφα multimedia συνδεδεμένα μεταξύ τους, δηλαδή ένα σύστημα hypermedia. Οι *σύνδεσμοι* (links) που χρησιμοποιούνται σε ένα σύστημα hypermedia έχουν κυρίως έννοια σημασιολογική, δηλαδή δηλώνουν ότι οι multimedia πληροφορίες που συνδέουν, σχετίζονται με κάποια σημασιολογική συσχέτιση. Υπάρχει βέβαια και η περίπτωση να έχουμε συνδέσμους που να φανερώνουν ότι κάποιες multimedia πληροφορίες πρέπει να παρουσιάζονται μαζί, κοκ., αλλά κατά κανόνα οι συνδέσεις μεταξύ εγγράφων θα είναι σημασιολογικές. Επιπλέον, οι ίδιοι οι σύνδεσμοι μπορούν να περιέχουν πληροφορίες, όπως για παράδειγμα στοιχεία για την ακριβή τους σημασία και το σημασιολογικό τους βάρος.

Ένας εναλλακτικός τρόπος με τον οποίο μπορούν να συνδεθούν τα ηλεκτρονικά έγγραφα που υπάρχουν σε ένα σύστημα hypermedia, είναι με τη χρήση *λέξεων-κλειδιών* (keywords) που φανερώνουν με ποια θέματα είναι σχετικό το περιεχόμενο κάθε εγγράφου. Στη συνέχεια, εφόσον έχουν οριστεί λέξεις κλειδιά για όλα τα έγγραφα που περιέχει το σύστημα, μπορούμε να αναζητάμε έγγραφα που περιέχουν συγκεκριμένες λέξεις-κλειδιά, έγγραφα που περιέχουν τις ίδιες λέξεις-κλειδιά, κλπ.

Το γεγονός ότι ένα σύστημα hypermedia έχει τη δομή ενός δικτύου, του οποίου οι κόμβοι περιέχουν πληροφορίες και οι ακμές αντιστοιχούν σε σημασιολογικούς συνδέσμους μεταξύ των πληροφοριών αυτών, εισηγείται και μια συγκεκριμένη μέθοδο για την ανάκτηση πληροφορίας, την *περιπλάνηση* (navigation). Η λογική της περιπλάνησης είναι ότι ένας χρήστης του συστήματος hypermedia ξεκινάει από έναν κόμβο του δικτύου, ανακτά την πληροφορία που υπάρχει στον κόμβο αυτόν, και στη συνέχεια, βλέποντας τους συνδέσμους που φεύγουν από τον κόμβο αυτόν προς άλλους κόμβους, διαλέγει ένα σύνδεσμο και τον διασχίζει. Έτσι πηγαίνει σε έναν άλλο κόμβο του δικτύου, κοκ. Αν πρόκειται για ένα μεγάλο σύστημα, οι χρήστες μπορούν να διαλέγουν εκείνους τους κόμβους και εκείνους τους συνδέσμους που ταιριάζουν στα ενδιαφέροντά τους (οι κόμβοι μπορούν να επιλεγούν με βάση το περιεχόμενό τους και την θέση τους μέσα στη βάση ενώ οι σύνδεσμοι με βάση τον τύπο τους ή το περιεχόμενο των κόμβων που συνδέουν).

Η αντιμετώπιση των εγγράφων ως κόμβων ενός δικτύου οι οποίοι συνδέονται μεταξύ τους σημασιολογικά είναι ένα ιδιαίτερα εξυπηρετικό σχήμα για την υλοποίηση συστημάτων που

(1) περιέχουν μεγάλες ποσότητες πληροφοριών, ή/και

(2) παρουσιάζουν μεγάλες απαιτήσεις για τη σύνδεση των πληροφοριών αυτών.

Σε μια εγκυκλοπαίδεια ή σε ένα λεξικό, για παράδειγμα, τα λήμματα αναπαρίστανται καλύτερα ως συνδυασμοί κειμένου και "αντικειμένων" multimedia και τα διαφορετικά λήμματα μπορούν να ενώνονται με περισσότερο δυναμικό τρόπο από ότι με την απλή παράθεση κοινών λέξεων-κλειδιών. Αυτό το γεγονός είναι πολύ σημαντικό δεδομένου ότι ο κάθε χρήστης έχει και τις δικές του ανάγκες και απαιτήσεις και τα δικά του ενδιαφέροντα, και κατά συνέπεια θα χρησιμοποιήσει την εγκυκλοπαίδεια ή το λεξικό κατά διαφορετικό τρόπο.

Πριν προχωρήσουμε περισσότερο στην περιγραφή των συστημάτων hypertext και hypermedia, θα πρέπει να ασχοληθούμε με την ακριβή φύση των δεδομένων multimedia και με τις τεχνολογίες αποθήκευσης και παρουσίασης των δεδομένων αυτών.

2.2. Οπτική αποθήκευση δεδομένων multimedia

Στο [WEI89β] βρίσκουμε μερικούς τεχνικούς ορισμούς για τους όρους hypertext, multimedia και hypermedia. Σύμφωνα με το [WEI89β], τα δεδομένα multimedia ορίζονται ως

ένα μίγμα από δεδομένα διαφορετικής μορφής, όπως κείμενο, ήχο, video και δεδομένα από υπολογιστές, όλα συγκεντρωμένα σε ένα εύχρηστο μέσο αποθήκευσης.

Από αυτό τον ορισμό, και από το γεγονός ότι τα δεδομένα multimedia συνδυάζονται για να σχηματίσουν ολοκληρωμένα έγγραφα multimedia, ανακύπτουν ορισμένα ζητήματα αποθήκευσης των δεδομένων αυτών.

Ένα από τα πιο πιεστικά προβλήματα των σχετικών εφαρμογών προκύπτει από το μέγεθος και την πολυπλοκότητα των εγγράφων multimedia. Οι *οπτικοί δίσκοι ανάγνωσης μόνο* (CD-ROMs) εμφανίστηκαν για εφαρμογές multimedia που απαιτούσαν μαζική αποθήκευση μεγάλου όγκου μη ενημερώσιμων δεδομένων. Για εφαρμογές που χρησιμοποιούν πολλά ενημερώσιμα δεδομένα, αναπτύχθηκαν πρόσφατα τεχνικές αποθήκευσης σε δίσκους *μίας εγγραφής-πολλών αναγνώσεων* (write once-read many disks, συντομ. WORM disks), οι οποίες όμως παραμένουν ακόμα σε πειραματικό στάδιο.

Ακόμα και στην περίπτωση των μη ενημερώσιμων εγγράφων, η ύπαρξη των CD-ROMs δεν έλυσε όλα τα σχετικά προβλήματα. Σε πείσμα της μεγάλης τους χωρητικότητας, τα CD-ROMs αποδεικνύονται ανεπαρκή για να φυλάξουν το πλήθος των δεδομένων που περιέχει ένα κομμάτι video ή μουσικής με ποιότητα CD, και υπάρχουν ακόμα πολλές εφαρμογές οι οποίες ξεπερνούν τις δυνατότητες των προϊόντων CD ([WEI89β]).

Καθώς νέες εφαρμογές εμφανίζονται κάθε μέρα, γίνονται όλο και πιο απαραίτητες οι τεχνικές για εκτεταμένη *συμπίεση* (compression) και *αποσυμπίεση* (decompression) σε πραγματικό χρόνο του μεγάλου όγκου δεδομένων που χρησιμοποιείται στις περιπτώσεις αυτές.

Η τεχνική DVI ([RIP89]) προσφέρει μια λύση στο πρόβλημα της αποθήκευσης εικόνων. Χρησιμοποιώντας μια ειδική μέθοδο συμπίεσης, επιτυγχάνει να αυξήσει δραματικά τη συνολική χωρητικότητα του μέσου αποθήκευσης. Η μέθοδος αυτή βασίζεται στην παρατήρηση ότι το ανθρώπινο μάτι δεν επιμένει τόσο στη μεγάλη ανάλυση χρωμάτων, όσο στην ανάλυση της φωτεινότητας.

Πέρα από την τεχνική DVI, πολλές προσεγγίσεις στο πρόβλημα της αποθήκευσης μεγάλου όγκου δεδομένων έχουν εμφανιστεί, από τις οποίες, βέβαια, οι περισσότερες βασίζονται σε προϊόντα της τεχνολογίας CD όπως το CD-i, η *Εκτεταμένη Αρχιτεκτονική CD-ROM* (CD-ROM Extended Architecture) και το περιβάλλον CD-ROM XA ([WEI89α]). Για παράδειγμα, η IPA/AIM ανέπτυξε σε CD-i μια εφαρμογή για εκπαιδευτικούς σκοπούς, η οποία προορίζεται για παιδιά από 7 ως 13 ετών και χειρίζεται ήχο και video ([FRE89]).

Θα πρέπει πάντως να σημειωθεί ότι πολλές από τις τεχνικές που έχουν αναπτυχθεί, όπως η DVI, δεν ακολουθούν αυστηρά τις καθιερωμένες προδιαγραφές και, από την άλλη πλευρά, και οι ίδιες οι προδιαγραφές φαίνονται συχνά υπερβολικά πολυάριθμες για να τηρηθούν.

Παρά την ανάπτυξη των CDs, οι *αναλογικοί βιντεοδίσκοι* (analog videodisks) δεν έπεσαν σε αχρηστία. Ένα εξαιρετικό παράδειγμα εφαρμογής multimedia που χρησιμοποιεί αναλογικούς δίσκους είναι το σύστημα παρουσίασης μεσαιωνικών χειρογράφων στο Μουσείο Γκεττύ στο Λος Άντζελες. Το σύστημα αυτό καθοδηγεί με αλληλεπίδραση τους χρήστες από το ένα χειρόγραφο στο άλλο χρησιμοποιώντας στατικές εικόνες ([FRE89]). Μερικοί ειδικοί ισχυρίζονται ότι το υψηλό κόστος και η σχετικά χαμηλή ποιότητα παρουσίασης των συσκευών CD-ROM σε σύγκριση με τους αναλογικούς βιντεοδίσκους θα κάνει τον ανταγωνισμό ανάμεσα στις αναλογικές και ψηφιακές τεχνολογίες να συνεχιστεί για τα 6 επόμενα χρόνια.

Μια άλλη κατηγορία τεχνικών για την αύξηση της χωρητικότητας των μέσων αποθήκευσης είναι οι τεχνικές δέλτα και οι παραλλαγές τους (για παράδειγμα, αποθήκευση δειγματοληπτημένων ηχητικών δεδομένων μετά από κωδικοποίηση ADPCM με 8 δυαδικά ψηφία ανά δείγμα αντί για τα 16 δυαδικά ψηφία ανά δείγμα του συνηθισμένου PCM). Είναι κοινή η διαπίστωση ότι η παρουσίαση δεδομένων video δε θα ήταν δυνατή χωρίς τη χρήση τεχνικών δέλτα, καθώς απαιτεί παροχή περισσότερης πληροφορίας ανά δευτερόλεπτο από το μέγιστο ρυθμό μεταβίβασης δεδομένων (transfer rate) των οπτικών μέσων αποθήκευσης ([WEI89α]).

Πέρα από τα προβλήματα που δημιουργεί ο μεγάλος όγκος των δεδομένων multimedia και οι μεγάλες ταχύτητες με τις οποίες αυτά θα πρέπει να μεταφέρονται, πολλά ζητήματα ανακύπτουν και από το μεγάλο φάσμα των υπολογιστών (μικρά PCs, Commodore κλπ) που θα πρέπει να προσαρμοστούν σε σχετικές εφαρμογές και από τις φτωχές ικανότητες των υπολογιστών αυτών για παρουσίαση.

2.3. Απαιτήσεις για την ανάπτυξη εφαρμογών multimedia

Με βάση τα όσα αναφέρθηκαν ως τώρα για τα ιδιαίτερα χαρακτηριστικά των εφαρμογών multimedia και hypermedia, είναι εμφανές το ότι οποιοδήποτε σύστημα προορίζεται για τέτοιες εφαρμογές θα πρέπει να διαθέτει κάποια συγκεκριμένα χαρακτηριστικά. Μερικά από τα χαρακτηριστικά αυτά παρουσιάζονται παρακάτω:

(1) Δυνατότητα για ενιαία, ομογενή επεξεργασία ετερογενών πληροφοριών. Οι εξειδικευμένες εφαρμογές που ασχολούνται με ένα μόνο είδος πληροφοριών χρησιμοποιούν διαφορετικές τεχνικές για το χειρισμό εικόνων, video, ήχου, κειμένου και δομημένων δεδομένων από αρχεία βάσεων δεδομένων. Ο χειρισμός των δομημένων δεδομένων μπορεί να γίνει από τις *γεννήτριες αναφορών* (report generators) και τις *φόρμες* (forms) που παρέχουν τα περισσότερα συμβατικά συστήματα διαχείρισης βάσεων δεδομένων. Τα πακέτα αυτά, ωστόσο, δεν είναι κατάλληλα για να χειριστούν πληροφορίες χωρίς καμιά δομή. Το λογισμικό που αναλαμβάνει την επεξεργασία ακατέργαστων δεδομένων, όπως το κείμενο ή οι εικόνες, είναι εξειδικευμένο ως προς τη μορφή των δεδομένων αυτών. Οι *επεξεργαστές κειμένου* (text processors) και οι text formatters, για παράδειγμα, είναι σχεδιασμένοι για την παρουσίαση και τροποποίηση κειμένου. Οι *επεξεργαστές εικόνας* χειρίζονται εικόνες˙ μπορούν να χειριστούν και κείμενο αλλά, για να το κάνουν αυτό, επεξεργάζονται το κείμενο σαν έναν πίνακα δυαδικών ψηφίων (bitmap array), και μ' αυτόν τον τρόπο δεν μπορούν να κάνουν αλλαγές χωρίς να κληθεί μια συγκεκριμένη λειτουργία. Τέλος, ο ήχος καταγράφεται, υφίσταται επεξεργασία και αναπαράγεται με εξειδικευμένα προγράμματα, τα οποία συνήθως απαιτούν επίσης και εξειδικευμένο υλικό. Όλα αυτά τα εργαλεία επεξεργασίας θα πρέπει να συνδεθούν κατά τρόπο ενιαίο κάτω από ένα κοινό interface, επιτρέποντας στους χρήστες να προσπελάζουν δεδομένα διαφορετικής μορφής χωρίς να ξέρουν οποιεσδήποτε ειδικές λεπτομέρειες για τη δομή τους.

(2) Ευφυές user interface. Καθώς η περιοχή των εφαρμογών multimedia είναι καινούρια και περιλαμβάνει μάλλον πολύπλοκες εφαρμογές, θα πρέπει σε οποιοδήποτε στάδιο της αλληλεπίδρασης μεταξύ ενός τέτοιου συστήματος και των χρηστών να υπάρχουν δυνατότητες για βοήθεια κατά τη χρήση και οδηγίες για το τι πρέπει να γίνει στο επόμενο βήμα. Ιδιαίτερα χρήσιμο θα ήταν ένα interface για χρήστες χωρίς εμπειρία.

(3) Αλληλεπίδραση με τους χρήστες σε μια εύληπτη γλώσσα. Η γλώσσα αλληλεπίδρασης παίζει προφανώς έναν πολύ σημαντικό ρόλο για τους χρήστες. Έχει καταστεί πλέον κοινός τόπος μετά την ανάπτυξη τόσων εφαρμογών, ότι οι χρήστες καταλαβαίνουν μια γλώσσα κοντά στη φυσική πολύ καλύτερα από οποιοδήποτε είδος κωδικοποιημένων μηνυμάτων. Η πολυπλοκότητα των εφαρμογών και η ετερογένεια των δεδομένων multimedia καθιστούν την ανάπτυξη μιας "φυσικής" γλώσσας γενικής χρήσης πιο δύσκολη αλλά ταυτόχρονα και περισσότερο αναγκαία.

(4) Δυνατότητα για ομοιογενή και συντονισμένη παρουσίαση των εγγράφων multimedia. Όπως έχουμε ήδη πει, τα έγγραφα μιας εφαρμογής multimedia θα αποτελούνται γενικά από κείμενο, γραφικά, εικόνες, ήχο, video και δομημένα δεδομένα, και τα στοιχεία αυτά θα συνδέονται μεταξύ τους τόσο σημασιολογικά όσο και χρονικά, με την έννοια ότι πρέπει να παρουσιάζονται άλλες φορές σειριακά και άλλες φορές ταυτόχρονα. Έτσι, οι συσχετίσεις ανάμεσα στα τμήματα ενός εγγράφου multimedia πρέπει να εντοπίζονται και να αναπαράγονται κατά την παρουσίαση στο χρήστη. Από την άλλη πλευρά, η παρουσίαση των multimedia εγγράφων δεν μπορεί να γίνει, γενικά, με μία και μόνο συσκευή (ένα έγγραφο, για παράδειγμα, το οποίο περιέχει γραφικά και ήχο δεν μπορεί να παρουσιαστεί από ένα μόνο περιφερειακό). Έτσι, πρέπει να υπάρχουν εργαλεία παρουσίασης τα οποία να ελέγχουν και να συντονίζουν την παρουσίαση των διαφόρων τμημάτων ενός εγγράφου multimedia στις κατάλληλες συσκευές εξόδου καλώντας το αντίστοιχο εξειδικευμένο λογισμικό.

(5) Δυνατότητα για αποθήκευση μεγάλου όγκου πληροφοριών. Το μέγεθος ενός εγγράφου multimedia είναι αρκετές τάξεις μεγαλύτερο από το μέγεθος ενός συμβατικού εγγράφου (περισσότερες πληροφορίες μπορούν να βρεθούν στο [WEI89β]). Μέσα στο ίδιο έγγραφο πρέπει να αποθηκεύονται video, κείμενο, ήχος, δομημένα δεδομένα κλπ., μαζί με τις χρονικές και σημασιολογικές τους συσχετίσεις. Επιπλέον, τα έγγραφα multimedia πρέπει να μπορούν να υποστούν επεξεργασία κατά τρόπο αποδοτικό. Αυτοί οι παράγοντες φανερώνουν την αναγκαιότητα συσκευών *μαζικής αποθήκευσης* (mass storage) που να εξασφαλίζουν και μεγάλη χωρητικότητα αλλά και γρήγορη προσπέλαση. Η ανάπτυξη μέσων οπτικής αποθήκευσης είναι το πιο σημαντικό βήμα που έχει γίνει ως τώρα για να αντιμετωπιστεί αυτό το πρόβλημα.

Θα πρέπει να τονιστεί ότι τα παραπάνω χαρακτηριστικά απαιτούνται από οποιοδήποτε σύστημα είναι κατασκευασμένο για εφαρμογές multimedia, χωρίς όμως αυτό να σημαίνει ότι τα χαρακτηριστικά αυτά καλύπτουν και τις ανάγκες των εφαρμογών hypermedia.

2.4. Συστήματα multimedia, hypertext και hypermedia

Η ερευνητική δραστηριότητα στον τομέα των συστημάτων multimedia, hypertext και hypermedia έχει ξεκινήσει εδώ και αρκετά χρόνια, με αποτέλεσμα μέχρι σήμερα να έχουν αναπτυχθεί πολυάριθμα τέτοια συστήματα, πειραματικά ή εμπορικά, για εξειδικευμένες εφαρμογές ή για γενική χρήση. Είναι σκόπιμο να αναφερθούν μερικά αντιπροσωπευτικά παραδείγματα τέτοιων συστημάτων, αφενός για να φανεί η εξέλιξή τους και αφετέρου για να καταδειχθούν οι περιορισμοί που εξακολουθούν να υφίστανται ακόμα και σήμερα. Η απαρίθμηση που ακολουθεί παρακάτω ξεκινά από τα συστήματα με τις λιγότερες και καταλήγει στα συστήματα με τις περισσότερες δυνατότητες. Είναι, ωστόσο εμφανές, ότι ακόμα και τα πιο σύγχρονα συστήματα hypermedia δεν είναι τόσο ισχυρά στη δομή τους όσο θα έπρεπε να είναι, ούτε υποστηρίζουν όλες τις μορφές δεδομένων που θα έπρεπε να υποστηρίζουν.

2.4.1. Συστήματα multimedia

Τα συστήματα που παρουσιάζονται εδώ μπορούν να χειριστούν πληροφορίες διαφορετικής μορφής αλλά δεν είναι σχεδιασμένα για να τις συνδέουν, με την έννοια που έχει η σύνδεση πληροφοριών σε περιβάλλον hypertext ή hypermedia. Και τα τρία συστήματα που αναφέρονται έχουν σχεδιαστεί για εφαρμογές ηλεκτρονικής επικοινωνίας.

(1) DARPA EMMS (Experimental Multimedia Mail System) ([REY85]) Σύστημα multimedia για *ηλεκτρονικό ταχυδρομείο* (electronic mail) σε *περιβάλλον γραφείου* (office environment). Αναπτύχθηκε στο Πανεπιστήμιο της Νότιας Καλιφόρνιας υπό την αιγίδα της DARPA (Defense Applied Research Projects Agency) και μπήκε σε πειραματική χρήση από το 1983. Το EMMS τρέχει σε περιβάλλον δικτύου, κάτω από το πρωτόκολλο ARPA Internet. Οποιοιδήποτε χρήστες του δικτύου μπορούν μέσα από το EMMS να ανταλλάξουν έγγραφα multimedia που να περιέχουν κείμενο, γραφικά και ομιλία, και υποστηρίζονται όλες οι τυπικές *λειτουργίες ταχυδρομείου* (mailing functions).

(2) DIMS (Distributed Interoffice Mail System) ([SAK85]) Πειραματικό (1985) σύστημα multimedia για ηλεκτρονικό ταχυδρομείο. Αναπτύσσεται από τη NEC για *περιβάλλον εταιρείας* (corporate environment). Υποστηρίζει όλες τις τυπικές λειτουργίες ταχυδρομείου και απαιτεί την ύπαρξη ισχυρών σταθμών εργασίας.

(3) CCWS (Command and Control Workstation System) ([POG85]) Σύστημα multimedia για ηλεκτρονικό ταχυδρομείο και τηλεσυνδιαλέξεις (teleconferencing). Αναπτύχθηκε για στρατιωτικές εφαρμογές από την SRI International και το 1985 λειτουργούσε ήδη σε πραγματικές συνθήκες. Το CCWS είναι ένα κατανεμημένο σύστημα και τρέχει σε ένα δίκτυο από πολλούς και ισχυρούς σταθμούς εργασίας. Ο κάθε σταθμός διαθέτει μια τοπική multimedia βάση δεδομένων. Μπορεί να διακινηθεί κείμενο, γραφικά και ομιλία, και έχουν αναπτυχθεί editors και για τις τρεις μορφές πληροφοριών. Οι τοπικές multimedia βάσεις δεδομένων μπορούν να προσπελαστούν με ερωτήσεις σε φυσική γλώσσα.

2.4.2. Συστήματα hypertext

Τα συστήματα που αναφέρονται εδώ υποστηρίζουν μόνο κείμενο ή/και γραφικά. Διαθέτουν όλα τη δομή του hypertext και ποικίλλουν ως προς τους μηχανισμούς με τους οποίους συνδέονται και μπορούν να ανακτηθούν οι πληροφορίες. Μια ειδική περίπτωση είναι το RUBRIC, το οποίο δεν μπορεί να θεωρηθεί ως σύστημα hypertext, αλλά προσφέρει τεχνικές που μπορούν να μεταφερθούν αυτούσιες σε hypertext περιβάλλον.

(1) RUBRIC ([CUN85]) Σύστημα για *ανάκτηση πληροφοριών με βάση κανόνες* (rule-based information retrieval). Αναπτύχθηκε από την Advanced Information & Decision Systems (1983). Το RUBRIC συνεργάζεται με μια βάση δεδομένων που περιέχει κείμενα χωρίς δομή, και επιτρέπει να οριστούν λέξεις-κλειδιά για κάθε κείμενο της βάσης. Στη συνέχεια μπορούν να οριστούν κριτήρια υπό τη μορφή PROLOG-κανόνων ως προς τις επιθυμητές λέξεις-κλειδιά και να επιλεγούν τα κείμενα που ανταποκρίνονται περισσότερο στα κριτήρια αυτά. Οι τεχνικές του RUBRIC μπορούν να μεταφερθούν αυτούσιες σε συστήματα hypertext και hypermedia.

(2) FRESS (File Retrieval and Editing System) ([YAN85]) Σύστημα hypertext που αναπτύχθηκε και λειτούργησε σε ακαδημαϊκό περιβάλλον, στο Πανεπιστήμιο Brown, από το 1960. Επιτρέπει να

οριστούν λέξεις-κλειδιά πάνω στα διαθέσιμα έγγραφα και να δημιουργηθούν σύνδεσμοι μεταξύ εγγράφων.

(3) Neptune ([TAN89α]) Σύστημα hypertext που δημιουργήθηκε από τους Delisle και Schwartz (1986). Λειτουργεί σε περιβάλλον δικτύου, είναι σχεδιασμένο για πολλούς χρήστες, και χρησιμοποιεί δοσοληψίες για να διατηρεί την ακεραιότητα και τη συνέπεια των δεδομένων. Για τα έγγραφα και τους συνδέσμους μεταξύ εγγράφων στο Neptune μπορούν να οριστούν ζευγάρια *χαρακτηριστικών/τιμών* (attribute/value pairs) και κάθε χρήστης μπορεί να βλέπει μόνο το υποδίκτυο που τον ενδιαφέρει με βάση τα ζευγάρια αυτά.

(4) Guide ([TAN89α]) Σύστημα hypertext που αναπτύχθηκε στο Πανεπιστήμιο του Kent. Υποστηρίζει κείμενο και γραφικά που αποθηκεύονται σε κόμβους, οι οποίοι στη συνέχεια ενώνονται με συνδέσμους. Οι σύνδεσμοι αυτοί μπορούν να σχηματίζουν ιεραρχίες κόμβων ή να καταλήγουν σε εξωτερικά εκτελέσιμα προγράμματα. Παρέχεται στους χρήστες δυνατότητα οπισθοχώρησης.

(5) HyperTies ([TAN89α]) Σύστημα hypertext που σχεδιάστηκε στο Πανεπιστήμιο του Maryland (ολοκληρώθηκε το 1987). Υποστηρίζει κείμενο και γραφικά, επιτρέπει να δημιουργηθούν σύνδεσμοι και διαθέτει μηχανισμούς περιπλάνησης και αναζήτησης για την ανάκτηση πληροφοριών.

(6) TextVision ([KOM89] και [TAN89α]) Σύστημα hypertext που δημιουργήθηκε πρόσφατα στο Πανεπιστήμιο του Twente στην Ολλανδία (Piet Kommers και άλλοι, 1988). Το TextVision δίνει ιδιαίτερο βάρος στη σημασιολογική σύνδεση των πληροφοριών σε ένα γράφο, παρέχει πολύ καλά εργαλεία εποπτείας και προορίζεται για εκπαιδευτικές εφαρμογές. Το σύστημα αυτό εισήγαγε για πρώτη φορά την έννοια της *κεντρικότητας των κόμβων* (node centrality) που υπολογίζεται αναδρομικά και υποδεικνύει το πόσο σημαντικός είναι ένας κόμβος, με βάση το πλήθος των συνδέσμων που δείχνουν στον κόμβο αυτόν. Οι χρήστες μπορούν να αλλάζουν εύκολα τη δομή του δικτύου hypertext.

(7) HyperNet ([KOM89]) Σύστημα hypertext με πολλές ομοιότητες με το TextVision. Παρέχει ισχυρά εποπτικά μέσα και εργαλεία για το χειρισμό της δομής του δικτύου hypertext. Υποστηρίζει όψεις σε διάφορα επίπεδα λεπτομέρειας πάνω στο δίκτυο και προκρίνει την περιπλάνηση ως βασική διαδικασία ανάκτησης πληροφοριών.

(8) HyperCard ([APP87] και [TAN89α]) Δημοφιλές εμπορικό σύστημα hypertext που αναπτύχθηκε από την Apple για υπολογιστές MacIntosh. Οι πληροφορίες στο HyperCard αποθηκεύονται σε κάρτες. Οι θεματικά σχετικές κάρτες σχηματίζουν στοίβες τις οποίες μπορούν οι χρήστες να διασχίζουν. Το HyperCard διαθέτει μια δική του γλώσσα χειρισμού δεδομένων, την HyperTalk. Δεν υποστηρίζει συνδέσμους αλλά επιτρέπει να δημιουργηθούν μέσα στο περιεχόμενο κάθε κάρτας κουμπιά και να οριστεί για κάθε κουμπί ένα script της HyperTalk, το οποίο θα εκτελείται όταν πατιέται το κουμπί αυτό. Έτσι, αντί για συνδέσμους έχουμε κουμπιά των οποίων τα scripts περιέχουν απλώς εντολές ανάκτησης συγκεκριμένων καρτών.

2.4.3. 4.3. Συστήματα hypermedia

Εδώ αναφέρονται τα περισσότερο εξελιγμένα συστήματα, δηλαδή εκείνα που υποστηρίζουν τα περισσότερα από τα χαρακτηριστικά που πρέπει θεωρητικά να διαθέτει ένα σύστημα hypermedia. Θα πρέπει, ωστόσο, να σημειωθεί ότι και τα τέσσερα συστήματα που παρουσιάζονται εδώ υποστηρίζουν μόνο οπτικές μορφές πληροφορίας, δηλαδή κείμενο, δομημένα δεδομένα, και γραφικά ή/και ψηφιοποιημένη εικόνα. Έτσι, αποφεύγουν σημαντικά προβλήματα ανομοιομορφίας και συγχρονισμού που θα δημιουργούσε η επεξεργασία και η παρουσίαση ήχου ή video.

(1) BALSA (Brown Algorithm Simulator and Animator) ([YAN85]) Περιβάλλον με χαρακτηριστικά συστήματος hypermedia για την ανάπτυξη εκπαιδευτικού υλικού. Υποστηρίζει την επεξεργασία κειμένου και γραφικών, επιτρέπει τη δημιουργία συνδέσμων και τον ορισμό φίλτρων για τις διαθέσιμες πληροφορίες, τα οποία λειτουργούν όπως οι όψεις στις βάσεις δεδομένων. Το 1985 βρισκόταν σε κανονική χρήση.

(2) EDS (Electronic Document System) ([YAN85]) Σύστημα hypermedia που δημιουργήθηκε στο Πανεπιστήμιο του Brown. Ολοκληρώθηκε το 1982, και η σχεδίασή του στηρίχθηκε κυρίως στα συμπεράσματα από το FRESS. Υποστηρίζει την επεξεργασία κειμένου και γραφικών και τη δημιουργία συνδέσμων.

(3) Intermedia ([GARR86], [SMI87], [YAN85] και [TAN89α]) Περιβάλλον με χαρακτηριστικά hypermedia, το οποίο έχει σχεδιαστεί ως πλατφόρμα για τη συνεργασία εφαρμογών και εργαλείων για δεδομένα διαφορετικής μορφής. Στα έγγραφα του Intermedia μπορεί να αποθηκευτεί κείμενο, γραφικά, δεδομένα από λογιστικά φύλλα και άλλα. Υποστηρίζεται η δημιουργία συνδέσμων και ο ορισμός λέξεων-κλειδιών, τόσο για τα έγγραφα όσο και για τους συνδέσμους του συστήματος. Το Intermedia παρέχει τοπικούς και γενικούς χάρτες του hypermedia δικτύου και γενικά δίνει μεγάλο βάρος στο user interface. Έχει αναπτυχθεί στο Πανεπιστήμιο Brown (ολοκληρώθηκε το 1986), και ως εκ τούτου διαθέτει όλα τα χαρακτηριστικά των προγενέστερων συστημάτων FRESS και EDS. Είναι σχεδιασμένο για περιβάλλον πολλών χρηστών και διαθέτει έναν ισχυρό μηχανισμό για τον έλεγχο της σύγχρονης προσπέλασης. Αν και το Intermedia έχει γραφτεί σε Inheritance C (μια object-oriented διάλεκτο της C), χρησιμοποιεί το INGRES, με προοπτική να συνεργαστεί στο μέλλον με μια object-oriented βάση δεδομένων.

(4) NoteCards ([HAL88] και [TAN89α]) Σύστημα hypermedia για κείμενο, γραφικά και εικόνες. Αναπτύχθηκε στο Xerx PARC (Halasz και άλλοι) και ολοκληρώθηκε το 1987. Οι πληροφορίες στο NoteCards αποθηκεύονται σε κάρτες και μεταξύ των καρτών μπορούν να δημιουργηθούν σύνδεσμοι. Τόσο οι κάρτες όσο και οι σύνδεσμοι έχουν συγκεκριμένους τύπους. Οι χρήστες μπορούν να δουν τη δομή του δικτύου hypertext και να την αλλάξουν απευθείας με έναν ειδικό editor. Υποστηρίζεται η περιπλάνηση και σε περιορισμένη κλίμακα η αναζήτηση κειμένου. Ένα μεγάλο πλεονέκτημα του NoteCards είναι οι δυνατότητές του για επέκταση.

(5) KMS (Knowledge Management System) ([AKS88] και [TAN89α]) Εμπορικό σύστημα hypermedia για κείμενο, γραφικά και ψηφιοποιημένες εικόνες. Ολοκληρώθηκε το 1987 και αποτελεί την εμπορική εκδοχή του προγενέστερου συστήματος ZOG (1984). Είναι σχεδιασμένο για τη δημιουργία βάσεων δεδομένων σε εταιρικό περιβάλλον. Οι πληροφορίες στο KMS αποθηκεύονται σε πλαίσια (το ανάλογο των καρτών του HyperCard και των εγγράφων των άλλων συστημάτων), και μπορούν να δημιουργηθούν σύνδεσμοι μεταξύ πλαισίων και να σχηματιστούν ιεραρχίες από πλαίσια. Οι σύνδεσμοι μπορούν να δείχνουν και σε εξωτερικά προγράμματα, τα οποία θα ενεργοποιούνται όταν οι σύνδεσμοι διασχίζονται. Το KMS δεν παρέχει στους χρήστες γραφική εποπτεία του δικτύου hypertext αλλά διαθέτει απεριόριστες ιδιότητες για ακύρωση ενεργειών και οπισθοχώρηση. Επίσης δεν έχει μηχανισμούς διαχείρισης και ο έλεγχος για σύγχρονη προσπέλαση στα πλαίσια από τα οποία αποτελείται η βάση δεδομένων είναι χαλαρός. Γενικά, το KMS δίνει ένα υποσύνολο των χαρακτηριστικών των συστημάτων hypertext και hypermedia, αλλά είναι ένα σύστημα καλά περιορισμένο και λειτουργεί αποδοτικά.

2.5. Συστήματα hypermedia και hypermedia βάσεις δεδομένων

Από την προηγούμενη αναφορά συστημάτων hypertext και hypermedia, είναι εύκολο να εντοπιστούν ορισμένα χαρακτηριστικά που μπορούν να θεωρούνται λίγο ή πολύ καθιερωμένα για τα συστήματα αυτά. Στη βιβλιογραφία γίνεται λόγος για την πρώτη γενιά συστημάτων hypermedia, και αναφέρεται ότι τα χαρακτηριστικά ενός μέσου συστήματος της γενιάς αυτής είναι τα εξής ([HAL88]):

(1) οι κόμβοι του δικτύου hypermedia έχουν συγκεκριμένο τύπο (κείμενο, γραφικά κλπ.), και υπάρχει μια ιεραρχία από τύπους δεδομένων

(2) οι κόμβοι είναι απλοί, δηλαδή δεν μπορούν να περιέχουν άλλους κόμβους

(3) οι σύνδεσμοι μεταξύ των κόμβων είναι αμφικατευθυνόμενοι και δεν έχουν τύπους

(4) οι σύνδεσμοι ξεκινάνε από και καταλήγουν σε είτε ολόκληρους κόμβους είτε περιοχές μέσα σε κόμβους

(5) παρέχονται γενικές όψεις της δομής του δικτύου

(6) η δομή του δικτύου μπορεί να αλλάξει με απευθείας χειρισμούς των διαγραμμάτων που την παριστάνουν

(7) υποστηρίζονται ειδικά τα ιεραρχικά δίκτυα

(8) το user interface χρησιμοποιεί πολλαπλά παράθυρα και οι εντολές δίνονται είτε με ποντίκι είτε με μενού

(9) υποστηρίζεται αναζήτηση κειμένου

(10) υποστηρίζονται πολλοί χρήστες αλλά με περιορισμένο έλεγχο σύγχρονης προσπέλασης

(11) δεν τηρούνται εκδοχές των αποθηκευμένων πληροφοριών

(12) τα δεδομένα αποθηκεύονται σε συνηθισμένα αρχεία ή σε σχεσιακές βάσεις δεδομένων.

Έτσι, γίνεται φανερό ότι τα συστήματα hypermedia διαθέτουν αρκετά εξελιγμένα χαρακτηριστικά, και υποστηρίζουν την επεξεργασία και την ανάκτηση δεδομένων παρόμοια με τις κλασικές βάσεις δεδομένων. Σε καμιά περίπτωση, ωστόσο, ένα σύστημα hypermedia το οποίο διαθέτει τις δυνατότητες που περιγράφτηκαν παραπάνω και μόνο, δεν μπορεί να χαρακτηριστεί ως hypermedia βάση δεδομένων. Για να μιλήσουμε για hypermedia βάσεις δεδομένων, θα πρέπει να εξασφαλίσουμε ότι υποστηρίζονται λειτουργίες όπως επεξεργασία δοσοληψιών, ανάκτηση δεδομένων με ερωτήσεις, τήρηση εκδοχών, ορισμός όψεων, επανόρθωση σε περίπτωση βλαβών κλπ. Σε επόμενα κεφάλαια θα ορίσουμε ένα μοντέλο δεδομένων για συστήματα hypermedia και μηχανισμούς που να υλοποιούν κάτω από το προτεινόμενο μοντέλο τις λειτουργίες αυτές, ώστε να μπορεί τελικά να θεωρηθεί ότι έχουμε μια hypermedia βάση δεδομένων.

Οι hypermedia βάσεις δεδομένων τώρα, έστω και αν βρίσκονται ως ένα σημείο στα σχέδια, εμφανίζονται ως διάδοχοι των συμβατικών βάσεων δεδομένων και φιλοδοξούν να αποδειχθούν πολύ πιο ευέλικτες και ισχυρές από τις τελευταίες όσον αφορά την αποθήκευση και την ανάκτηση των πληροφοριών. Η βασική διαφορά ανάμεσα σε μια βάση δεδομένων hypermedia και σε μια συμβατική βάση δεν είναι το ευρύτερο φάσμα πληροφοριών που μπορεί η πρώτη να αποθηκεύσει αλλά μάλλον το γεγονός ότι, η συμβατική βάση περιέχει δομημένες πληροφορίες, ενώ η βάση hypermedia περιέχει διασυνδεδεμένες πληροφορίες.

Μια βάση δεδομένων hypermedia πρέπει να επιτύχει δύο βασικούς σκοπούς:

(1) Όπως και μια συμβατική βάση, πρέπει να καταστήσει το σύνολο ή ένα μεγάλο μέρος των αποθηκευμένων πληροφοριών έτοιμο για ανάκτηση και επεξεργασία από τους τελικούς χρήστες.

(2) Αντίθετα από μια συμβατική βάση, όπου οι συνδέσεις μεταξύ των δεδομένων καθορίζονται έμμεσα (στο σχεσιακό μοντέλο από το γεγονός ότι τα δεδομένα βρίσκονται στην ίδια σχέση ή ότι υπάρχει κάποιο πεδίο με το ίδιο όνομα και πεδίο ορισμού σε περισσότερες από μία σχέσεις, και στο δικτυωτό και το ιεραρχικό μοντέλο από το γεγονός ότι τα δεδομένα βρίσκονται στον ίδιο κόμβο ή ότι υπάρχει σύνδεση μεταξύ τους, η οποία όμως δεν είναι άμεσα ορατή στους χρήστες), πρέπει να καταστήσει το σύνολο ή ένα μεγάλο μέρος των συνδέσεων μεταξύ των αποθηκευμένων πληροφοριών έτοιμο για ανάκτηση και επεξεργασία από τους τελικούς χρήστες.

Οι χρήστες των συστημάτων hypermedia τείνουν να ανακτούν, να προσθέτουν, να διαγράφουν, να αλλάζουν και να εξερευνούν τις συνδέσεις μεταξύ των πληροφοριών τουλάχιστον τόσο συχνά όσο κάνουν τα αντίστοιχα πράγματα και για τις ίδιες τις πληροφορίες. Στον περίπλοκο κόσμο που ζούμε, οι συσχετίσεις των πληροφοριών φαίνονται πιο σημαντικές και από τις ίδιες τις πληροφορίες.

Είναι φανερό, επομένως, ότι ένα μοντέλο δεδομένων για βάσεις hypermedia πρέπει να εξασφαλίζει την ίδια λειτουργικότητα τόσο για τις αποθηκευμένες πληροφορίες όσο και για τις συνδέσεις μεταξύ των πληροφοριών αυτών. Ένα μοντέλο σχεδιασμού που εξασφαλίζει τη σημασιολογική ισορροπία ανάμεσα στις έννοιες και στις συσχετίσεις μεταξύ εννοιών, είναι ο γράφος. Ένας γράφος αποτελείται από κόμβους και ακμές, που είναι ανεξάρτητες οντότητες της ίδιας σπουδαιότητας. Έτσι, το σχήμα αυτό φαίνεται κατάλληλο για να μοντελοποιήσει μια βάση δεδομένων hypermedia.

Μια βάση δεδομένων hypermedia, λοιπόν, ορίζεται ως ένα γράφος αποτελούμενος από κόμβους που αποθηκεύουν multimedia δεδομένα και ακμές που συνδέουν τους κόμβους αυτούς. Στα επόμενα, αντί του όρου "κόμβος" θα χρησιμοποιείται ο όρος "item" για να δηλώσει την αντίστοιχη οντότητα του μοντέλου δεδομένων και αντί του όρου "ακμή" θα χρησιμοποιείται ο όρος "σύνδεσμος" ανάλογα. Οι όροι "γράφος" και "δίκτυο" θα χρησιμοποιούνται ισοδύναμα για να δηλώσουν την όλη δομή της βάσης δεδομένων. Με τη μοντελοποίηση αυτή και τα items και οι σύνδεσμοι τοποθετούνται στο ίδιο επίπεδο από άποψη λειτουργικότητας και θεωρούνται ισοδύναμες οντότητες.

Θα πρέπει να σημειωθεί ότι η δομή του γράφου είναι μια άμεση λύση για τη μοντελοποίηση συστημάτων hypermedia και χρησιμοποιείται σε ένα πολύ μεγάλο μέρος της σχετικής βιβλιογραφίας. Μια διαφωτιστική συζήτηση για μοντελοποίηση με γράφους και τους σχετικούς περιορισμούς μπορεί να βρεθεί στο [KOM89], όπου τονίζεται και η σπουδαιότητα της δήλωσης τύπων και βαρών για τις συνδέσεις μεταξύ multimedia πληροφοριών, χαρακτηριστικό το οποίο το προτεινόμενο μοντέλο διαθέτει, όπως θα φανεί από την περιγραφή του σε επόμενα κεφάλαια.

2.6. Βιβλιογραφία

1. [AKS88] Robert M. Akscyn, Donald L. McCracken και Elise A. Yoder, "KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations", CACM, τεύχος 31/7, Ιούλιος 1988.
2. [APP87] Apple Computers, "HyperCard: User's Guide", Apple Computers, 1987.
3. [CUN85] Brian P. Mc Cune, Richard M. Tong, Jeffrey S. Dean και Daniel G. Shapiro, "RUBRIC: A System for Rule-Based Information Retrieval", IEEE Transactions on Software Engineering, τεύχος SE-11/9, Σεπτέμβριος 1985.
4. [FRE89] K. A. Frenkel, "The Next Generation of Interactive Technologies", CACM, τεύχος 32/7, Ιούνιος 1989.
5. [GARR86] L. Garrett, K. Smith και N. Meyrowitz, "Intermedia: Issues, Strategies and Tactics in the Design of a Hypermedia Document System", πρακτικά CSCW'86, Δεκέμβριος 1986.
6. [HAL88] Frank G. Halasz, "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems", CACM, τεύχος 31/7, Ιούλιος 1988.
7. [KOM89] P. A. M. Kommers, I.A. Ferreira και I. E. Jonker, "HYPERTEXT and Conceptual Mapping", SAFE/HYP/6/HCI-del/mk/fw/lr/pk, Σεπτέμβριος 1989.
8. [NIE90] J. Nielsen, "The Art of Navigating through Hypertext", CACM, τεύχος 33/3, Μάρτιος 1990.
9. [POG85] A. Poggio, J. Garcia Luna Aceves, E. Craighill, D. Morgan, L. Aguilar, D. Worthington και J. Hight, "CCWS: A Computer-Based, Multimedia Information System", IEEE Computer, Οκτώβριος 1985.
10. [REY85] Joyce K. Reynolds, Jonathan B. Postel, Alan R. Katz, Greg G. Finn και Annette L. DeSchon, "The DARPA Experimental Multimedia Mail System", IEEE Computer, Οκτώβριος 1985.
11. [RIP89] G. D. Ripley, "DVI - A Digital Multimedia Technology", CACM, τεύχος 32/7, Ιούνιος 1989.
12. [SAK85] Shiro Sakata και Tetsuo Ueda "A Distributed Interoffice Mail System", IEEE Computer, Οκτώβριος 1985.
13. [SMI87] Karen E. Smith και Stanley B. Zdonic, "Intermedia: A Case Study of the Differences Between Relational and Object-Oriented Database Systems", πρακτικά OOPSLA'87, 1987.
14. [TAN89α] Gary Tanner, "Navigating Through Hyperchaos", SAFE/HYP/HCI-pap/GT240789, Ιούλιος 1989.
15. [WEI89α] F. Weimar, "Storage Techniques for Hypermedia Material", HYP/6 First Intermediate Report on Hypermedia Structures, SAFE/HYP/6/HCI-del/mk/fw/lr/pk, 1989.
16. [WEI89β] F. Weimar, SAFE/HYP/Phil-Rep/FW06, September 1989.
17. [YAN85] Nicole Yankelowich, Norman Meyrowitz και Andries van Dam, "Reading and Writing the Electronic Book", IEEE Computer, Οκτώβριος 1985.

3. OBJECT-ORIENTED ΚΑΙ ΣΧΕΣΙΑΚΟΣ ΣΧΕΔΙΑΣΜΟΣ ΓΙΑ HYPERMEDIA ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

3.1. Απαιτήσεις σχεδιασμού για hypermedia βάσεις δεδομένων

Όπως έχει φανεί από την ως τώρα ανάλυση, οι κλασικές βάσεις δεδομένων δεν έχουν αρκετά πλούσια δομή για να υποστηρίξουν τα χαρακτηριστικά που πρέπει να έχουν τα βάσεις δεδομένων hypermedia. Έτσι, στο κεφάλαιο αυτό αναδεικνύονται ορισμένες απαιτήσεις σχεδιασμού που πρέπει να καλύπτει μια βάση δεδομένων hypermedia ([WOE86] και [SMI87]). Οι απαιτήσεις που απαριθμούνται παρακάτω αφορούν

(1) τις ειδικές ανάγκες που δημιουργεί ο χειρισμός δεδομένων multimedia

(2) τις ειδικές ανάγκες που δημιουργεί η υποστήριξη της δομής hypermedia, και

(3) τις γενικές ανάγκες που υπάρχουν σε μια οποιαδήποτε βάση δεδομένων.

Οι απαιτήσεις που προκύπτουν από τους δύο πρώτους παράγοντες απαιτούν τη χρήση ενός μοντέλου σχεδιασμού με συγκεκριμένες ιδιότητες, ενώ οι απαιτήσεις της τρίτης κατηγορίας ικανοποιούνται με οποιοδήποτε από τα καθιερωμένα μοντέλα σχεδιασμού.

Οι απαιτήσεις που δημιουργεί ο χειρισμός δεδομένων multimedia είναι οι ακόλουθες:

(1α) Χρήση γενικευμένων *δομών μοντελοποίησης* (modelling constructs).

Οι πληροφορίες multimedia που αποθηκεύονται σ' ένα σύστημα hypermedia δεν μπορούν να περιγραφτούν κατά τρόπο ενιαίο καθώς είναι διαφορετικής μορφής, και έτσι δεν μπορούν όλες να δεχθούν δόμηση στον ίδιο βαθμό. Για παράδειγμα, έχουμε ήχο και κινούμενο video, που μπορούν να δομηθούν πολύ χαλαρά, αλλά έχουμε επίσης και φόρμες κειμένου με ισχυρή οργάνωση. Το μοντέλο δεδομένων θα πρέπει να παρέχει γενικευμένες δομές μοντελοποίησης που να ικανοποιούν και τις δύο ακραίες περιπτώσεις.

(1β) Υποστήριξη αφαιρέσεων.

Με αφαιρέσεις είναι δυνατό πληροφορίες multimedia διαφορετικής μορφής να μπορούν να συνυπάρξουν μέσα στο ίδιο πλαίσιο.

(1γ) Υποστήριξη γενικεύσεων.

Με τη χρήση γενικεύσεων σχηματίζεται μια ιεραρχία τύπων κόμβων, στην οποία κάθε τύπος έχει τα χαρακτηριστικά και τις ιδιότητες όλων των υπερτύπων του, και κάποια ακόμα. Έτσι θα μπορεί να δηλωθεί ότι ένα καινούριο είδος κόμβων multimedia πρέπει να εφοδιαστεί με τις ιδιότητες και τα χαρακτηριστικά ενός από τα είδη που προϋπάρχουν.

(1δ) Υποστήριξη κανόνων σχετικά με την παρουσίαση των πληροφοριών.

Μπορεί ένας χρήστης να θέλει οι κόμβοι που δημιουργεί να μην παρουσιάζονται με τον προκαθορισμένο τρόπο που ακολουθεί το σύστημα, αλλά διαφορετικά. Αυτό θα πρέπει να μπορεί να δηλωθεί μέσα στη βάση των δεδομένων.

(1ε) Αποθήκευση πληροφοριών σχετικά με τη φυσική αποθήκευση ενός εγγράφου.

Η ανάγκη αυτή γίνεται ακόμα πιο αισθητή όταν ετερογενή δεδομένα πρέπει να παρουσιαστούν ταυτόχρονα, ή ακόμα περισσότερο, όταν οι χρήστες έχουν το δικαίωμα να αλλάξουν ορισμένα χαρακτηριστικά της παρουσίασης με σκοπό να αναδείξουν ή να ερευνήσουν κάποιες συγκεκριμένες ιδιότητες των παρουσιαζόμενων πληροφοριών.

Οι απαιτήσεις που δημιουργεί η hypermedia δομή της βάσης δεδομένων είναι οι ακόλουθες:

(2α) Υποστήριξη συσχετίσεων μεταξύ διαφορετικών κόμβων.

Οι τύποι αυτών των συσχετίσεων δεν μπορούν πάντα να οριστούν εκ των προτέρων, καθώς ένας κόμβος μπορεί να συσχετίζεται με οποιοδήποτε άλλον με κριτήρια που να αφορούν γενίκευση, κοινή τιμή κάποιων χαρακτηριστικών, κοινές ιδιότητες ή οποιοδήποτε άλλη σχέση μπορεί να αντιληφθεί ένας χρήστης. Η βάση δεδομένων θα πρέπει να μπορεί να υποστηρίξει τέτοιες συσχετίσεις οι οποίες εμπεριέχουν και σημασιολογική πληροφορία.

(2β) Υποστήριξη πολλαπλών συσχετίσεων.

Μπορεί να υπάρχουν πολλαπλές συσχετίσεις, με διαφορετικές σημασίες, μεταξύ των ίδιων κόμβων πληροφορίας. Το μοντέλο δεδομένων θα πρέπει να επιτρέπει και τις πολλαπλές συσχετίσεις.

(2γ) Υποστήριξη ιεραρχικών συσχετίσεων.

Πολύ συχνά η σημασία μιας συσχέτισης είναι τέτοια ώστε η συσχέτιση αυτή θα πρέπει να είναι ιεραρχική, να σχηματίζει δηλαδή ένα δέντρο ή, στη γενικότερη περίπτωση, έναν ακυκλικό γράφο. Ο χρήστης θα πρέπει να έχει τη δυνατότητα να δηλώσει ότι κάποια συσχέτιση έχει τέτοιες ιδιότητες, και η βάση δεδομένων θα πρέπει να μπορεί στη συνέχεια να τις διατηρήσει.

(2δ) Δυνατότητα για δυναμική τροποποίηση των συσχετίσεων.

Οι συσχετίσεις μεταξύ των κόμβων ορίζονται στη βάση δεδομένων σύμφωνα με τις απόψεις των χρηστών για το πώς πρέπει να συνδεθούν οι αποθηκευμένες πληροφορίες και, κατά συνέπεια, είναι εξαιρετικά επιρρεπείς σε αλλαγές. Το σύστημα hypermedia θα πρέπει να διευκολύνει τη δημιουργία, τροποποίηση και αναζήτηση των συσχετίσεων.

Τέλος, απαιτήσεις σχεδιασμού που μπαίνουν σε μια οποιαδήποτε βάση δεδομένων είναι οι ακόλουθες:

(3α) Υποστήριξη περιορισμών σχετικά με την τυπική δομή ενός κόμβου.

Για παράδειγμα, μπορεί ένας χρήστης να απαιτεί να έχουν μια συγκεκριμένη δομή οι multimedia κόμβοι που δημιουργεί.

(3β) Υποστήριξη περιορισμών ως προς το ποιες τιμές είναι δεκτές για τα χαρακτηριστικά ενός κόμβου και το ποιες λειτουργίες μπορούν να εκτελεστούν πάνω στον κόμβο αυτόν.

(3γ) Δυνατότητα για δυναμική επέκταση του σχήματος της βάσης.

Αν κάποιος χρήστης θέλει να ορίσει έναν καινούριο τύπο κόμβων, θα πρέπει να μπορεί να προσθέσει τον ορισμό του τύπου αυτού στο σχήμα της βάσης δεδομένων.

(3δ) Ομοιόμορφη αντιμετώπιση των δεδομένων και του κώδικα στο επίπεδο της διαχείρισης της βάσης των δεδομένων.

Στη συνέχεια συγκρίνονται το σχεσιακό και το object-oriented μοντέλο σχεδιασμού για να φανεί ποιο από τα δύο ταιριάζει καλύτερα στην ανάπτυξη μιας βάσης δεδομένων hypermedia. Προκαταβολικά σημειώνεται ότι ο object-oriented σχεδιασμός καλύπτει τις περισσότερες από τις παραπάνω απαιτήσεις, χωρίς όμως αυτό να σημαίνει ότι και το συγκεκριμένο object-oriented μοντέλο δεδομένων που περιγράφεται σε επόμενο κεφάλαιο καλύπτει όλες τις απαιτήσεις που διαπιστώνεται εδώ ότι καλύπτονται από τον object-oriented σχεδιασμό. Το προτεινόμενο μοντέλο καλύπτει κάποιες από τις απαιτήσεις αυτές, τις βασικότερες, αλλά όπως θα φανεί μπορεί εύκολα να αλλάξει ή/και να επεκταθεί ώστε να καλύψει και τις υπόλοιπες.

3.2. Σύγκριση object-oriented και σχεσιακού σχεδιασμού ως προς τις απαιτήσεις

Ο σχεσιακός και ο object-oriented σχεδιασμός για βάσεις δεδομένων μπορούν να συγκριθούν ως προς τις απαιτήσεις που διατυπώθηκαν στην προηγούμενη παράγραφο.

Ως προς τις απαιτήσεις που προκύπτουν από το χειρισμό των δεδομένων multimedia ισχύουν τα εξής:

(1α) Χρήση γενικευμένων δομών μοντελοποίησης.

Το σχεσιακό μοντέλο υποστηρίζει μια μοναδική δομή για την αποθήκευση των πληροφοριών, τη σχέση, της οποίας όμως τα προκαθορισμένα, αναγκαστικά, χαρακτηριστικά, μπορεί να είναι ακατάλληλα για την πλειοψηφία των εφαρμογών.

Το object-oriented μοντέλο παρέχει μια εξαιρετικά δυναμική δομή μοντελοποίησης, την *κλάση* , της οποίας τα χαρακτηριστικά μπορούν να αλλάζουν για κάθε καινούρια εφαρμογή.

Είναι σημαντικό να σημειωθεί ότι το σχεσιακό μοντέλο μπορεί πολύ εύκολα να προσομοιωθεί από το object-oriented, αν οριστεί ως ρίζα της ιεραρχίας των κλάσεων μια κλάση Relation, η οποία να συμπεριφέρεται ακριβώς όπως οι συνηθισμένες σχέσεις του σχεσιακού μοντέλου.

(1β) Υποστήριξη αφαιρέσεων.

Τόσο στο σχεσιακό όσο και στο object-oriented μοντέλο, για να δημιουργήσουμε αφαιρέσεις, δηλαδή είδη κόμβων για τους οποίους δεν ορίζεται εκ των προτέρων τι multimedia πληροφορία θα περιέχουν, πρέπει μέσα στους κόμβους αυτούς να αποθηκεύουμε όχι τις ίδιες τις πληροφορίες multimedia, αλλά δείκτες προς αυτές.

(1γ) Υποστήριξη γενικεύσεων.

Το σχεσιακό μοντέλο δεν μπορεί να υποστηρίξει ιεραρχίες από γενικεύσεις κλάσεων, και γενικότερα δεν μπορεί να υποστηρίξει κληρονομικότητα ιδιοτήτων. Αν μια σχέση πρέπει να έχει και τα πεδία μιας ή κάποιων άλλων, αυτά θα πρέπει να επαναληφθούν στον ορισμό της καινούριας σχέσης.

Το object-oriented μοντέλο επιτρέπει να οριστούν ιεραρχίες κλάσεων και υποστηρίζει κληρονομικότητα ιδιοτήτων από τον ορισμό του. Έτσι, μπορεί να δηλωθεί έμμεσα το ότι μια κλάση θα διαθέτει και τις μεθόδους και μεταβλητές κάποιων άλλων κλάσεων.

(1δ) Υποστήριξη κανόνων σχετικά με την παρουσίαση των πληροφοριών.

Στο σχεσιακό μοντέλο, οι ίδιες ρουτίνες χειρίζονται όλες τις σχέσεις. Κατά συνέπεια, δεν μπορεί να οριστεί διαφορετικός τρόπος παρουσίασης για κάποιες σχέσεις, ούτε, πολύ περισσότερο, για κάποιες πλειάδες κάποιων σχέσεων.

Στο object-oriented μοντέλο, κάθε κλάση έχει τις δικές της μεθόδους. Έτσι, μπορούν να οριστούν διαφορετικές μέθοδοι παρουσίασης ανά κλάση. Επιπλέον, οι μέθοδοι παρουσίασης μιας κλάσης μπορούν να κληρονομηθούν από τις υποκλάσεις της και να τροποποιηθούν σε αυτές.

(1ε) Αποθήκευση πληροφοριών σχετικά με τη φυσική αποθήκευση ενός εγγράφου.

Και στα δύο μοντέλα μπορούμε να κρατήσουμε τέτοιες πληροφορίες μέσα σε πλειάδες σχέσεων/στιγμιότυπα κλάσεων. Επειδή όμως οι πληροφορίες αυτές το πιθανότερο είναι να μην έχουν την ίδια πάντα μορφή, είναι φανερό ότι η αποθήκευσή τους θα είναι ευκολότερη στο object-oriented μοντέλο, όπου η δόμηση είναι χαλαρότερη. Επιπλέον, στο μοντέλο αυτό, τέτοιες πληροφορίες μπορούν να κληρονομηθούν.

Ως προς τις απαιτήσεις που προκύπτουν από τη hypermedia δομή των βάσεων δεδομένων ισχύουν τα εξής:

(2α) Υποστήριξη συσχετίσεων μεταξύ διαφορετικών κόμβων.

Στο σχεσιακό μοντέλο, για να δηλώσουμε ότι δύο είδη πληροφοριών συσχετίζονται, είτε τις αποθηκεύουμε στην ίδια σχέση (οπότε, αν η συσχέτιση δεν είναι 1:1, έχουμε πρόβλημα κανονικοποίησης), είτε τις αποθηκεύουμε σε δυο ξεχωριστές σχέσεις και φτιάχνουμε μια τρίτη σχέση για τη συσχέτιση, η οποία ορίζεται τώρα έμμεσα από την ύπαρξη πεδίων με το ίδιο πεδίο ορισμού και, ενδεχομένως, το ίδιο όνομα στις τρεις αυτές σχέσεις. Αυτή η τακτική αυξάνει την πολυπλοκότητα του σχήματος, ιδιαίτερα αν το πλήθος των συσχετίσεων που πρέπει να αποθηκευτούν είναι μεγάλο. Επιπλέον, αν οι συσχετίσεις υλοποιούνται ως ξεχωριστές σχέσεις τότε μπορούν να δηλωθούν σημασιολογικά στοιχεία αλλά δεν μπορούν να επιβληθούν σημασιολογικοί περιορισμοί.

Σε ένα object-oriented μοντέλο οι συσχετίσεις μπορούν να υλοποιηθούν άμεσα ως ξεχωριστά αντικείμενα που κρατάνε τις δικές τους πληροφορίες. Έτσι είναι δυνατό αφενός να δηλώσουμε σημασιολογικά στοιχεία και αφετέρου να επιβάλλουμε αντίστοιχους περιορισμούς, ορίζοντας ότι κάθε φορά που οι περιορισμοί αυτοί παραβιάζονται, οι συσχετίσεις θα τροποποιούνται ανάλογα.

(2β) Υποστήριξη πολλαπλών συσχετίσεων.

Αν στο σχεσιακό μοντέλο όλες οι συσχετίσεις παριστάνονται με σχέσεις τότε οι πολλαπλές συσχετίσεις, δηλαδή οι συσχετίσεις 1:n ή n:m, μπορούν να παρασταθούν χωρίς προβλήματα κανονικοποίησης, αφού για κάθε επιπλέον πλειάδα μιας σχέσης Β με την οποία σχετίζεται μια πλειάδα της σχέσης Α θα προστίθεται απλώς ένα ακόμα ζεύγος κλειδιών στη σχέση ΑΒ.

Στο object-oriented μοντέλο, αφού οι συσχετίσεις υλοποιούνται και αυτές, όπως έχουμε ήδη πει, ως αντικείμενα (σύνδεσμοι), μπορούν χωρίς πρόβλημα να παρασταθούν πολλαπλές συσχετίσεις.

(2γ) Υποστήριξη ιεραρχικών συσχετίσεων.

Αν έχουμε μια ιεραρχία κόμβων, δηλαδή ένα δέντρο ή έναν ακυκλικό γράφο από κόμβους, οποιαδήποτε κι αν είναι η σημασιολογική της αξία θα πρέπει να κρατήσουμε το λιγότερο ποιος κόμβος είναι πατέρας ποιου και το περισσότερο ποιος κόμβος είναι πρόγονος ποιου.

Στο σχεσιακό μοντέλο μπορούμε να ξέρουμε μόνο ποιος κόμβος είναι παιδί ποιου γιατί μπορούμε να κρατήσουμε μόνο ένα επίπεδο της αντίστοιχης ιεραρχίας. (Όλα τα επίπεδα δεν μπορούν να κρατηθούν σε μια σχέση, αφενός γιατί δεν ξέρουμε πόσα είναι, ώστε να ορίσουμε στη σχέση ισάριθμα πεδία, και αφετέρου γιατί από το δεύτερο επίπεδο και μετά θα επαναλαμβάναμε παππού και πατέρα για κάθε εγγόνι και θα είχαμε πρόβλημα κανονικοποίησης). Στην ειδική, ωστόσο, περίπτωση που η ιεραρχία που έχουμε σχηματίζει ένα ή περισσότερα δέντρα, οπότε σε όλα τα δέντρα κάθε κόμβος εκτός της μοναδικής ρίζας θα έχει έναν και μόνο πρόγονο, μπορούμε να ορίσουμε μια σχέση "Πρόγονος", που να περιέχει τα πεδία "Πρόγονος", "Απόγονος" και "Βαθμός Συγγένειας" (η δομή του δέντρου μας εξασφαλίζει ότι από τον πρόγονο στον απόγονο θα υπάρχει ένα και μόνο μονοπάτι και

έτσι ο βαθμός συγγένειας θα είναι μονοσήμαντα ορισμένος· σε ένα γενικό ακυκλικό γράφο δε θα ίσχυε κάτι τέτοιο). Το τελευταίο πεδίο θα αντιπροσωπεύει το πλήθος των άμεσων απογόνων που παρεμβάλλονται μεταξύ των items που αναφέρονται στα δύο πρώτα πεδία (0 για άμεσο απόγονο, 1 για σχέση παππού/εγγονιού, κλπ.) Η λύση αυτή, όμως, παραβιάζει την αρχή του σχεσιακού μοντέλου σχεδιασμού να χρησιμοποιείται μία σχέση για κάθε λογική συσχέτιση, χρησιμοποιεί κάποια "τεχνητά" δεδομένα και απαιτεί την αποθήκευση μεγάλης ποσότητας πλεονάζουσας δευτερογενούς πληροφορίας και καθιστά πολύπλοκες τις λειτουργίες της εισαγωγής και της διαγραφής μιας συσχέτισης μεταξύ αντικειμένων, γιατί θα πρέπει να γίνουν κατά τρόπο τέτοιο που να εξασφαλίζει την διατήρηση της συνέπειας της σχέσης. Έτσι, όποια τεχνάσματα και αν χρησιμοποιήσουμε, στο σχεσιακό μοντέλο οι ιεραρχίες χρειάζεται τελικά να πλατύνουν.

Στο object-oriented μοντέλο αντίθετα, μπορούμε να έχουμε άμεση προσπέλαση στους απογόνους μέσα από δείκτες, και οι ιεραρχίες δε χρειάζεται να πλατύνουν.

Κατά συνέπεια, για να ανακτηθεί μια ολόκληρη ιεραρχία στο σχεσιακό μοντέλο χρειάζεται ένα πλήθος ερωτήσεων, οι οποίες μάλιστα απαιτούν την φύλαξη ενδιαμέσων αποτελεσμάτων και την ύπαρξη επαναληπτικών διαδικασιών που δεν καλύπτονται από το σχεσιακό μοντέλο αλλά απαιτούν την μεσολάβηση κάποιας αλγοριθμικής γλώσσας στην οποία οι ερωτήσεις θα εμφυτεύονται, οι οποίες πρέπει να εκτελεστούν με μια προκαθορισμένη σειρά που δεν μπορεί να βελτιστοποιήσει το DBMS.

Σε ένα object-oriented μοντέλο αντίθετα, μια ιεραρχία μπορεί να ανακτηθεί με μια μόνο ερώτηση που επιστρέφει τη ρίζα της και από εκεί και πέρα χρησιμοποιεί εσωτερικό πέρασμα μηνυμάτων για να πάρει τα παιδιά. Ακόμα, εφόσον οι ιεραρχίες δεν αποσυντίθενται για να αποθηκευτούν, δεν υπάρχει και η ανάγκη να ανακατασκευαστούν πριν από την παρουσίασή τους· μπορούν να ανακτηθούν και να δοθούν στους τελικούς χρήστες με τη μορφή που είχαν από την αρχή.

(2δ) Δυνατότητα για δυναμική τροποποίηση των συσχετίσεων.

Τόσο σε ένα σχεσιακό μοντέλο όπου όλες οι συσχετίσεις παριστάνονται ως σχέσεις, όσο και σε ένα object-oriented μοντέλο όπου όλες οι συσχετίσεις παριστάνονται ως αντικείμενα, είναι δυνατό να δημιουργήσουμε, να διαγράψουμε και να τροποποιήσουμε συσχετίσεις.

Ως προς τις γενικές απαιτήσεις για βάσεις δεδομένων ισχύουν τα εξής:

(3α) Υποστήριξη περιορισμών σχετικών με την τυπική δομή ενός κόμβου.

Οι απαιτήσεις για την τυπική δομή των κόμβων είναι δύσκολο να δηλωθούν σε ένα σχεσιακό μοντέλο όπου, λόγω του αυστηρού μαθηματικού υπόβαθρου, μία μόνο επιτρεπτή δομή μπορεί να οριστεί για κάθε είδος κόμβων.

Σε ένα object-oriented μοντέλο μπορούν να δηλωθούν εναλλακτικές επιτρεπτές δομές για τους κόμβους μιας κλάσης.

(3β) Υποστήριξη περιορισμών για αποδεκτές τιμές και λειτουργίες.

Σε ένα σχεσιακό σχήμα μπορούν να δηλωθούν περιορισμοί ακεραιότητας για τις τιμές που εκχωρούνται στα πεδία κάποιων σχέσεων, αλλά δεν μπορεί με κανέναν τρόπο να δηλωθεί ποιες ρουτίνες θα χρησιμοποιούνται για το χειρισμό της κάθε σχέσης.

Σε ένα object-oriented μοντέλο μπορεί να περιληφθούν οι ίδιοι περιορισμοί ορθότητας στις τιμές που εκχωρούνται στις μεταβλητές των κλάσεων και να εξασφαλιστεί μέσα από τους μηχανισμούς περιχαράκωσης και κληρονομικότητας ότι κάθε κλάση υφίσταται επεξεργασία μόνο με τις μεθόδους που έχουν οριστεί για αυτήν και τις προγόνους της.

(3γ) Δυνατότητα για δυναμική τροποποίηση του σχήματος της βάσης.

Τόσο ο σχεσιακός όσο και ο object-oriented σχεδιασμός επιτρέπουν επεκτάσεις του σχήματος της βάσης δεδομένων. Το object-oriented μοντέλο, ωστόσο, διευκολύνει τέτοιες επεκτάσεις ελαχιστοποιώντας, μέσω της κληρονομικότητας των ιδιοτήτων, τον όγκο της καινούριας δουλειάς που απαιτείται. Από την άλλη πλευρά, στο σχεσιακό μοντέλο, ο ορισμός νέων σχέσεων δεν απαιτεί *επαναμεταγλώττιση* (recompilation) του σχήματος (π.χ. SQL), ενώ στο object-oriented μοντέλο ο ορισμός νέων κλάσεων κατά την ώρα της λειτουργίας του συστήματος είναι πιθανό να απαιτήσει ολική ή μερική επαναμεταγλώττιση, ή να μην την απαιτήσει καθόλου, ανάλογα με την υλοποίηση του συστήματος (η Smalltalk-80, για παράδειγμα, αποφεύγει την επαναμεταγλώττιση διότι χρησιμοποιεί διερμηνευτή).

(3δ) Ομοιόμορφη αντιμετώπιση των δεδομένων και του κώδικα.

Σε ένα σχεσιακό DBMS τα δεδομένα αποθηκεύονται στις σχέσεις της βάσης και ο κώδικας διατηρείται στον πυρήνα του DBMS ή σε βιβλιοθήκες. Με αυτόν τον τρόπο, τα δεδομένα και ο

κώδικας αντιμετωπίζονται διαφορετικά από το RDBMS. Αυτή η διαφορετική αντιμετώπιση απαιτεί και διαφορετικούς μηχανισμούς για να υποστηριχθούν, για παράδειγμα, η τήρηση εκδοχών και η επανόρθωση. Επίσης, καθιστά πολυπλοκότερη τη σχεδίαση του DBMS και υποβαθμίζει τη συνολική απόδοσή του.

Σε ένα object-oriented DBMS τόσο τα δεδομένα όσο και ο κώδικας αποθηκεύονται ως αντικείμενα της βάσης. Έτσι, αντιμετωπίζονται και τα δύο από το OODBMS με τον ίδιο γενικά τρόπο και οι ίδιοι μηχανισμοί μπορούν να χρησιμοποιηθούν και στις δύο περιπτώσεις για τήρηση εκδοχών, επανόρθωση και αρκετά άλλα διαχειριστικά καθήκοντα. Αυτή η ομοιομορφία απλοποιεί το σχεδιασμό του DBMS και αυξάνει σημαντικά την απόδοσή του.

3.3. Συμπεράσματα

Η παραπάνω σύγκριση δείχνει ότι το σχεσιακό και το object-oriented μοντέλο μπορούν να ικανοποιήσουν τις ίδιες γενικά απαιτήσεις, αλλά το object-oriented μοντέλο υπερτερεί στα σημεία. Οι μηχανισμοί για ορισμό κλάσεων και κληρονομικότητα του μοντέλου αυτού επιτρέπουν το σχεδιασμό ενός σχήματος βάσης δεδομένων που να φιλοξενεί όλα τα ετερογενή αντικείμενα ενός περιβάλλοντος multimedia. Ο μηχανισμός για το πέρασμα μηνυμάτων επιτρέπει τη δυναμική αλληλεπίδραση των αντικειμένων με το περιβάλλον τους και μεταφέρει τον έλεγχο για τις τιμές εκχώρησης και τις απαιτούμενες λειτουργίες στα ίδια τα επηρεαζόμενα αντικείμενα. Τέλος, οι συσχετίσεις μεταξύ των κόμβων πληροφορίας μπορούν να δηλωθούν ως ξεχωριστά αντικείμενα με δικές τους ιδιότητες. Οι κλασικές απαιτήσεις των βάσεων δεδομένων τώρα μπορούν επίσης να ικανοποιηθούν με ένα object-oriented μοντέλο, παρά το γεγονός ότι το σχεσιακό μοντέλο είναι παλιότερο και κατά συνέπεια προσφέρει ασφαλέστερες τεχνικές.

Με λίγα λόγια, ένα object-oriented μοντέλο μπορεί να ανταποκριθεί συνολικά στις ανάγκες του σχεδιασμού χρησιμοποιώντας λίγο ή πολύ απλούς μηχανισμούς και επιτυγχάνοντας, με αυτόν τον τρόπο, ένα ευέλικτο σχήμα με αυθαίρετο βαθμό περιπλοκότητας.

Η διαπίστωση αυτή είχε ως αποτέλεσμα τα περισσότερα από τα συστήματα hypermedia που αναπτύχθηκαν πρόσφατα να βασιστούν σε object-oriented γλώσσες όπως οι Smalltalk, C++, Objective-C, Objective-Lisp, Flavors κλπ., και στις περιπτώσεις που απαιτήθηκε ο σχεδιασμός μιας βάσης δεδομένων για την υποστήριξη μιας εφαρμογής hypermedia, να υιοθετηθεί η object-oriented προσέγγιση.

Πρέπει ωστόσο να παρατηρηθεί ότι η προηγούμενη σύγκριση αφορά κυρίως τη φάση σχεδιασμού από ολόκληρο τον κύκλο της ανάπτυξης ενός συστήματος hypermedia. Από τη στιγμή που δημιουργείται ένα συγκεκριμένο object-oriented μοντέλο για ένα τέτοιο σύστημα, το μοντέλο αυτό μπορεί να υλοποιηθεί χρησιμοποιώντας ένα σχεσιακό σύστημα διαχείρισης βάσης δεδομένων με ορισμένες επεκτάσεις, ή τουλάχιστον υιοθετώντας ορισμένες τεχνικές που χρησιμοποιούνται στις σχεσιακές βάσεις δεδομένων.

Η σχεσιακή ή η ενισχυμένη με σχεσιακά στοιχεία υλοποίηση επιτρέπει να λυθούν εύκολα πολλά σύνθετα προβλήματα που αφορούν έλεγχο σύγχρονης προσπέλασης, διατήρηση ορθότητας, επανόρθωση, τήρηση εκδοχών κλπ., χρησιμοποιώντας καθιερωμένες μεθοδολογίες και εργαλεία. Από την άλλη πλευρά, ωστόσο, η τακτική αυτή έχει τα μειονεκτήματα που ήδη αναφέρθηκαν. Οι υπεύθυνοι για την ανάπτυξη του hypermedia βάσης δεδομένων μπορεί να συναντήσουν μεγάλες δυσκολίες στην προσπάθειά τους να επεκτείνουν ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων με σκοπό να υποστηρίξουν τέτοια εξωτικά χαρακτηριστικά όπως, για παράδειγμα, η κληρονομικότητα, ενώ μερικά ενδογενή προβλήματα των σχεσιακών συστημάτων μπορεί να αποδειχθούν ακόμα και άλυτα.

Αν βέβαια την εποχή της υλοποίησης υπάρχει ένα εμπορικά διαθέσιμο object-oriented σύστημα διαχείρισης βάσεων δεδομένων, θα ήταν ευχής έργον να υλοποιηθεί το object-oriented μοντέλο στο προϊόν αυτό.

Όπως κι αν έχουν τα πράγματα, δύο βασικά συμπεράσματα προέκυψαν ως τώρα σχετικά με τη φάση του σχεδιασμού: το πρώτο είναι ότι το σύστημα hypermedia θα πρέπει να σχεδιαστεί χρησιμοποιώντας την object-oriented μοντελοποίηση, και το δεύτερο είναι ότι το σύστημα hypermedia θα πρέπει να μοντελοποιηθεί ως ένα δίκτυο από items και συνδέσμους.

Στη συνέχεια, περιγράφεται ο προτεινόμενος object-oriented σχεδιασμός για μια βάση δεδομένων hypermedia μοντελοποιημένη ως δίκτυο από items και συνδέσμους.

3.4. Βιβλιογραφία

1. [SMI87] Karen E. Smith και Stanley B. Zdonic, "Intermedia: A Case Study of the Differences Between Relational and Object-Oriented Database Systems", πρακτικά OOPSLA'87, 1987.
2. [WOE86] D. Woelk, W. Kim και W. Luther "An Object-Oriented Approach to Multimedia Databases", ACM 0-89791-191-1/86/0500/0311, 1986.

4. ΜΟΝΤΕΛΟ ΔΕΔΟΜΕΝΩΝ ΓΙΑ HYPERMEDIA ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Στο κεφάλαιο αυτό προτείνεται ένα object-oriented μοντέλο δεδομένων για hypermedia βάσεις δεδομένων. Το προτεινόμενο μοντέλο ακολουθεί το καθιερωμένο object-oriented σχήμα, δηλαδή απαιτεί απλή και όχι πολλαπλή κληρονομικότητα, και για κάθε κλάση περιέχει απλώς μεταβλητές και μεθόδους χωρίς να τις διαχωρίζει σε δημόσιες, ιδιωτικές κλπ. Αυτό έγινε για να περιοριστεί το μοντέλο στις δυνατότητες που προσφέρουν όλες οι object-oriented γλώσσες προγραμματισμού και, κατά συνέπεια, να είναι ανεξάρτητο από οποιαδήποτε συγκεκριμένη γλώσσα ή σύστημα και να μπορεί να υλοποιηθεί κατά διαφορετικούς τρόπους και να μεταφερθεί σε διαφορετικά περιβάλλοντα. Οι τελικές αποφάσεις για το περιβάλλον στο οποίο θα αναπτυχθεί το μοντέλο που προτείνεται μπορούν να ληφθούν κατευθείαν στο στάδιο της υλοποίησης.

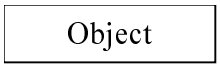
4.1. Περιγραφή της ιεραρχίας κλάσεων

Εφόσον περιγράφουμε ένα object-oriented μοντέλο δεδομένων, είναι προφανές ότι όλες οι οντότητες μέσα στο μοντέλο αυτό θα αντιμετωπίζονται ως αντικείμενα και θα ανήκουν σε κάποιες υποκλάσεις μιας κλάσης Object. Κατά συνέπεια, πρέπει να οριστεί μια γενικευμένη κλάση Object ως ρίζα ολόκληρης της ιεραρχίας κλάσεων.

Είναι επίσης προφανές ότι η βάση δεδομένων hypermedia θα περιέχει αντικείμενα που θα συμπεριφέρονται ως items και αντικείμενα που θα συμπεριφέρονται ως σύνδεσμοι. Όπως ήδη σημειώθηκε, τα αντικείμενα αυτά θα έχουν διαφορετικούς αλλά εξίσου σημαντικούς ρόλους, και θα πρέπει να έχουν διαφορετική αλλά ανάλογη λειτουργικότητα. Σε ένα object-oriented μοντέλο, τα αντικείμενα που έχουν ανάλογη λειτουργικότητα ανήκουν σε κλάσεις στο ίδιο επίπεδο της ιεραρχίας. Έτσι, στο περιβάλλον που έχουμε εδώ, τα items και οι σύνδεσμοι θα ανήκουν σε υποκλάσεις της κλάσης Object οι οποίες θα βρίσκονται στο ίδιο επίπεδο της ιεραρχίας κλάσεων. Καθώς τα items αποθηκεύουν multimedia πληροφορίες και οι σύνδεσμοι αποθηκεύουν συνδέσεις, μπορούν να οριστούν δύο αντίστοιχες υποϊεραρχίες.

Επιπλέον, η βάση δεδομένων πρέπει να κρατάει πληροφορίες για τους χρήστες που έχουν συγγραφικά δικαιώματα (δικαιώματα για δημιουργία και τροποποίηση items ή/και συνδέσμων), ώστε να μπορεί (α) να αποθηκεύσει για κάθε item και σύνδεσμο το ποιος και πότε το δημιούργησε ή το τροποποίησε και (β) να αποθηκεύσει για κάθε συγγραφέα ποια items και ποιους συνδέσμους αυτός δημιούργησε ή τροποποίησε. Έτσι, μπορεί να οριστεί και μια τρίτη υποϊεραρχία από κλάσεις.

Η αρχική πάντως ιεραρχία κλάσεων είναι η



4.1.1. Υποϊεραρχία για τα items

Η κλάση Item ορίζεται ως μια άμεση υποκλάση της κλάσης Object και είναι η ρίζα της ιεραρχίας των κλάσεων για τις multimedia πληροφορίες. Τα items θα περιέχουν δεδομένα multimedia τύπου κειμένου, γραφικών, εικόνας, ήχου, video κλπ. Αποφασίστηκε να περιέχει το κάθε item δεδομένα μόνο μιας συγκεκριμένης μορφής, καθώς τα δεδομένα διαφορετικών μορφών απαιτούν και διαφορετική επεξεργασία. Κατά συνέπεια:

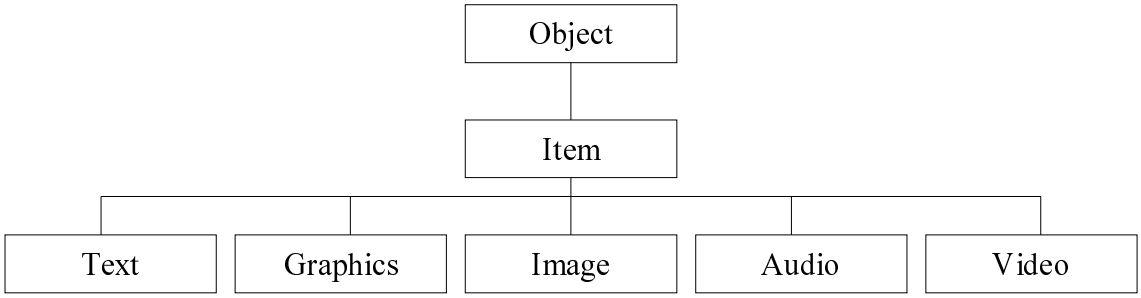
(1) Μπορούν να αναπτυχθούν και να καλούνται ανεξάρτητα μέθοδοι που εφαρμόζονται ειδικά σε items συγκεκριμένης μορφής, και

(2) items τα οποία περιέχουν πληροφορίες διαφορετικής μορφής που σχετίζονται όμως λογικά μπορούν να συνδεθούν αρκετά στενά ώστε να φαίνονται στους χρήστες ως μία οντότητα, χωρίς να πέσει η απόδοση του συστήματος.

Έτσι, ορίζονται κάτω από την κλάση Item οι υποκλάσεις Text, Graphics, Image, Sound και Video, για τα items που αποθηκεύουν τις αντίστοιχες multimedia πληροφορίες. Για κάθε υποκλάση ορίζεται ένας

αριθμός από μεταβλητές που περιγράφουν τα αντικείμενά της, ένας δείκτης σε μια περιοχή του (πιθανότατα οπτικού) μέσου που περιέχει τα αντίστοιχα δεδομένα multimedia, και κάποιες μέθοδοι που χειρίζονται τόσο τα δεδομένα multimedia όσο και τα υπόλοιπα μεταδεδομένα. Εννοείται ότι, πέρα από τις υποκλάσεις που προτείνονται εδώ, και νέες υποκλάσεις μπορούν να οριστούν για την κλάση Item, στην περίπτωση που μια hypermedia βάση δεδομένων υποστηρίζει και άλλες μορφές πληροφορίας.

Η ιεραρχία των κλάσεων γίνεται τώρα



Ο ορισμός των υποκλάσεων της Item δε σημαίνει ότι όλα τα items μπαίνουν σε κάποια από τις υποκλάσεις αυτές και η κλάση Item είναι αφηρημένη, δεν έχει δηλαδή στιγμιότυπα. Μπορούν να οριστούν στιγμιότυπα της κλάσης Item, δηλαδή dummy items που να μην περιέχουν πληροφορίες multimedia και να χρησιμεύουν ως ρίζες σε ιεραρχίες από άλλα items. Ακόμα, αν ένα item δημιουργείται χωρίς να καθορίζεται το τι μορφή πληροφορίας θα περιέχει, το item αυτό θα θεωρείται προσωρινά dummy.

4.1.2. Υποϊεραρχία για τους συνδέσμους

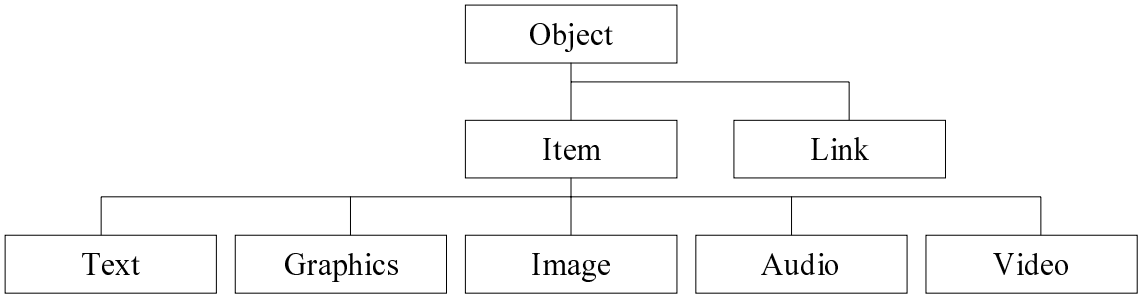
Αποφασίστηκε να υλοποιηθούν οι συνδέσεις μεταξύ των items ως σύνδεσμοι που θα ενώνουν items και θα αποτελούν στιγμιότυπα μιας υποκλάσης Link της κλάσης Object, και όχι ως επιπλέον μεταβλητές των items, όπως θα ταίριαζε στη διαισθητική αντίληψη των χρηστών ότι τα items περιέχουν συνδέσμους. Η απόφαση αυτή ελήφθη για τους εξής λόγους:

(1) Ο ορισμός μιας ξεχωριστής κλάσης Link επιτρέπει τον *απευθείας χειρισμό* των συνδέσμων με εξειδικευμένες μεθόδους, ανεξάρτητα από τα items που αυτοί ενώνουν.

(2) Ο ορισμός δύο ανεξάρτητων κλάσεων για τα items και τους συνδέσμους επιτρέπει να τροποποιείται ή να επεκτείνεται οποιαδήποτε από τις δύο κλάσεις χωρίς να πρέπει να αλλάζει και η άλλη, πράγμα το οποίο αποτελεί ένα ουσιαστικό πλεονέκτημα του object-oriented σχεδιασμού.

Έτσι, η υποϊεραρχία για τους συνδέσμους εκφυλίζεται σε μία μόνο κλάση Link, η οποία είναι παιδί της κλάσης Object.

Η ιεραρχία των κλάσεων γίνεται τώρα

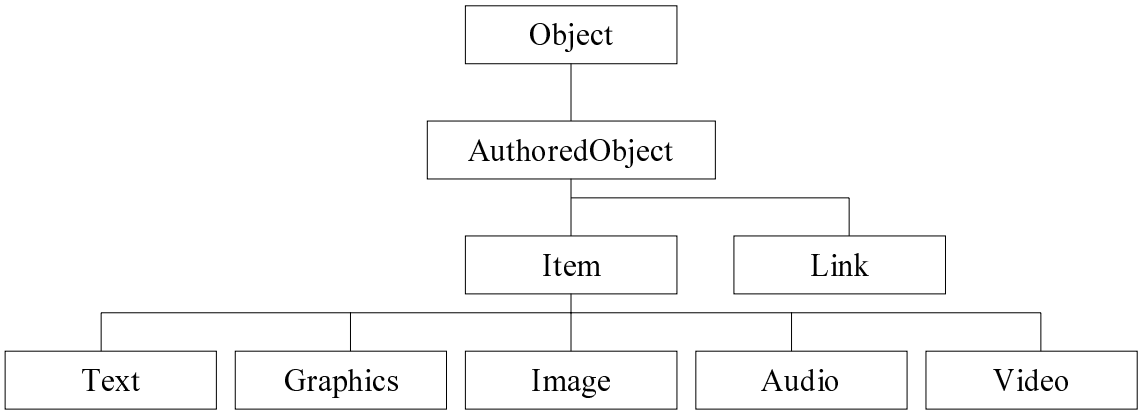


Θα πρέπει να σημειωθεί ότι ο ορισμός των items και των συνδέσμων ως ανεξάρτητων οντοτήτων στο επίπεδο του μοντέλου δεδομένων του συστήματος hypermedia υιοθετείται σε πολλές εφαρμογές, όπως για παράδειγμα στο [MAD89] όπου αναπτύσσεται, χρησιμοποιώντας την ιδέα αυτή, ένα σύστημα hypertext για την υποστήριξη της συνεργασίας σε περιβάλλον γραφείου.

4.1.3. Κοινή υποϊεραρχία για τα αντικείμενα που συγγράφονται

Όπως αναφέρθηκε ήδη, ο συγγραφέας και η στιγμή της δημιουργίας ή της τελευταίας τροποποίησης πρέπει να καταγράφονται για κάθε item και σύνδεσμο, ώστε να διευκολύνεται ο έλεγχος της συνολικής διαδικασίας δημιουργίας της βάσης. Έτσι, τα items και οι σύνδεσμοι έχουν και τα δύο ένα κοινό χαρακτηριστικό, την καταγραφή του συγγραφέα τους και της χρονικής στιγμής της δημιουργίας τους ή της τελευταίας τροποποίησής τους. Υπό αυτή την έννοια, και τα items και οι σύνδεσμοι μπορούν να ενταχθούν σε μια κοινή κατηγορία αντικειμένων, τα "αντικείμενα με συγγραφέα " ("authored objects"), και μπορεί να οριστεί για όλα τα αντικείμενα αυτά μια κοινή υπερϊεραρχία, κάτω βέβαια από την κλάση Object. Έτσι, ορίζεται μια κλάση AuthoredObject που είναι παιδί της κλάσης Object, και οι κλάσεις Item και Link γίνονται τώρα παιδιά της AuthoredObject.

Η ιεραρχία των κλάσεων γίνεται τώρα

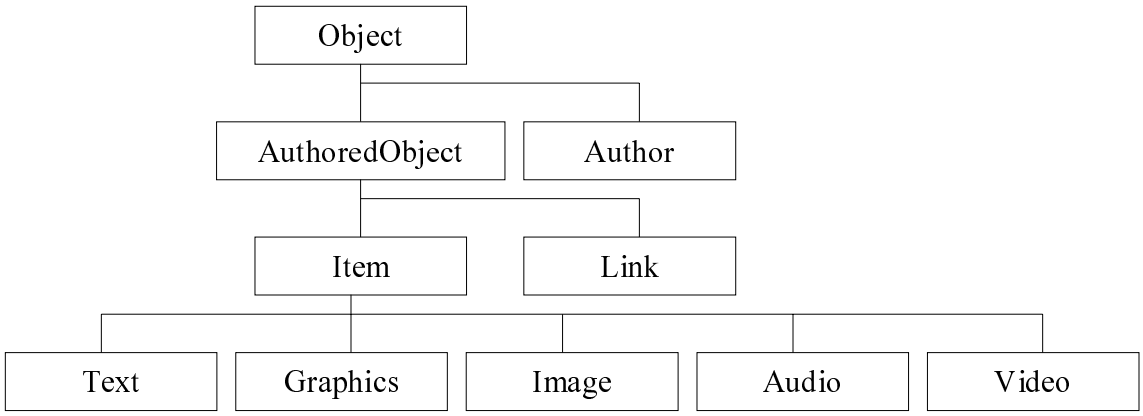


Πρέπει να σημειωθεί ότι η AuthoredObject είναι μια αφηρημένη κλάση, δηλαδή δεν έχει στιγμιότυπα, αλλά ορίζεται αποκλειστικά και μόνο για να ομαδοποιήσει κάποιες κοινές μεταβλητές και μεθόδους των κλάσεων Item και Link, ώστε αυτές να μην οριστούν δύο φορές στις κλάσεις αυτές, αλλά να δηλωθούν μόνο μία φορά την AuthoredObject και έπειτα να κληρονομηθούν.

4.1.4. Υποϊεραρχία συγγραφέων

Η υποϊεραρχία αυτή αποτελείται από μία κλάση μόνο, την κλάση Author, η οποία ορίζεται ως παιδί της γενικής κλάσης Object. Τα στιγμιότυπα της κλάσης Author αντιστοιχούν στους συγγραφείς της hypermedia βάσης δεδομένων και κρατάνε πληροφορίες για αυτούς. Συγγραφέας της βάσης δεδομένων θεωρείται όποιος χρήστης έχει συγγραφικά δικαιώματα (authoring rights), δηλαδή δικαιώματα να δημιουργήσει ή/και να τροποποιήσει items ή/και συνδέσμους. Οι τροποποιήσεις που αφορούν τις σημασιολογικές κυρίως πληροφορίες, αντανακλούν τις προσωπικές απόψεις και προτιμήσεις των χρηστών που τις κάνουν και γι' αυτό το λόγο θα πρέπει κάθε τέτοια τροποποίηση να παρουσιάζεται μαζί με την ταυτότητα του εμπνευστή της, ώστε να μπορούν οι υπόλοιποι χρήστες να καταλάβουν πώς και γιατί έγινε.

Η ιεραρχία των κλάσεων γίνεται τώρα



Αυτή είναι και η βασική ιεραρχία κλάσεων του μοντέλου δεδομένων που προτείνεται. Η ιεραρχία αυτή θα επεκταθεί σε επόμενα κεφάλαια για να καλυφθούν τα πολύ σημαντικά ζητήματα του ορισμού όψεων πάνω στη hypermedia βάση δεδομένων και της τήρησης εκδοχών. Προς το παρόν, θα ορίσουμε αυτή τη βασική ιεραρχία.

4.2. Ορισμός της ιεραρχίας κλάσεων

Στην παράγραφο αυτή ορίζονται οι μεταβλητές και οι μέθοδοι κάθε μιας από τις κλάσεις που έχουν περιγραφεί. Σημειώνεται ότι οι τυπικές παράμετροι των μεθόδων δίνονται εδώ μόνο για να φαίνεται καθαρά το τι κάνει η κάθε μέθοδος, και είναι πολύ πιθανό να αλλάξουν στην υλοποίηση. Επιπλέον, για τις μεθόδους που επιστρέφουν απλές τιμές οι παράμετροι εξόδου παραλείπονται για απλότητα, εφόσον η τιμή που επιστρέφεται κάθε φορά φαίνεται από το όνομα της μεθόδου. Οι τελικές παράμετροι των μεθόδων θα καθοριστούν κατά την υλοποίηση.

4.2.1. Κλάση Object

Η κλάση Object είναι μια αφηρημένη κλάση, δηλαδή δεν έχει στιγμιότυπα, αλλά απλώς χρησιμοποιείται ως ρίζα ολόκληρης της ιεραρχίας των κλάσεων. Στην κλάση Object ορίζεται ο μοναδικός προσδιοριστής των αντικειμένων όλης της βάσης δεδομένων (μια καλή παρουσίαση των αφηρημένων κλάσεων και ειδικότερα του ρόλου που έχει η κλάση Object σε ένα object-oriented μοντέλο δεδομένων, βλ. [PIN88]).

α. Μεταβλητές

ObjectId

Ο προσδιοριστής ενός αντικειμένου είναι μοναδικός για οποιοδήποτε αντικείμενο στη βάση δεδομένων. Οι προσδιοριστές των αντικειμένων που διαγράφονται κρατιούνται για να χρησιμοποιηθούν αργότερα, όταν δημιουργηθούν νέα αντικείμενα.

β. Μέθοδοι

Retrieve(ObjectId)

Η μέθοδος αυτή φέρνει στην κεντρική μνήμη την πληροφορία που περιέχει ένα αντικείμενο με προσδιοριστή ObjectId. Όσον αφορά τα items, η ανάκτηση των multimedia πληροφοριών θα γίνεται κατά διαφορετικό τρόπο ανάλογα με τη μορφή τους, και έτσι η μέθοδος Retrieve θα επανορίζεται στις υποκλάσεις της Item. Επειδή, ωστόσο, οι λεπτομέρειες του επανορισμού εξαρτώνται από την υλοποίηση, η επανοριζόμενη μέθοδος δεν ξαναπαρουσιάζεται στις επόμενες παραγράφους.

Save()

Delete()

Οι μέθοδοι αυτές ορίζονται για την αποθήκευση και διαγραφή αντικειμένων της βάσης δεδομένων. Είναι πιθανό να επανορίζονται στις υποκλάσεις της Object.

Θα πρέπει να σημειωθεί ότι κατά τη διαγραφή ενός item δε διαγράφονται οι σύνδεσμοι που ξεκινάνε από ή καταλήγουν στο item αυτό, όπως και αντίστροφα κατά τη διαγραφή ενός συνδέσμου δεν ενημερώνονται το αρχικό και το τελικό του item. Τα καθήκοντα αυτά εκτελούνται από τις υψηλότερου επιπέδου λειτουργίες διαγραφής item και συνδέσμου αντίστοιχα.

GetObjectId()

Επιστρέφει την τιμή της μεταβλητής ObjectId του αντίστοιχου αντικειμένου.

4.2.2. Κλάση AuthoredObject

α. Μεταβλητές

AuthorId

Το πεδίο αυτό είναι ο προσδιοριστής του συγγραφέα που δημιούργησε το αντικείμενο (item ή σύνδεσμο) ή, αν το αντικείμενο έχει τροποποιηθεί μετά τη δημιουργία του, που το τροποποίησε για τελευταία φορά. Το πεδίο αυτό παίρνει αυτόματα τιμή από το σύστημα κάθε φορά που ένα αντικείμενο δημιουργείται ή τροποποιείται.

TimeStamp

Το πεδίο αυτό περιέχει με κάποια κωδικοποίηση (πιθανότατα αυτήν που χρησιμοποιεί το λειτουργικό σύστημα) τη χρονική στιγμή κατά την οποία έγινε η δημιουργία ή, αν το αντικείμενο έχει τροποποιηθεί μετά τη δημιουργία του, η πιο πρόσφατη τροποποίηση του αντικειμένου. Το πεδίο αυτό παίρνει αυτόματα τιμή από το σύστημα κάθε φορά που ένα αντικείμενο δημιουργείται ή τροποποιείται.

β. Μέθοδοι

Ορίζονται οι μέθοδοι

GetAuthorId()
GetTimeStamp()
SetAuthorId(NewAuthorId)
SetTimeStamp(NewTimeStamp)

για να επιστρέφουν και να θέτουν τις τιμές των αντίστοιχων μεταβλητών ενός αντικειμένου με συγγραφέα που έχει ήδη ανακτηθεί στην κεντρική μνήμη.

Το γεγονός ότι τα items συνδέονται με συνδέσμους θέτει το πρόβλημα αν θα έπρεπε να θεωρούμε ότι τροποποιούνται τα items κάθε φορά που τροποποιούνται οι σύνδεσμοι που φεύγουν από ή καταλήγουν σε αυτά, και αντίστροφα. Προτείνεται να μη γίνονται τέτοιες τροποποιήσεις για απλότητα. Έτσι:

(1) Η λειτουργία υψηλού επιπέδου που τροποποιεί ένα item θα έπρεπε να ενημερώνει τις μεταβλητές Author και TimeStamp του item αυτού, αλλά όχι και τις αντίστοιχες μεταβλητές των συνδέσμων που ξεκινάνε από ή καταλήγουν σε αυτό.

(2) Η λειτουργία υψηλού επιπέδου που δημιουργεί ή τροποποιεί ένα σύνδεσμο θα πρέπει να ενημερώνει τις μεταβλητές Author και TimeStamp του συνδέσμου αυτού, αλλά όχι και τις αντίστοιχες μεταβλητές του αρχικού και του τελικού του item.

(3) Η λειτουργία υψηλού επιπέδου που διαγράφει ένα σύνδεσμο δε χρειάζεται να ενημερώνει τις μεταβλητές Author και TimeStamp του αρχικού και του τελικού του item.

4.2.3. Κλάση Item

α. Μεταβλητές

Καθώς οι κλάσεις Text, Graphics, Image, Audio και Video είναι όλες υποκλάσεις της Item, τα στιγμιότυπά τους θα κληρονομούν αυτόματα όλες τις μεταβλητές και τις μεθόδους που ορίζονται για τα items. Έτσι, κρίθηκε σκόπιμο να οριστούν οι κοινές μεταβλητές όλων αυτών των κλάσεων όχι στις ίδιες τις κλάσεις, αλλά μόνο μία φορά στην υπερκλάση Item.

Οι κοινές μεταβλητές που ορίζονται στην Item είναι οι (α) Description , (β) ContentType, (γ) Properties και (δ) AnchoredLinks. Οι μεταβλητές αυτές αφορούν το κείμενο, τα γραφικά και όλες τις υπόλοιπες πληροφορίες που περιέχονται στα αντικείμενα των κλάσεων Text, Graphics κλπ. Στην ακόλουθη

περιγραφή αναφερόμαστε στο "περιεχόμενο των items" εννοώντας το κείμενο που περιέχουν τα στιγμιότυπα της κλάσης Text, τα γραφικά που περιέχουν τα στιγμιότυπα της κλάσης Graphics, κλπ.

Description

Οι χρήστες πρέπει να έχουν στη διάθεσή τους μια σύντομη περιγραφή του περιεχομένου ενός item. Έτσι, κάθε item έχει μια μεταβλητή Description , η οποία περιέχει ένα σύντομο κείμενο που περιγράφει το περιεχόμενό του. Οι περιγραφές αυτές είναι ιδιαίτερα σημαντικές για τα items που δεν περιέχουν κείμενο: αν δεν υπήρχε ένα δικό του κείμενο που να το περιγράφει, κάθε item με γραφικά, εικόνες, ήχο ή video θα έπρεπε να ενώνεται με ένα ξεχωριστό item που θα περιείχε ένα μικρό κείμενο ως υπότιτλο.

ContentType

Η μεταβλητή αυτή δηλώνει τη μορφή των ακατέργαστων δεδομένων που αντιστοιχούν σε αυτό το item. Η τιμή της μπορεί να είναι TEXT, IMAGE, GRAPHICS, VIDEO, AUDIO ή μια άλλη τιμή, αν έχουν οριστεί από τους χρήστες καινούριες υποκλάσεις της Item. Για dummy items, η μεταβλητή ContentType θα έχει τιμή DUMMY.

Properties

Είναι γενικά αποδεκτό ότι ακόμα και τα συστήματα hypermedia δεν αναιρούν εντελώς την ανάγκη για ανάκτηση πληροφοριών με ερωτήσεις. Γι' αυτό το λόγο, αποφασίστηκε να διατηρηθούν μερικές δυνατότητες για υποβολή ερωτήσεων, πράγμα που απαιτεί, στο επίπεδο του μοντέλου δεδομένων, να υπάρχουν μερικές μεταβλητές που να χαρακτηρίζουν τυπικά τις πληροφορίες που έχουμε. Αυτό μπορεί να γίνει χρησιμοποιώντας για κάθε item μια λίστα ιδιοτήτων.

Οι *ιδιότητες* (properties) είναι διατεταγμένα ζευγάρια χαρακτηριστικών/τιμών. Οι *χαρακτηριστικές* (attributes) δηλώνουν κάποια συγκεκριμένα χαρακτηριστικά του περιεχομένου του item, και οι *τιμές* (values) δηλώνουν την αξιολόγηση του περιεχομένου του item ως προς τις αντίστοιχες χαρακτηριστικές. Στην πράξη, οι χαρακτηριστικές θα είναι πιθανότατα κάποιες γενικές λογικές κατηγορίες και οι τιμές των χαρακτηριστικών αυτών θα είναι αφηρημένα ή κύρια ουσιαστικά (π.χ. "φιλοσοφία" ή "Σωκράτης") ή φράσεις που θα ενέχουν θέση ουσιαστικού (π.χ. "αγγλοσαξωνικός νεοποζιτιβισμός" ή "Μανιφέστο του Υπερρεαλισμού"). Οι χαρακτηριστικές και οι τιμές των χαρακτηριστικών ενός item δεν είναι απαραίτητο να βρίσκονται μέσα στο περιεχόμενό του, και έτσι μπορούν να οριστούν ιδιότητες και για items που δεν περιέχουν κείμενο. Για ένα item της κλάσης Image για παράδειγμα, που απεικονίζει έναν πίνακα του Γκόγια, θα μπορούσε να οριστεί μια χαρακτηριστική "καλλιτέχνης" με τιμή "Γκόγια".

Οι συγγραφείς μπορούν να ορίσουν οσεσδήποτε ιδιότητες πάνω στα items, ώστε η βάση δεδομένων να μπορεί αργότερα να δεχθεί ερωτήσεις ως προς το ποια items έχουν ποιες χαρακτηριστικές, ή ποιες τιμές, ή ποιες τιμές ως προς συγκεκριμένες χαρακτηριστικές.

Είναι φανερό ότι μια ειδική κατηγορία ιδιοτήτων θα είναι οι *λέξεις-κλειδιά* (keywords), δηλαδή οι λέξεις που αντιπροσωπεύουν κεντρικά νοήματα μέσα στο περιεχόμενο ενός item. Αν θέλουμε να δηλώσουμε ότι ένα item περιέχει τις λέξεις-κλειδιά "Μαργκερίτ Γιουρσενάρ", "Μαργκερίτ Ντυράς" και "Σιμόν Ντε Μποβουάρ", θα ορίσουμε τρεις ιδιότητες με χαρακτηριστικές "λέξη-κλειδί" και τιμές "Μαργκερίτ Γιουρσενάρ", "Μαργκερίτ Ντυράς" και "Σιμόν Ντε Μποβουάρ" αντίστοιχα. Έτσι, είναι φανερό ότι οι διδιάστατες ιδιότητες αποτελούν ένα ισχυρότερο μηχανισμό αφαίρεσης από τις μονοδιάστατες λέξεις-κλειδιά, αφού οι τελευταίες μπορούν να υλοποιηθούν ως ιδιότητες που έχουν τη μία διάσταση σταθερή. Περισσότερα για τις λέξεις-κλειδιά θα αναφερθούν σε επόμενη παράγραφο.

AnchoredLinks

Τα items σχετίζονται με συνδέσμους˙ έτσι, μπορεί να κρατιέται σε κάθε item μια λίστα από τους συνδέσμους που ξεκινάνε από ή καταλήγουν στο item αυτό. Οι σύνδεσμοι, ωστόσο, είναι στιγμιότυπα της κλάσης Link και ως τέτοιοι αποθηκεύονται, ανακτώνται και υφίστανται επεξεργασία ξεχωριστά. Έτσι, ένα item δε θα πρέπει να περιέχει τους ίδιους τους συνδέσμους που ξεκινάνε από ή καταλήγουν σε αυτό (αυτό θα φαινόταν τόσο αφύσικο όσο το να λέγαμε ότι ένας σύνδεσμος περιέχει το αρχικό και το τελικό του item) αλλά δείκτες προς τους συνδέσμους αυτούς.

Ένα άλλο πρόβλημα που τίθεται συχνά είναι το ότι οι σημασιολογικές σχέσεις που εκφράζονται φυσικά με τους συνδέσμους είναι σε μερικές περιπτώσεις πολύ ακριβείς, με την έννοια ότι δεν αφορούν ολόκληρο το περιεχόμενο ενός item αλλά μόνο ένα συγκεκριμένο τμήμα του. Έτσι, ένας σύνδεσμος μπορεί να ξεκινά από ή να καταλήγει σε μια καθορισμένη περιοχή μέσα στο περιεχόμενο του αρχικού ή του τελικού του item, όπως μια φράση ή μια λέξη μέσα σε ένα κείμενο, ένα γραφικό μέσα σε μια παράσταση γραφικών, μια συγκεκριμένη περιοχή μέσα σε μια εικόνα, κάποια μέτρα μέσα σε ένα κομμάτι μουσικής, ή κάποιες σκηνές μέσα σε ένα κομμάτι video.

Το τμήμα του περιεχομένου ενός item από όπου ξεκινά ένας σύνδεσμος μπορεί να ονομαστεί *περιοχή αναχώρησης* (departure region) του συνδέσμου αυτού, και το τμήμα ενός item όπου καταλήγει ένας σύνδεσμος μπορεί να ονομαστεί *περιοχή άφιξης* (arrival region) του συνδέσμου αυτού. Η περιοχή αναχώρησης και οι περιοχή άφιξης ενός συνδέσμου λέγονται και *άγκυρες* (anchors) του συνδέσμου αυτού. Οι άγκυρες των συνδέσμων είναι μια πολύ δημοφιλής μεταφορά που συναντάμε σε ολόκληρη τη βιβλιογραφία γύρω από τα συστήματα hypertext και hypermedia (ενδεικτικά παραδείγματα είναι τα [AKS88] και [GARR86]).

Στο προτεινόμενο μοντέλο, η άγκυρα ενός συνδέσμου μπορεί να υλοποιηθεί ως μια λίστα από αριθμούς οι οποίοι, σύμφωνα με κάποια κωδικοποίηση, θα περιγράφουν πλήρως την περιοχή αναχώρησης ή άφιξης του συνδέσμου μέσα στο περιεχόμενο ενός item. Ανάλογα με τον τύπο του περιεχομένου του item, μπορούν να χρησιμοποιηθούν για τις άγκυρες των συνδέσμων οι ακόλουθες κωδικοποιήσεις:

(1) Εφόσον η άγκυρα ενός συνδέσμου μέσα σε κείμενο είναι ένα συνεχόμενο τμήμα κειμένου, μπορεί να παρασταθεί ως ένα ζευγάρι αριθμών από τους οποίους ο πρώτος δηλώνει τη θέση του πρώτου χαρακτήρα της άγκυρας από την αρχή του κειμένου και ο δεύτερος το μέγεθος της άγκυρας σε χαρακτήρες.

(2) Εφόσον η άγκυρα ενός συνδέσμου μέσα σε παράσταση γραφικών είναι ένα σύνολο γραφικών, μπορεί να παρασταθεί ως το σύνολο των κωδικών των στοιχειωδών γραφικών που περιέχει.

(3) Εφόσον η άγκυρα ενός συνδέσμου μέσα σε εικόνα είναι ένα ορθογώνιο παραλληλόγραμμο, μπορεί να παρασταθεί με μια τετράδα αριθμών, από τους οποίους οι δύο πρώτοι είναι οι συντεταγμένες μιας κορυφής και οι δύο επόμενοι οι συντεταγμένες της αντιδιαμετρικής της.

(4) Εφόσον ο ήχος χωρίζεται σε κάποια στοιχειώδη τμήματα και η άγκυρα ενός συνδέσμου μέσα σε ήχο είναι μια ακολουθία από τέτοια τμήματα, μπορεί να παρασταθεί ως ένα ζευγάρι αριθμών από τους οποίους ο πρώτος δηλώνει τη θέση του πρώτου τμήματος της άγκυρας από την αρχή του ήχου και ο δεύτερος το μέγεθος της άγκυρας σε τμήματα ήχου.

(5) Εφόσον η άγκυρα ενός συνδέσμου μέσα σε video είναι μια ακολουθία από σκηνές, μπορεί να παρασταθεί ως ένα ζευγάρι αριθμών από τους οποίους ο πρώτος δηλώνει τη θέση της πρώτης σκηνής της άγκυρας από την αρχή του ήχου και ο δεύτερος το μέγεθος της άγκυρας σε σκηνές.

Η άγκυρα ενός συνδέσμου που ξεκινάει από ή καταλήγει σε ένα ολόκληρο item μπορεί σε κάθε περίπτωση να παρασταθεί ως μια κενή λίστα.

Θα πρέπει να σημειωθεί ότι πολλοί σύνδεσμοι μπορούν να έχουν την ίδια άγκυρα ή επικαλυπτόμενες άγκυρες. Αυτό σημαίνει ότι οποιεσδήποτε συμβάσεις γίνουν για τον τρόπο που οι άγκυρες των συνδέσμων θα αναδεικνύονται μέσα στο περιεχόμενο ενός item (με *εικονίδια* (icons), *αντίστροφο ή έντονο φωτισμό* (reverse video, highlighting), κλπ.), πρέπει να εξασφαλίζουν ότι οι άγκυρες που συμπίπτουν ή επικαλύπτονται θα παριστάνονται στην οθόνη καλόγουστα και χωρίς ασάφειες (μια σύνοψη των σχετικών προβλημάτων μπορεί να βρεθεί στα [GARR86] και [TAN89α]).

Με βάση τα παραπάνω αποφασίστηκε να οριστεί για τα items μια μεταβλητή AnchoredLinks, δηλαδή μια λίστα της οποίας κάθε στοιχείο θα είναι ένα ζευγάρι από τον προσδιοριστή ενός συνδέσμου και μια άγκυρα. Τα στοιχεία της λίστα αυτής θα αντιστοιχούν ένα προς ένα στους σύνδεσμους που ξεκινάνε από το item ή καταλήγουν σε αυτό. Καθώς ο προσδιοριστής κάθε συνδέσμου είναι μοναδικός, η λίστα AnchoredLinks του item θα λειτουργεί ως ένα σύνολο από δείκτες προς συνδέσμους. Με αυτόν τον τρόπο αποφεύγεται η ανάγκη να χρησιμοποιηθούν οι φυσικές διευθύνσεις των συνδέσμων ως δείκτες, επιλογή που θα δημιουργούσε προβλήματα σε ορισμένες περιπτώσεις ενημερώσεων. Η λίστα Anchoredlinks είναι κενή αμέσως μετά τη δημιουργία ενός item και αποκτά ή χάνει μία τιμή μετά από κάθε δημιουργία ή διαγραφή συνδέσμου.

Πρέπει να σημειωθεί ότι οι άγκυρες θα μπορούσαν να είχαν οριστεί εναλλακτικά εξολοκλήρου στην κλάση Link. Η επιλογή που έγινε εδώ αντικατοπτρίζει το γεγονός ότι οι άγκυρες αναφέρονται απευθείας στο περιεχόμενο ενός item και υπό αυτή την έννοια είναι προτιμότερο να οριστούν ως μεταβλητές των items. Με αυτόν τον τρόπο, όταν το περιεχόμενο ενός item αλλάζει και πρέπει φυσιολογικά να αλλάξουν και οι άγκυρες των αντίστοιχων συνδέσμων, αυτές θα υπάρχουν μέσα στο ίδιο το item. Αν αντίθετα αποθηκεύονταν στους συνδέσμους, τότε κάθε φορά που θα άλλαζε το περιεχόμενο ενός item θα έπρεπε να ανακτηθούν οι σύνδεσμοι του item αυτού και να αλλάξει σε κάθε σύνδεσμο η άγκυρά του, πράγμα που θα είχε απαγορευτικό κόστος.

Για να ανακεφαλαιώσουμε, για την κλάση Item ορίζονται οι επόμενες μεταβλητές:

- Description (σύντομο κείμενο που περιγράφει το περιεχόμενο του item)
- ContentType (τι μορφή multimedia πληροφορίας περιέχει το item)
- Properties (λίστα από ζευγάρια χαρακτηριστικής/τιμής που δηλώνουν σημαντικά χαρακτηριστικά του περιεχομένου του item)
- AnchoredLinks (λίστα από τους προσδιοριστές και τις άγκυρες των συνδέσμων που ξεκινάνε από ή καταλήγουν στο item)

β. Μέθοδοι

Πρώτα από όλα τα άλλα, χρειάζονται μέθοδοι για να δημιουργούνται, να ανακτώνται, να αποθηκεύονται και να διαγράφονται items.

Create(AuthorId,TimeStamp,Description,ContentType, Properties)

Η μέθοδος αυτή δημιουργεί ένα item και τοποθετεί στις μεταβλητές του τις τιμές που παίρνει ως παραμέτρους. Δεν αποθηκεύει το item. Η μέθοδος απαιτεί τιμές για όλες τις μεταβλητές του Item εκτός από την AnchoredLinks, η οποία θα είναι έτσι κι αλλιώς μια κενή λίστα, αλλά στην πράξη η μόνη τιμή που πρέπει οπωσδήποτε να δώσει ο χρήστης που δημιουργεί ένα item είναι ο τύπος του περιεχομένου του. Όλες οι άλλες μεταβλητές μπορούν να πάρουν ή όχι τιμές από το χρήστη. Στη δεύτερη περίπτωση, η υψηλότερη λειτουργία δημιουργίας ενός item τους δίνει κάποιες προκαθορισμένες τιμές και μετά καλεί την Create. Οι μεταβλητές AuthorId και TimeStamp παίρνουν τιμή από το σύστημα, το οποίο ξέρει και την ταυτότητα του χρήστη που δημιούργησε το item και τη χρονική στιγμή της δημιουργίας.

GetContentType()

Η μέθοδος αυτή επιστρέφει την τιμή της μεταβλητής ContentType ενός item που έχει ήδη ανακτηθεί. Η μεταβλητή αυτή μπορεί να πάρει τιμή μόνο μία φορά.

Οι μέθοδοι

GetDescription()
GetProperties()
SetDescription(NewDescription)
AddProperty(Property)
DelProperty(Property)

χρησιμοποιούνται για να επιστρέψουν και να θέσουν τις τιμές των μεταβλητών Description και Properties ενός item το οποίο έχει ήδη ανακτηθεί. Εφόσον η Properties είναι μια λίστα δεν ορίζεται μία μέθοδος που να θέτει απλώς την τιμή της, αλλά δύο μέθοδοι για πρόσθεση και διαγραφή στοιχείων.

Αντίστοιχα, οι μέθοδοι

GetAnchoredLinks()
AddAnchoredLink(LinkId,Anchor)
DelAnchoredLink(LinkId)

ορίζονται για το χειρισμό της μεταβλητής AnchoredLinks ενός item που έχει ήδη ανακτηθεί. Η παράμετρος LinkId αντιστοιχεί στον προσδιοριστή ενός συνδέσμου.

Ακόμα, οι μέθοδοι

GetAnchor(LinkId)
SetAnchor(LinkId,NewAnchor)

επιστρέφουν και θέτουν την τιμή της άγκυρας ενός συγκεκριμένου συνδέσμου.

4.2.4. Υποκλάσεις της Item

Ορίζεται μία υποκλάση της Item για κάθε διαφορετική μορφή multimedia πληροφορίας που υποστηρίζει η βάση δεδομένων. Αρχικά ορίζονται οι υποκλάσεις Text, Graphics, Image, Audio και Video, και καινούριες υποκλάσεις μπορούν να προστεθούν αργότερα.

α. Μεταβλητές

Content

Κάθε μια από τις παραπάνω υποκλάσεις περιέχει μια επιπλέον καινούρια μεταβλητή εκτός από τις μεταβλητές τις οποίες κληρονομεί από την Item και την Object, την Content. Η μεταβλητή αυτή αντιπροσωπεύει ακριβώς το πραγματικό περιεχόμενο του αντίστοιχου item, δηλαδή τα multimedia

δεδομένα που αποθηκεύονται σε αυτό. Εφόσον, ωστόσο, τα multimedia δεδομένα αποθηκεύονται σε ένα εξωτερικό, και πιθανότατα οπτικό, μέσο, τιμή της Content δεν είναι τα ίδια τα multimedia δεδομένα αλλά ένας δείκτης προς αυτά, του οποίου η κωδικοποίηση εξαρτάται από το σύστημα αρχείων του μέσου όπου βρίσκονται τα δεδομένα multimedia. Γενικά μπορούμε να υποθέσουμε ότι ο δείκτης προς τα πραγματικά δεδομένα θα είναι το όνομα ενός αρχείου ή/και μια φυσική διεύθυνση του μέσου αποθήκευσης.

Είναι λογικό να αναμένουμε ότι πάνω στο μέσο που αποθηκεύει τα δεδομένα multimedia, τα δεδομένα διαφορετικής μορφής θα έχουν και διαφορετική φυσική οργάνωση, δηλαδή αλλιώς θα είναι φυσικά οργανωμένο το κείμενο, αλλιώς ο ήχος, κλπ. Αυτό σημαίνει ότι και οι δείκτες προς τα δεδομένα αυτά θα έχουν διαφορετική μορφή, και επομένως ότι η μεταβλητή Content έπρεπε αναγκαστικά να οριστεί στις υποκλάσεις της Item. Στην περίπτωση, ωστόσο, που με κάποιο τρόπο εξασφαλιστεί ότι όλοι οι δείκτες θα έχουν την ίδια μορφή, ανεξάρτητα από τη μορφή των δεδομένων στα οποία δείχνουν, η Content μπορεί να οριστεί στην Item, χωρίς αυτό να σημαίνει ότι οι υποκλάσεις της Item δεν έχουν λόγο ύπαρξης. Οι τελευταίες θα πρέπει να παραμείνουν στο μοντέλο για να οριστούν σε αυτές οι μέθοδοι που χειρίζονται τις multimedia πληροφορίες, και οι οποίες θα παραμένουν πάντα εξαρτημένες από τη μορφή των πληροφοριών αυτών.

β. Μέθοδοι

Οι μέθοδοι για την παρουσίαση του περιεχομένου ενός item πρέπει να οριστούν εδώ, καθώς θα διαφέρουν σημαντικά ανάλογα με το είδος της πληροφορίας που παρουσιάζεται. Έτσι, για κάθε υποκλάση ορίζονται οι ακόλουθες μέθοδοι:

GetContent()

Present(WinWidth,WinHeight, ...)

Οι μέθοδοι αυτές φέρνουν το περιεχόμενο ενός item που έχει ήδη ανακτηθεί και παρουσιάζουν το περιεχόμενο αυτό σε μια κατάλληλη συσκευή παρουσίασης (οθόνη κειμένου, οθόνη γραφικών, ηχείο, κλπ.). Η παρουσίαση οπτικής πληροφορίας απαιτεί τουλάχιστον δύο παραμέτρους για να καθοριστεί το μέγεθος του αντίστοιχου παραθύρου˙ η απαίτηση αυτή δεν υπάρχει, προφανώς, για την παρουσίαση ηχητικών πληροφοριών. Ωστόσο, ο ακριβής αριθμός και η σημασία των παραμέτρων που πρέπει να έχει η μέθοδος παρουσίασης εξαρτώνται στενά από το user interface και δεν μπορούν να καθοριστούν πριν από την υλοποίηση, κατά την οποία όλες οι μέθοδοι παρουσίασης θα αναπτυχθούν έτσι ώστε να παρεμβάλλονται μεταξύ του user interface και του λειτουργικού συστήματος και να περνάνε τις κατάλληλες τιμές από το πρώτο στο δεύτερο για να ελέγχεται η παρουσίαση.

Ακόμα, η μέθοδος

SetContent(NewContent)

θα δίνει καινούριο περιεχόμενο σε ένα item. Η τιμή NewContent θα έχει την κωδικοποίηση που αναφέρθηκε πιο πάνω για τους δείκτες προς δεδομένα multimedia. Περιεχόμενο μπορεί να δοθεί με τη SetContent μόνο σε ένα item που δημιουργείται εκείνη τη στιγμή ή σε ένα dummy item. Στη δεύτερη περίπτωση θα πρέπει η υψηλότερου επιπέδου λειτουργία που αποδίδει περιεχόμενο να αλλάζει ανάλογα και τη μεταβλητή ContentType.

Τα παραπάνω, βέβαια, ισχύουν για την περίπτωση που οι multimedia πληροφορίες που αποθηκεύονται στα items δεν είναι ενημερώσιμες. Αν οι πληροφορίες αυτές μπορούν να αλλάξουν, τότε η SetContent πρέπει να επεκταθεί για να καλύψει και την περίπτωση που δεν αποδίδεται περιεχόμενο σε ένα item το οποίο προηγουμένως δεν είχε, αλλά απλώς το περιεχόμενο ενός item αλλάζει. Τότε, θα υπάρχει το πρόσθετο πρόβλημα ότι θα αλλάζουν και οι άγκυρες των συνδέσμων που ξεκινάνε από ή καταλήγουν στο περιεχόμενο αυτό, και συγκεκριμένα ότι οι άγκυρες μπορεί να μετατοπίζονται προς οποιαδήποτε κατεύθυνση, να μεγαλώνουν ή να μικραίνουν, ή να εξαφανίζονται εντελώς. Η υψηλότερου επιπέδου λειτουργία που θα αλλάζει το περιεχόμενο των items, θα πρέπει να καλύψει και τις αλλαγές αυτές. Μια απλή πολιτική είναι να κάνει όλους τους συνδέσμους των οποίων οι άγκυρες αλλάζουν να συνδέονται πλέον με ολόκληρο το item, ενώ το αμέσως λιγότερο πολύπλοκο είναι το να περνάει για όλες τις άγκυρες που αλλάζουν τον έλεγχο στο χρήστη. Η ιδανική συμπεριφορά θα ήταν να υπολογίζει το σύστημα μόνο του την καινούρια θέση του περιεχομένου κάθε άγκυρας και να κάνει αυτόματα τις αναπροσαρμογές.

Τέλος, σημειώνεται ότι το περιεχόμενο των υποκλάσεων Text, Graphics, Image, Audio και Video όπως ορίζεται εδώ δεν έχει καμιά δομή. Αν είναι επιθυμητό και εφικτό να επιβληθεί δόμηση, μπορούν να

οριστούν καινούριες υποκλάσεις των πέντε αυτών κλάσεων με δομημένο περιεχόμενο και να επανοριστούν σε αυτές όσες μεταβλητές και μέθοδοι χρειαστεί.

4.2.5. Κλάση Link

α. Μεταβλητές

Ο ορισμός μεταβλητών για την κλάση Link είναι ένα πολύ σημαντικό σημείο της όλης σχεδίασης γιατί επηρεάζει την ικανότητα της βάσης δεδομένων (α) να υποστηρίξει την περιπλάνηση, (β) να αποθηκεύσει σημασιολογικές πληροφορίες και (γ) να λειτουργήσει αποδοτικά. Έχοντας κατά νου ότι θα πρέπει να ικανοποιούνται οι τρεις αυτές απαιτήσεις ορίζονται οι μεταβλητές που περιγράφονται παρακάτω.

Source, Target

Καθώς οι σύνδεσμοι αποθηκεύονται, ανακτώνται και υφίστανται επεξεργασία ανεξάρτητα από τα items, κάθε σύνδεσμος αναφέρεται στο αρχικό και στο τελικό του item κρατώντας τους προσδιοριστές τους στις μεταβλητές Source και Target αντίστοιχα. Με αυτήν την πληροφορία οι σύνδεσμοι μπορούν να διασχιστούν πολύ εύκολα τόσο προς τα εμπρός όσο και προς τα πίσω, όπου *διάσχιση ενός συνδέσμου προς τα εμπρός* (forward traversal) σημαίνει ανάκτηση του τελικού του item και *διάσχιση ενός συνδέσμου προς τα πίσω* (backward traversal) σημαίνει ανάκτηση του αρχικού του item. Υπενθυμίζεται ότι ο προσδιοριστής ενός συνδέσμου που ενώνει δύο items κρατείται στη μεταβλητή AnchoredLinks τόσο του αρχικού όσο και του τελικού item, με αποτέλεσμα η μεταβλητή AnchoredLinks ενός item να περιέχει τους προσδιοριστές και τις άγκυρες τόσο των συνδέσμων που ξεκινάνε από όσο και των συνδέσμων που καταλήγουν στο item αυτό. Το αν ένας σύνδεσμος φεύγει ή έρχεται μπορεί να εξακριβωθεί συγκρίνοντας τις τιμές που έχει ο σύνδεσμος στις μεταβλητές Source και Target με τον προσδιοριστή του αντίστοιχου item. Θα πρέπει να σημειωθεί ότι η αναγκαιότητα να φυλαχθεί πληροφορία για τη συσχέτιση δύο items μέσω ενός συνδέσμου τόσο στο σύνδεσμο που συσχετίζει όσο και στα items που συσχετίζονται, προκύπτει ακριβώς από το γεγονός ότι οι σύνδεσμοι και τα items συγκροτούν δύο ξεχωριστές κλάσεις αντικειμένων ([MAD89]).

Type, MirrorType

Η μεταβλητή Type δηλώνει τη *σημασιολογία της συσχέτισης* των συνδεόμενων items. Οι πιθανοί τύπου των συνδέσμων ομαδοποιούνται σε τρεις κατηγορίες:

(1) Τύποι που δηλώνουν μια *συμμετρική συσχέτιση* (symmetric relationship) όπως "κοινή θεματολογία" (COMMON_TOPIC), "αντίθετη άποψη" (OPPOSITE_OPINION), "παρεμφερής άποψη" (SIMILAR_OPINION) κλπ. Στην περίπτωση αυτή, η μεταβλητή MirrorType έχει την ίδια τιμή με την Type, δηλώνοντας ότι έχει νόημα η διάσχιση των συνδέσμων και προς τις δύο κατευθύνσεις και ότι η ίδια σχέση δηλώνεται είτε πηγαίνουμε προς τα εμπρός είτε πηγαίνουμε προς τα πίσω. Αυτή η κατηγορία τύπων περιέχει και έναν τύπο "απλή συσχέτιση" (DUMMY_RELATIONSHIP) που δηλώνει μια γενική και αόριστη συσχέτιση, η οποία δεν εμπίπτει αναγκαστικά σε κάποια από τις προηγούμενες κατηγορίες.

(2) Τύποι που δηλώνουν μια *αντισυμμετρική συσχέτιση* (antisymmetric relationship) όπως "συνηγορεί υπέρ" / "υποστηρίζεται από" (SUPPORTS / SUPPORTED_BY) "αναφέρεται σε" / "αναφέρεται από" (REFERS_TO / REFERRED_FROM) "περισσότερες λεπτομέρειες" / "γενικότερο πλαίσιο" (MORE_DETAILS / GENERAL_FRAME). Στις περιπτώσεις αυτές η μεταβλητή MirrorType θα έχει την αντίθετη σημασιολογικά τιμή από την Type, δηλώνοντας ότι έχει νόημα η διάσχιση των συνδέσμων και προς τις δύο κατευθύνσεις αλλά ότι διαφορετική σχέση δηλώνεται αν πηγαίνουμε προς τα εμπρός και διαφορετική αν πηγαίνουμε προς τα πίσω.

(3) Τύποι που δηλώνουν μια μη αντιστρέψιμη συσχέτιση (non-invertible relationship), όπως η "παράδειγμα" (EXAMPLE) και άλλες. Στην περίπτωση αυτή η MirrorType έχει τιμή NULL, δηλώνοντας ότι έχει νόημα η διάσχιση των συνδέσμων μόνο προς τη μία κατεύθυνση. Είναι, πάντως, γεγονός, ότι οι μη αντιστρέψιμες συσχετίσεις είναι πολύ λίγες, καθώς στην πλειοψηφία των περιπτώσεων έχει κάποια νόημα και η αντίθετη σημασιολογικά συσχέτιση.

Οι σύνδεσμοι των οποίων οι τύποι ανήκουν στις δύο πρώτες ομάδες μπορούν να θεωρηθούν αμφικατευθυνόμενοι, με την έννοια ότι είναι απόλυτα σωστό να διασχιστούν προς οποιαδήποτε κατεύθυνση, ενώ οι σύνδεσμοι των οποίων οι τύποι ανήκουν στην τρίτη ομάδα μπορούν να θεωρηθούν μονοκατευθυνόμενοι, με την έννοια ότι δεν είναι σωστό να διασχιστούν προς την αντίστροφη κατεύθυνση. Στο επίπεδο της βάσης δεδομένων, ωστόσο, τόσο οι αμφι- όσο και οι μονοκατευθυνόμενοι σύνδεσμοι

αναφέρονται και στα αρχικά και στα τελικά τους items. Αν πρόκειται να επιβληθεί ο σημασιολογικός περιορισμός να μη διασχίζονται οι μονοκατευθυνόμενοι σύνδεσμοι αντίστροφα, αυτό θα πρέπει να γίνει με ένα εργαλείο υψηλότερου επιπέδου (πιθανότατα στο user interface). Το εργαλείο αυτό θα πρέπει επίσης να εξασφαλίζει ότι κατά τη δημιουργία μονοκατευθυνόμενων συνδέσμων ενός συγκεκριμένου τύπου δεν κατασκευάζονται κύκλοι, πράγμα το οποίο θα ήταν σημασιολογικά λανθασμένο.

Weight

Κάθε σύνδεσμος έχει ένα κανονικοποιημένο βάρος, το οποίο δηλώνει πόσο στενά συσχετίζονται το αρχικό και το τελικό item του συνδέσμου, ως προς τη συγκεκριμένη συσχέτιση. Το βάρος αυτό ορίζεται ως κανονικοποιημένο για να υπάρχει μια κοινά κατανοητή κλίμακα σύγκρισης και οι χρήστες που δημιουργούν συνδέσμους έχουν την υπευθυνότητα να τους αποδίδουν βάρη σύμφωνα με κάποια κοινά αποδεκτά κριτήρια.

Έτσι, για τους συνδέσμους ορίζονται οι εξής καινούριες μεταβλητές:

- Source (προσδιοριστής του αρχικού item)
- Target (προσδιοριστής του τελικού item)
- Type, MirrorType (σημασιολογία της συσχέτισης)
- Weight (σημασιολογικό βάρος της συσχέτισης)

β. Μέθοδοι

Για το χειρισμό των συνδέσμων ορίζονται οι μέθοδοι που αναφέρονται παρακάτω, έχοντας κατά νου ότι οι σύνδεσμοι καθορίζονται μοναδικά από τους οριζόμενους από το σύστημα προσδιοριστές τους.

Create(AuthorId,TimeStamp,Source,Target,Type,MirrorType,Weight)

Η μέθοδος αυτή δημιουργεί ένα σύνδεσμο, του αποδίδει τον πρώτο διαθέσιμο προσδιοριστή και εκχωρεί στις μεταβλητές του τις τιμές που έχει διαβάσει ως παραμέτρους. Δεν αποθηκεύει το σύνδεσμο και δεν ενημερώνει τις μεταβλητές AnchoredLinks του αρχικού και του τελικού του item. Τα τελευταία καθήκοντα ανήκουν στην υψηλότερου επιπέδου λειτουργία της δημιουργίας συνδέσμων.

Όταν κάποιος χρήστης δημιουργεί ένα σύνδεσμο θα πρέπει να παράσχει τιμές τουλάχιστον για το αρχικό και τελικό item και για τον τύπο του συνδέσμου, ενώ για τις μεταβλητές MirrorType και Weight μπορούν να χρησιμοποιηθούν και προκαθορισμένες τιμές (το MirrorType μπορεί να υποτεθεί ότι είναι ίδιο με το Type και το Weight μπορεί να υποτεθεί ότι έχει μια τιμή στο μέσο της κλίμακας). Για τις άγκυρες του συνδέσμου μπορεί επίσης να γίνει η σύμβαση ότι αν δεν τους δίνεται τιμή θα είναι ολόκληρο το περιεχόμενο των αντίστοιχων items, αλλά σε κάθε περίπτωση θα πρέπει η υψηλότερου επιπέδου λειτουργία να καλέσει και τη μέθοδο SetAnchor για το αρχικό και το τελικό item για να εκχωρήσει τιμές. Τέλος, τα AuthorId και TimeStamp παρέχονται από το σύστημα.

Με το να απαιτείται να δοθούν από την αρχή το αρχικό και το τελικό item ενός συνδέσμου εξασφαλίζεται το ότι δεν μπορούν οι χρήστες να φτιάξουν συνδέσμους που να αιωρούνται και ότι αμέσως μετά τη δημιουργία τους οι σύνδεσμοι είναι έτοιμοι για οποιαδήποτε επεξεργασία.

Για να επιστρέφονται οι τιμές των μεταβλητών ενός συνδέσμου που έχει ήδη ανακτηθεί, ορίζονται οι μέθοδοι

GetSource()
GetTarget()
GetType()
GetMirrorType()
GetWeight()

και για να τίθενται νέες τιμές σε αυτές τις μεταβλητές ορίζονται οι μέθοδοι

SetTypes(NewType,NewMirrorType)
SetWeight(NewWeight)

Σημειώνεται ότι αφενός δεν έχει νόημα να αλλάξει το αρχικό ή/και το τελικό item ενός συνδέσμου (αν χρειάζεται μια τέτοια αλλαγή θα πρέπει να διαγραφτεί ο σύνδεσμος και να δημιουργηθεί ένας άλλος), και αφετέρου οι μεταβλητές Type και MirrorType πρέπει να πάρουν τιμή μαζί για να αποφευχθούν οποιεσδήποτε ασυνέπειες.

Ορίζεται επίσης η μέθοδος

Present(WinWidth,WinHeight,...)

για να παρουσιάζονται τα σημασιολογικά δεδομένα των συνδέσμων σε ένα παράθυρο του οποίου τουλάχιστον οι διαστάσεις θα περνούν ως παράμετροι στην μέθοδο παρουσίασης.

4.2.6. 2.6. Κλάση Author

α. Μεταβλητές

Name

Πλήρες όνομα του συγγραφέα. Για απλότητα θεωρείται εδώ ότι δε χρειάζεται να αποθηκεύονται στη βάση οποιαδήποτε άλλα προσωπικά στοιχεία. Στην περίπτωση που χρειάζεται να κρατηθούν και άλλες πληροφορίες, μπορούν να οριστούν καινούριες μεταβλητές, και μέθοδοι που θα διαβάζουν και θα επιστρέφουν τις τιμές τους.

Password

Σύνθημα που χρειάζεται οποιοσδήποτε συγγραφέας (δηλαδή χρήστης με δικαιώματα αλλαγών ή/και τροποποιήσεων items ή/και συνδέσμων) για να μπει στη hypermedia βάση δεδομένων. Η χρήση συνθήματος είναι μια ελάχιστη προστασία απέναντι στη μη εξουσιοδοτημένη προσπέλαση, και είναι αναγκαία δοθέντος ότι ένας συγγραφέας μπορεί να κάνει αλλαγές σε ολόκληρη τη βάση δεδομένων.

β. Μέθοδοι

Όπως και για τις άλλες κλάσεις αντικειμένων, ορίζεται και εδώ η μέθοδος

Create(Name,Password)

με την ανάλογη λειτουργία. Για να επιστρέφονται και να τίθενται οι τιμές των μεταβλητών ορίζονται οι μέθοδοι

GetName()
SetName(NewName)
GetPassword()
SetPassword(NewPassword)

Τέλος, ορίζεται η μέθοδος

Present(WinWidth,WinHeight,...)

για να παρουσιάζονται σε ένα παράθυρο οι πληροφορίες που κρατιούνται για ένα συγγραφέα.

4.3. Χρήση συνδέσμων και ιδιοτήτων σε μια hypermedia βάση δεδομένων

4.3.1. Υποστήριξη της ανάκτησης πληροφοριών

Μια από τις βασικές απαιτήσεις που έχουμε από το μοντέλο δεδομένων μιας hypermedia βάσης δεδομένων είναι το να διευκολύνει την ανάκτηση πληροφοριών και να υποστηρίξει όσο το δυνατόν περισσότερους εναλλακτικούς μηχανισμούς για αυτήν. Όταν κάποιος χρήστης μπαίνει σε μια hypermedia βάση δεδομένων για να βρει κάποιες πληροφορίες, υπάρχουν δύο πιθανότητες:

(1) ενδιαφέρεται για κάποιες συγκεκριμένες πληροφορίες, τις οποίες και μπορεί να περιγράψει, ή

(2) δεν έχει αποφασίσει ή δεν ξέρει ακόμα για ποιες πληροφορίες ενδιαφέρεται, και έτσι δεν μπορεί να τις περιγράψει.

Όταν ένας χρήστης μπορεί να περιγράψει τις πληροφορίες που τον ενδιαφέρουν, δε χρειάζεται να δει όλες τις διαθέσιμες πληροφορίες του συστήματος, αλλά μπορεί να χρησιμοποιήσει ιδιότητες για να εντοπίσει τις πληροφορίες που ζητάει. Επιπλέον, στην περίπτωση αυτή ο χρήστης μπορεί να χρησιμοποιήσει κατευθείαν κατάλληλες όψεις της βάσης δεδομένων, όπως συνίσταται και στη σχετική βιβλιογραφία ([NAN89]) (βλ. στο επόμενο κεφάλαιο για τη χρήση των όψεων).

Όταν, αντίθετα, ένας χρήστης δεν μπορεί να περιγράψει την πληροφορία που τον ενδιαφέρει, θα πρέπει να του παρουσιαστούν όλες οι διαθέσιμες πληροφορίες ώστε να διαλέξει τα κομμάτια που φαίνονται πιο χρήσιμα. Σε αυτό ακριβώς το σημείο, για να μπορέσει να κάνει μια πρώτη εξερεύνηση και επιλογή, είναι απόλυτα απαραίτητο να διασχίσει κάποιους συνδέσμους.

Έτσι, η βάση δεδομένων hypermedia θα πρέπει να παρέχει στους χρήστες αφενός ιδιότητες, ώστε να μπορούν γρήγορα να εντοπίζουν αυτό που θέλουν, και αφετέρου συνδέσμους, ώστε να μπορούν να εξερευνούν με το δικό τους ρυθμό το υλικό που υπάρχει μέσα στη βάση. Αυτή είναι μια καθιερωμένη

απαίτηση, όχι μόνο για τα συστήματα hypertext και hypermedia, αλλά και γενικότερα για οποιαδήποτε συστήματα ανάκτησης πληροφορίας (στο σύστημα RUBRIC ([CUN85]), για παράδειγμα, χρησιμοποιούνται λέξεις-κλειδιά).

Για αυτό το λόγο, το προτεινόμενο μοντέλο δεδομένων περιλαμβάνει τόσο συνδέσμους όσο και ιδιότητες, και έτσι υποστηρίζει τόσο την περιπλάνηση όσο και την υποβολή ερωτήσεων. Όπως αναφέρεται και στο [TAN89α], οι δύο αυτές διαδικασίες θα πρέπει να παρουσιαστούν από το user interface της βάσης δεδομένων ως δύο συμπληρωματικά εργαλεία που χρησιμεύουν σε διαφορετικές περιστάσεις, και όχι ως δύο ανταγωνιστικοί μηχανισμοί που αλληλοαναιρούνται. Θα πρέπει να αναδειχθούν οι περιπτώσεις που είναι πιο σωστό να χρησιμοποιείται περιπλάνηση και οι περιπτώσεις που είναι πιο σωστό να χρησιμοποιούνται ερωτήσεις, έτσι ώστε τελικά να συνδυάζουν οι χρήστες κατάλληλα και τις δύο μεθόδους και να φτάνουν στις ζητούμενες πληροφορίες γρήγορα και απλά.

4.3.2. Υποστήριξη της διασύνδεσης πληροφοριών

Το δεύτερο σημείο στη λειτουργία μιας hypermedia βάσης δεδομένων στο οποίο οι ιδιότητες και οι σύνδεσμοι χρησιμοποιούνται συμπληρωματικά, είναι για τη διασύνδεση των πληροφοριών.

Ένα βασικό έργο των συγγραφέων μιας hypermedia βάσης δεδομένων είναι ο εντοπισμός και η παρουσίαση στους τελικούς χρήστες όλων των συσχετίσεων που υφίστανται μεταξύ του διαθέσιμου multimedia υλικού. Η διαδικασία αυτή μπορεί να ονομαστεί *κατασκευή διασυνδέσεων* (interconnection construction). Επιπλέον, όταν καινούριο υλικό αποθηκεύεται στη βάση, όλες οι συσχετίσεις μεταξύ του υπάρχοντος και του νέου υλικού πρέπει επίσης να εντοπιστούν και να παρουσιαστούν. Η διαδικασία αυτή μπορεί να ονομαστεί *ενημέρωση διασυνδέσεων* (interconnection updating). Ολόκληρη η εργασία της κατασκευής και της ενημέρωσης των διασυνδέσεων μπορεί να ονομαστεί *διαχείριση των διασυνδέσεων* (interconnection administration).

Οι διασυνδέσεις μεταξύ των πληροφοριών μπορούν να αναπαρασταθούν *άμεσα* (explicitly) ή *έμμεσα* (implicitly). Οι σύνδεσμοι αναπαριστούν τις διασυνδέσεις άμεσα, δηλαδή το γεγονός ότι υπάρχει ένας σύνδεσμος από το item A στο item B δηλώνει άμεσα ότι υπάρχει μια συσχέτιση μεταξύ του περιεχομένου των items A και B.

Οι ιδιότητες αναπαριστούν τις διασυνδέσεις έμμεσα, δηλαδή το γεγονός ότι τα items A και B περιέχουν και τα δύο (α) την ίδια χαρακτηριστική με την ίδια τιμή, ή (β) την ίδια χαρακτηριστική με διαφορετικές τιμές, ή (γ) την ίδια τιμή για διαφορετικές χαρακτηριστικές δηλώνει έμμεσα ότι υπάρχει μια συσχέτιση μεταξύ του περιεχομένου των items A και B (λόγω της κοινής χαρακτηριστικής ή/και τιμής).

Οι διασυνδέσεις που αναπαριστάνονται άμεσα με συνδέσμους είναι απευθείας ορατές και επεξεργάσιμες. Ωστόσο, ακριβώς για να εξασφαλιστεί αυτή η ορατότητα οι διασυνδέσεις αυτές πρέπει να έχουν αποκαλυφθεί από κάποιους συγγραφείς, και παρόλο που η κατασκευή των διασυνδέσεων είναι σχετικά εύκολη όταν μια βάση δεδομένων hypermedia χτίζεται προσεκτικά βήμα με βήμα, η ενημέρωση των διασυνδέσεων μπορεί να γίνει πολύ δύσκολη όταν, για παράδειγμα, το 10.001ο item προστεθεί σε μια βάση που περιέχει ήδη άλλα 10.000 items, τα οποία θα πρέπει στη συνέχεια να εξεταστούν για να αποφασιστεί αν θα συνδεθούν με το καινούριο item ή όχι.

Από την άλλη πλευρά, οι διασυνδέσεις που αναπαριστάνονται έμμεσα με κοινές ιδιότητες δεν είναι απευθείας ορατές και επεξεργάσιμες. Μπορούν να ανακαλυφθούν μόνο μετά από επί τούτου ερωτήσεις, που θα αποκαλύψουν όλα τα διασυνδεδεμένα items. Ωστόσο, η κατασκευή και η ενημέρωση των διασυνδέσεων αυτών είναι απλά υπόθεση εισαγωγής, διαγραφής και τροποποίησης ιδιοτήτων. Όταν ένα καινούριο item (με ένα κατάλληλα ορισμένο σύνολο ιδιοτήτων) προστίθεται στη βάση δεδομένων, διασυνδέεται έμμεσα με όλα τα υπάρχοντα items που έχουν ορισμένες κοινές χαρακτηριστικές ή/και τιμές με αυτό. Έτσι, μια ερώτηση ως προς τις συγκεκριμένες κοινές χαρακτηριστικές ή/και τιμές θα επιστρέψει τα υπάρχοντα items μαζί με το καινούριο και με αυτόν τον τρόπο θα αναδείξει τη συσχέτισή τους. Είναι λοιπόν εμφανές το ότι με αυτό το σχήμα η ενημέρωση των διασυνδέσεων είναι αυτόματη και δεν απαιτεί καμιά ειδική ενέργεια είτε από την πλευρά του συστήματος είτε από την πλευρά του συγγραφέα που προσθέτει καινούριο υλικό.

Συμπερασματικά, υπάρχουν δυο μηχανισμοί για την αναπαράσταση των διασυνδέσεων:

(1) Οι σύνδεσμοι αναπαριστούν τις διασυνδέσεις άμεσα, οπότε οι συσχετίσεις φαίνονται αμέσως αλλά απαιτούν εκτεταμένη προσπέλαση στη βάση δεδομένων για να κατασκευαστούν και να ενημερωθούν.

(2) Οι ιδιότητες αναπαριστούν τις διασυνδέσεις έμμεσα, οπότε οι συσχετίσεις φαίνονται μόνο μετά από επί τούτου ερωτήσεις, αλλά κατασκευάζονται και ενημερώνονται αυτόματα.

Στο προτεινόμενο μοντέλο υποστηρίζονται τόσο οι σύνδεσμοι όσο και οι ιδιότητες, με την προοπτική να συνδυαστούν οι δύο αυτοί μηχανισμοί για να αναπαρίστανται οι πιο σημαντικές ή συνηθισμένες συσχετίσεις άμεσα και οι λιγότερο σημαντικές ή συνηθισμένες συσχετίσεις έμμεσα. Επιπλέον, θα ήταν πολύ χρήσιμο να υπάρχουν *διεργασίες-δαίμονες* (daemon processes) που να τρέχουν στο *παρασκήνιο* (background) και να αναζητούν στη βάση δεδομένων ισχυρές έμμεσες συσχετίσεις (δηλαδή ζευγάρια από items των οποίων οι ιδιότητες ταιριάζουν ικανοποιητικά) τις οποίες θα αναπαριστούν άμεσα (δηλαδή δημιουργώντας συνδέσμους ανάμεσα στα αντίστοιχα items). Με αυτό τον τρόπο η βάση δεδομένων hypermedia θα αποκτά και στοιχεία τεχνητής νοημοσύνης, εφόσον θα μπορεί ως ένα βαθμό να συσχετίσει μόνη της το υλικό που θα της παρέχουν οι συγγραφείς της.

4.4. Βιβλιογραφία

1. [AKS88] Robert M. Akscyn, Donald L. McCracken και Elise A. Yoder, "KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations", CACM, τεύχος 31/7, Ιούλιος 1988.
2. [CUN85] Brian P. Mc Cune, Richard M. Tong, Jeffrey S. Dean και Daniel G. Shapiro, "RUBRIC: A System for Rule-Based Information Retrieval", IEEE Transactions on Software Engineering, τεύχος SE-11/9, Σεπτέμβριος 1985.
3. [GARR86] L. Garrett, K. Smith και N. Meyrowitz, "Intermedia: Issues, Strategies and Tactics in the Design of a Hypermedia Document System", πρακτικά CSCW'86, Δεκέμβριος 1986.
4. [MAD89] C.M.Madsen, "Using Persistent Objects to Implement an Environment for Cooperative Work", ACM SIGOIS Bulletin, τεύχος 10/4, Δεκέμβριος 1989.
5. [NAN88] J.Nanard, M.Nanard και H.Richy, "Conceptual Documents: A Mechanism for Specifying Active Views in Hypertext", ACM Conf. on Document Processing Systems, 1988.
6. [PIN88] Lewis J. Pinson και Richard S. Wiener, "An Introduction to Object-Oriented Programming and Smalltalk", Addison-Wesley, 1988.
7. [TAN89α] Gary Tanner, "Navigating Through Hyperchaos", SAFE/HYP/HCI-pap/GT240789, Ιούλιος 1989.

5. ΟΨΕΙΣ ΓΙΑ HYPERMEDIA ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

5.1. Η έννοια των όψεων σε hypermedia βάσεις δεδομένων

Τα συστήματα hypermedia χρησιμοποιούνται και θα χρησιμοποιούνται για ένα ευρύ φάσμα εφαρμογών σε διαφορετικά περιβάλλοντα, ξεκινώντας από εκπαιδευτικές εφαρμογές σε ακαδημαϊκά ιδρύματα και φτάνοντας σε στρατιωτικές εφαρμογές σε περιβάλλοντα υψηλής ασφάλειας. Οι περισσότερες εφαρμογές hypermedia, ωστόσο, χειρίζονται ένα μεγάλο όγκο πληροφοριών οι οποίες θα πρέπει να υποστούν κάποιο φιλτράρισμα πριν να παρουσιαστούν στους τελικούς χρήστες. Το φιλτράρισμα των πληροφοριών είναι απαραίτητο είτε για να επιτρέψει στους χρήστες να συγκεντρωθούν στις πληροφορίες που τους αφορούν περισσότερο, είτε για να προστατεύσει απόρρητα ή πολύτιμα στοιχεία από μη εξουσιοδοτημένη προσπέλαση. Κατά συνέπεια, τα συστήματα hypermedia θα πρέπει να υποστηρίζουν φίλτρα πληροφοριών με τη μορφή των όψεων.

Στη βιβλιογραφία για βάσεις δεδομένων ([DAT84]) αναφέρονται πέντε βασικά πλεονεκτήματα της χρήσης όψεων:

(1) Οι χρήστες δεν αντιλαμβάνονται την επέκταση της βάσης δεδομένων. Όταν νέες δομές δεδομένων (νέες κλάσεις) δημιουργούνται, οι όψεις που ήδη υπάρχουν δε χρειάζεται να αλλάξουν.

(2) Οι χρήστες είναι δυνατόν να μην αντιλαμβάνονται την αναδιοργάνωση της βάσης των δεδομένων. Οι φυσικές δομές των δεδομένων μπορούν ως ένα σημείο να αλλάξουν, χωρίς να πρέπει να τροποποιηθούν οι ορισμοί των όψεων.

(3) Η αντίληψη των χρηστών για τη βάση δεδομένων απλοποιείται σημαντικά.

(4) Τα ίδια δεδομένα μπορούν να παρατηρηθούν από διαφορετικούς χρήστες με διαφορετικό τρόπο.

(5) Παρέχεται αυτόματα ασφάλεια για τα δεδομένα που πρέπει να παραμένουν κρυφά.

Αν και αυτά τα πλεονεκτήματα αναφέρονται έχοντας τις σχεσιακές βάσεις δεδομένων κατά νου, εξακολουθούν να ισχύουν για οποιοδήποτε σύστημα ανάκτησης πληροφοριών. Επιπλέον, οι όψεις είναι απαραίτητες στα συστήματα hypermedia για να παρέχουν στους χρήστες ένα πιο χρήσιμο κομμάτι της βάσης δεδομένων, στο οποίο η περιπλάνηση θα είναι ευκολότερη και πολύ πιο αποδοτική ([NAN88]). Παραπέρα αιτιολόγηση της χρήσης views στα συστήματα hypermedia μπορεί να βρεθεί στο [HAL88].

Δεν υπάρχουν ασάφειες για τη χρήση του όρου "όψη" στο περιβάλλον μιας συμβατικής βάσης δεδομένων. Παρατίθεται ο ακόλουθος ορισμός για τις views στις σχεσιακές βάσεις δεδομένων ([DAT84]):

> Μια όψη μπορεί να εννοείται ως ένας *εικονικός πίνακας* (virtual table), δηλαδή ένας πίνακας ο οποίος δεν υπάρχει από μόνος του αλλά παράγεται από έναν ή περισσότερους υποκείμενους *βασικούς πίνακες* (base tables). Με άλλα λόγια, δεν υπάρχει κανένα αποθηκευμένο αρχείο που να αντιπροσωπεύει άμεσα την ίδια τη view. Αντί γι' αυτό, ο ορισμός της view αποθηκεύεται στο *λεξικό* (dictionary). Ο ορισμός της view δείχνει πώς αυτή παράγεται από τους υποκείμενους βασικούς πίνακες.

Συχνά, ωστόσο, υπάρχουν παρανοήσεις για την ακριβή έννοια του όρου αυτού σ' ένα περιβάλλον hypermedia. Μέσα στο πνεύμα των βάσεων δεδομένων, γράφεται ([KOR86]) ότι μια view είναι

> το υψηλότερο επίπεδο αφαίρεσης (μετά το *φυσικό* (physical) και το *ιδεατό* (conceptual)), στο οποίο περιγράφεται μέρος μόνο της βάσης δεδομένων

και ο ορισμός αυτός δικαιολογείται με το επιχείρημα ότι

> Πολλοί χρήστες της βάσης δεδομένων δε θα ενδιαφέρονται για όλες τις πληροφορίες που αυτή περιέχει. Αντί για το σύνολο των πληροφοριών λοιπόν, οι χρήστες αυτοί θέλουν μόνο ένα μέρος της βάσης των δεδομένων. Για να απλοποιηθεί η αλληλεπίδραση των χρηστών αυτών με το σύστημα, ορίζεται το επίπεδο αφαίρεσης των όψεων. Μπορεί το σύστημα να παρέχει πολυάριθμες όψεις για την ίδια βάση δεδομένων.

Στο περιβάλλον μιας hypermedia βάσης δεδομένων, ωστόσο, η παρουσίαση εστιάζεται στις διαφορετικές απόψεις από τις οποίες μπορούν να αντιμετωπιστούν οι ίδιες πληροφορίες. Κατά συνέπεια, οι όψεις πρέπει να χρησιμεύουν για να εκθέτουν στους χρήστες πολλούς εναλλακτικούς τρόπους με τους οποίους μπορούν να αντιληφθούν τις ίδιες πληροφορίες και τις συσχετίσεις ανάμεσα στις ενότητες

πληροφοριών. Με αναφορά σε *ευφυείς βάσεις δεδομένων* (intelligent databases), σημειώνεται ότι ([PAR89])

> Οι όψεις αντιπροσωπεύουν συγκεκριμένους τρόπους θεώρησης των αποθηκευμένων πληροφοριών και υλοποιούνται ως σύνολα από διαδικασίες συμπερασμού (inference procedures) οι οποίες ελέγχουν την παρουσίαση ή όχι δυναμικών συνδέσμων. Ως τέτοιες λοιπόν, οι όψεις πρέπει να τροποποιούνται δυναμικά και ο τύπος και το πλήθος των χρησιμοποιούμενων όψεων θα ποικίλλει γενικά από τη μια βάση δεδομένων στην άλλη.

Όσον αφορά τη χρήση των όψεων σε μια hypermedia βάση δεδομένων, υπάρχουν δύο πιθανές προσεγγίσεις οι οποίες μπορούν να επεκταθούν και παραπέρα:

(1) Οι χρήστες ομαδοποιούνται και οι views ορίζονται στο επίπεδο των *ομάδων χρηστών* (user groups), και όχι στο επίπεδο των συγκεκριμένων χρηστών. Στην περίπτωση αυτή θα μπορούν να υπάρχουν και ομάδες με ένα μόνο χρήστη. Κάθε χρήστης έχει πρόσβαση μόνο στην όψη που έχει οριστεί για την ομάδα του και θα δουλεύει μέσα σ' αυτή την όψη όσο είναι συνδεδεμένος με το σύστημα.

(2) Οι όψεις ορίζονται χωρίς να σχετίζονται με συγκεκριμένους χρήστες ή ομάδες χρηστών. Κάθε όψη έχει ένα *σύνθημα* (password) και μπορεί να χρησιμοποιηθεί μόνο από τους χρήστες που ξέρουν το σύνθημα αυτό. Κάθε χρήστης μπορεί να προσπελάσει όλες τις όψεις των οποίων ξέρει τα συνθήματα, ακόμα και κατά την ίδια σύνοδο με το σύστημα.

Η δεύτερη προσέγγιση είναι περισσότερο ευέλικτη από την πρώτη, καθώς επιτρέπει τον ορισμό μικρών όψεων ειδικού σκοπού και παρέχει στους χρήστες περισσότερα από ένα περιβάλλοντα εργασίας, κατασκευασμένα κατά τρόπο ώστε να εξασφαλίζεται και η ελεγχόμενη πρόσβαση και η χρησιμότητα των πληροφοριών που ανακτώνται. Έτσι, μπορούμε να αφήσουμε τους χρήστες ελεύθερους να δουλεύουν με όλες τις όψεις των οποίων ξέρουν τα συνθήματα. Αν, από την άλλη πλευρά, υπάρχουν περιπτώσεις όπου δεν πρέπει να χρησιμοποιεί ένας χρήστης οποιαδήποτε όψη της οποίας τυχαίνει να ξέρει το σύνθημα, μπορούμε να περιορίσουμε λίγο το σχήμα αυτό δηλώνοντας από την αρχή ποιες όψεις μπορεί γενικά να χρησιμοποιήσει ο κάθε χρήστης, οπότε από εκεί και πέρα κάθε χρήστης θα χρησιμοποιεί μόνο τις όψεις που έχουν δηλωθεί για αυτόν και των οποίων ξέρει το σύνθημα.

5.2. Απαιτήσεις για όψεις σε hypermedia βάσεις δεδομένων

Ένα σύστημα hypermedia περιέχει ένα μεγάλο πλήθος από διασυνδεδεμένες πληροφορίες. Οι όψεις πρέπει να επιβληθούν τόσο πάνω στις πληροφορίες αυτές όσο και στις συνδέσεις μεταξύ των πληροφοριών. Στη μοντέλο δεδομένων που σχεδιάζεται, το σύστημα hypermedia είναι ένα δίκτυο από items και συνδέσμους ανάμεσα σε items. Κατά συνέπεια, μια όψη θα είναι ένα υποσύνολο τον στοιχείων του δικτύου (items και συνδέσμων).

Εισάγουμε τα ακόλουθα τεχνικά κριτήρια για την εκτίμηση της λειτουργικότητας των views:

(1) *Συνέπεια των όψεων* (view consistency).

Οι όψεις πρέπει να ορίζονται με τέτοιο τρόπο ώστε όλοι οι σύνδεσμοι που περιέχονται στον ορισμό μιας όψης να ενώνουν items που επίσης περιέχονται στον ορισμό της όψης. Αν αυτή η απαίτηση αγνοηθεί, δηλαδή αν έχουμε συνδέσμους που δείχνουν σε items ή πηγάζουν από items έξω από την όψη, τότε οι χρήστες θα έπαιρναν άσχετες με τα ενδιαφέροντά τους πληροφορίες ακολουθώντας συνδέσμους μιας υποθετικά χρήσιμης όψης, οι οποίες απλώς θα τους αποσπούσαν την προσοχή. Μια σοβαρότερη συνέπεια είναι ότι ένας μη εξουσιοδοτημένος χρήστης θα μπορούσε να χρησιμοποιήσει συνδέσμους μιας όψης για να προσπελάσει πληροφορίες προστατευμένες έξω από τα όρια της όψης αυτής. Έτσι, δίνουμε τον επόμενο ορισμό:

> Μια όψη είναι *συνεπής* αν και μόνο αν όλοι οι σύνδεσμοι που περιλαμβάνονται στον ορισμό της συνδέουν items που επίσης περιλαμβάνονται στον ορισμό της.

(2) *Ακρίβεια των όψεων* (view preciseness).

Οι όψεις πρέπει να ορίζονται με ακρίβεια, δηλαδή κατά τέτοιο τρόπο ώστε να περικλείουν όλες τις σχετικές πληροφορίες και να αποκλείουν όλα τα άσχετα δεδομένα. Αν οι χρήστες δουλεύουν με ανακριβείς όψεις τότε ή θα πρέπει να ανεχθούν μια ορισμένη ποσότητα από άσχετα δεδομένα προσπαθώντας να φτάσουν στις πληροφορίες που τους ενδιαφέρουν μέσα από πολύ "πλατιές" όψεις, ή θα χάσουν μερικές, σημαντικές ίσως, πληροφορίες, δουλεύοντας με πολύ "στενές" όψεις για να αποφύγουν τα περιττά δεδομένα. Έτσι, δίνουμε τον επόμενο ορισμό:

Μια όψη είναι *ακριβής* αν και μόνο αν περικλείει όλες τις σχετικές με το αντικείμενό της πληροφορίες και αποκλείει όλες τις άσχετες με το αντικείμενό της πληροφορίες.

(3) *Πρακτικότητα των όψεων* (view practicality).

Οι αλγόριθμοι για τον ορισμό και την τροποποίηση των views θα πρέπει να μην είναι πολύπλοκοι και να μη χρειάζονται ως είσοδο μεγάλο πλήθος μεταπληροφοριών. Έτσι, δίνουμε τον ακόλουθο ορισμό:

Μια όψη είναι *πρακτική* αν και μόνο αν ορίζεται και τροποποιείται από απλούς αλγόριθμους που χρησιμοποιούν μικρές ποσότητες μεταδεδομένων.

(4) *Συντηρησιμότητα των όψεων* (view maintainability).

Όταν ένας χρήστης δοκιμάζει να ανακτήσει πληροφορία, να διασχίσει συνδέσμους ή να δημιουργήσει καινούριους συνδέσμους, πρέπει να εξασφαλίζεται το ότι δεν παραβιάζεται ο ορισμός της *ενεργής όψης* (active view), δηλαδή της όψης η οποία χρησιμοποιείται εκείνη τη στιγμή. Οι αλγόριθμοι για αυτήν τη διαδικασία, που μπορεί να ονομαστεί *συντήρηση των όψεων* (view maintenance), πρέπει να εκτελούνται σε χρόνο πολύ μικρότερο από τον πραγματικό χρόνο εκτέλεσης της λειτουργίας που σκοπεύουν να ελέγξουν αν επιτρέπεται ή όχι. Έτσι, δίνουμε τον επόμενο ορισμό:

Μια όψη είναι *συντηρήσιμη* αν και μόνο αν συντηρείται από έναν απλό αλγόριθμο ο οποίος εκτελείται σε ένα μικρό ποσοστό του χρόνου εκτέλεσης των λειτουργιών που ελέγχει αν επιτρέπονται ή όχι.

Έτσι, οι όψεις πάνω στα items και στους συνδέσμους του δικτύου hypermedia πρέπει να είναι συνεπείς, ακριβείς, πρακτικές και συντηρήσιμες.

5.3. Μοντέλο για όψεις σε μια hypermedia βάση δεδομένων

Το προτεινόμενο μοντέλο θα χρησιμοποιεί τον ακόλουθο μηχανισμό για τον ορισμό των όψεων:

Μια όψη θα αποτελείται από τα items και τους συνδέσμους που ικανοποιούν έναν αριθμό από *κριτήρια επιλογής* (selection criteria) αναφορικά με συνδυασμούς ιδιοτήτων, τύπους και βάρη των links, και ακόμα αναφορικά με τη θέση των items μέσα στο σημασιολογικό δίκτυο. Επιπλέον συγκεκριμένα items μπορούν να περικλείονται σε μια όψη ή να αποκλείονται από αυτήν, όταν αυτό κρίνεται σκόπιμο από το σχεδιαστή της. Οι όψεις που ορίζονται με αυτόν τον τρόπο μπορούν να συντηρηθούν με τις ακόλουθες δοκιμές:

(1) Δοκιμή για ανάκτηση item: Ένα item επιτρέπεται να ανακτηθεί αν και μόνο αν (α) περικλείεται ονομαστικά στον ορισμό της όψης ή (β) ικανοποιεί ένα κριτήριο επιλογής και δεν αποκλείεται ονομαστικά από τον ορισμό της όψης.

(2) Δοκιμή για διάσχιση συνδέσμου: Ένας σύνδεσμος μπορεί να διασχιστεί αν και μόνο αν (α) ικανοποιεί ένα κριτήριο επιλογής ή (β) το item από το οποίο ξεκινά η διάσχιση του συνδέσμου περικλείεται στον ορισμό της όψης.

(3) Δοκιμή για δημιουργία συνδέσμου: 'Ενας νέος σύνδεσμος μπορεί να δημιουργηθεί αν και μόνο αν το item από το οποίο ο σύνδεσμος ξεκινά και το item όπου ο σύνδεσμος καταλήγει μπορούν να ανακτηθούν μέσα από την όψη.

Με αυτή την πολιτική συντήρησης των όψεων, υπάρχει η περίπτωση να διασχίσει ένας χρήστης ένα σύνδεσμο και να ανακαλύψει ότι το item όπου έφτασε δεν μπορεί να ανακτηθεί μέσω της ενεργής όψης. Στην περίπτωση αυτή, το item ανακτάται αλλά δεν παρουσιάζεται στο χρήστη. Μόνο μερικά από τα μεταδεδομένα του item, όπως το ObjectId και το Description θα μπορούσαν να παρουσιαστούν αν το ζητήσει ο χρήστης. Οι σύνδεσμοι που πηγάζουν από αυτό το item ή που δείχνουν σε αυτό δεν μπορούν να χρησιμοποιηθούν.

Αυτό το σχήμα κατασκευής και συντήρησης όψεων παράγει λειτουργικές όψεις, σύμφωνα με τα κριτήρια που τέθηκαν στην προηγούμενη ενότητα:

(1) Η συνέπεια των όψεων εξασφαλίζεται σε όλες τις περιπτώσεις εκτός από αυτήν που προαναφέρθηκε. Ακόμα και σε αυτήν την περίπτωση όμως, παρόλο που διασχίζεται ένας σύνδεσμος που δείχνει έξω από την όψη, το τελικό item δεν παρουσιάζεται στο χρήστη πλήρως και έτσι η όψη δεν παραβιάζεται.

(2) Η ακρίβεια των όψεων είναι θέμα σχεδιασμού: οι σχεδιαστές (συγγραφείς) μπορούν να περικλείσουν στις όψεις που δημιουργούν ή να αποκλείσουν από αυτές συγκεκριμένα items και με αυτόν τον τρόπο να απομονώσουν ακριβώς τις ενδιαφέρουσες πληροφορίες.

(3) Η πρακτικότητα και η συντηρησιμότητα των όψεων είναι θέμα υλοποίησης και δεν υπάρχει καμιά ενδογενής πολυπλοκότητα στο προτεινόμενο σχήμα. Τα κριτήρια επιλογής μπορούν να εκφραστούν

όμοια με κανόνες τύπου PROLOG, ενώ οι προσδιοριστές των items που περικλείονται στις όψεις ή αποκλείονται από αυτές ονομαστικά μπορούν να διατηρούνται σε λίστες. Έτσι, οι όψεις μπορούν να κατασκευάζονται εύκολα. Οι κανόνες και οι λίστες μπορούν να ενημερώνονται γρήγορα, και έτσι οι όψεις μπορούν να τροποποιούνται εύκολα. Ο σωστός σχεδιασμός μιας συγκεκριμένης όψης μπορεί να εξασφαλίσει τη γρήγορη αποτίμηση των κανόνων που χρησιμοποιούνται, ενώ η επεξεργασία λιστών δε δημιουργεί συνήθως σημαντική επιβάρυνση. Έτσι, οι όψεις μπορούν να συντηρούνται εύκολα.

Με αυτήν την πολιτική διαχείρισης των όψεων, το home item μιας όψης, δηλαδή το item στο οποίο τοποθετείται ένας χρήστης μόλις μπαίνει στην όψη αυτήν, μπορεί να υπολογιστεί ως το πρώτο item που περικλείεται ονομαστικά στη όψη ή, αν δεν υπάρχει τέτοιο item, ως το πρώτο item στη βάση δεδομένων που ικανοποιεί ένα κριτήριο επιλογής. Αν και οι δύο υπολογισμοί αποτύχουν, τότε η αντίστοιχη όψη δεν περιέχει items και δεν μπορεί να χρησιμοποιηθεί. Αυτός είναι ο λόγος για τον οποίο επιτρέπεται μεν, αλλά δε συνίσταται, η δημιουργία μιας άδειας όψης.

5.4. Επέκταση της ιεραρχίας κλάσεων για την υποστήριξη όψεων

Οι όψεις αποτελούνται, όπως είπαμε, από items και συνδέσμους. Ακριβώς όπως σε ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων οι ορισμοί των όψεων αποθηκεύονται ως ερωτήσεις, έτσι και οι ορισμοί των όψεων πάνω σε μια object-oriented βάση δεδομένων μπορούν να αποθηκευτούν ως αντικείμενα. Κατά συνέπεια, η ιεραρχία κλάσεων που έχουμε ορίσει θα πρέπει να επεκταθεί κατά μία ακόμα κλάση View, της οποίας τα στιγμιότυπα θα είναι ορισμοί όψεων. Εδώ για απλότητα θα δεχθούμε ότι δε μας ενδιαφέρουν οι συγγραφείς και οι χρόνοι τροποποίησης των όψεων, και έτσι η κλάση View δε θα μπει στην ιεραρχία των κλάσεων ως παιδί της AuthoredObject αλλά ως απευθείας απόγονος της αφηρημένης κλάσης Object.

Στο υπόλοιπο αυτού του κεφαλαίου θα χρησιμοποιηθεί ο όρος "όψη" για να δηλωθεί ένα στιγμιότυπο της κλάσης View, δηλαδή με την έννοια του "ορισμού όψης".

Για να οριστεί η κλάση View ορίζουμε πρώτα τις μεταβλητές και στη συνέχεια τις μεθόδους της.

α. Μεταβλητές

Name

Μοναδικό όνομα που δίνει σε μια όψη ο δημιουργός της. Το Name ορίζεται από τους χρήστες, ενώ ταυτόχρονα υπάρχει και ο προσδιοριστής της όψης, που είναι μοναδικός και ορίζεται από το σύστημα. Έτσι, με ορολογία βάσεων δεδομένων, το Name είναι ένα *υποψήφιο κλειδί* (candidate key) για την όψη και χρησιμοποιείται για να μπορούν *οι χρήστες* να αναφέρονται σε όψεις χρησιμοποιώντας μοναδικά ονόματα τα οποία όμως μπορούν να έχουν κάποιο νόημα εφόσον δεν τα ορίζει το σύστημα αλλά οι ίδιοι. Σημειώνεται ότι όψεις μπορούν να δημιουργήσουν είτε οι χρήστες που έχουν προνόμια συγγραφέα είτε ο διαχειριστής του συστήματος.

Password

Σύνθημα το οποίο απαιτείται να γνωρίζει όποιος θέλει να χρησιμοποιήσει την όψη. Ο καθορισμός ενός συνθήματος κατά τη δημιουργία μιας όψης είναι υποχρεωτικός για λόγους ασφάλειας, διότι οποιοσδήποτε συγγραφέας μπει σε μια όψη έχει σε αυτήν δικαιώματα ενημέρωσης και μπορεί να επιφέρει ανεπιθύμητες αλλαγές.

Description

Το Description μιας όψης είναι ένα πολύ σύντομο κείμενο που περιγράφει το περιεχόμενό της.

IncItems , ExcItems

Οι μεταβλητές IncItems και ExcItems είναι λίστες από τους μοναδικούς προσδιοριστές των items που εγκλείονται άμεσα στην ή αποκλείονται άμεσα από την όψη. Έχουν τη δομή

[ItemId_1, ... ItemId_N]

όπου το κόμμα ',' δηλώνει σύζευξη.

CriteriaList

Αυτή η μεταβλητή είναι μια λίστα από περισσότερο γενικά κριτήρια για τον εγκλεισμό ενός item ή ενός συνδέσμου σε μια όψη. Τα κριτήρια αυτά ενώνονται διαζευκτικά. Κάθε κριτήριο είναι μια σύζευξη συνθηκών που αφορούν items ή συνδέσμους. Οι συνθήκες που αφορούν items έχουν να κάνουν με χαρακτηριστικά των ίδιων των items ή των συγγραφέων που τα δημιούργησαν, των συνδέσμων που

συνδέονται με αυτά, ή των όψεων που τα περιέχουν. Οι συνθήκες που αφορούν συνδέσμους έχουν να κάνουν με τα χαρακτηριστικά των ίδιων των συνδέσμων, ή των items που ενώνονται, ή των συγγραφέων που δημιούργησαν τους συνδέσμους, ή των όψεων που τους περιέχουν (περισσότερες πληροφορίες μπορούν να βρεθούν στο παράρτημα Β όπου ορίζεται σε BNF μια γλώσσα χειρισμού δεδομένων της οποίας μέρος είναι και τα κριτήρια αυτά).

Θα πρέπει να σημειωθεί ότι, όπως ήδη ειπώθηκε, αν το τελικό ή/και το αρχικό item ενός συνδέσμου ανήκει σε μια όψη, τότε ο σύνδεσμος μπορεί να διασχιστεί προς τα εμπρός ή/και προς τα πίσω μέσα από την όψη αυτή. Το γεγονός αυτό δεν καθιστά τα κριτήρια επιλογής συνδέσμων περιττά, διότι τα τελευταία έχουν προτεραιότητα απέναντι στον έμμεσο μηχανισμό επιλογής συνδέσμων με βάση τα items. Έτσι, πρώτα ελέγχουμε αν ένας σύνδεσμος ανήκει σε μια όψη με βάση τα κριτήρια που έχει η όψη αυτή, και μετά ελέγχουμε αν ο σύνδεσμος αυτός μπορεί να διασχιστεί προς τα εμπρός ή/και προς τα πίσω.

Αν υποθέσουμε ότι μια όψη περιέχει N κριτήρια C_1, C_2, .. , C_N, από τα οποία το καθένα παράγει ένα σύνολο S_i από items και συνδέσμους, και ακόμα ότι IncI και ExcI είναι τα σύνολα των άμεσα εγκλειόμενων και αποκλειόμενων items, τότε τα itesm και οι σύνδεσμοι που τελικά περιέχονται στην όψη θα είναι το σύνολο

$$V = \left[\bigcup_{n=1..N} (S_n) \bigcup IncI \right] - ExcI$$

Για να ανακεφαλαιώσουμε, μια όψη περιέχει τις εξής μεταβλητές:

Name	(μοναδικό όνομα της όψης)
Password	(σύνθημα που απαιτείται για να χρησιμοποιηθεί η όψη)
Description	(σύντομο κείμενο που περιγράφει το περιεχόμενο της όψης)
IncItems	(προσδιοριστές των άμεσα εγκλειόμενων items)
ExcItems	(προσδιοριστές των άμεσα αποκλειόμενων items)
CriteriaList	(διάζευξη από κριτήρια επιλογής για items και συνδέσμους)

Το γεγονός ότι οι όψεις μπορούν να περιέχουν μοναδικούς προσδιοριστές των items στις μεταβλητές IncItems και ExcItems υποδηλώνει ότι υπάρχει μια εξάρτηση ανάμεσα στις όψεις και στα items. Όταν διαγράφεται ένα item, η υψηλότερου επιπέδου λειτουργία της διαγραφής θα πρέπει επίσης να προσπελάσει όλες τις όψεις και να σβήσει τον προσδιοριστή του item αυτού από όλες τις λίστες στις οποίες αναφέρεται. Με αυτόν τον τρόπο ο προσδιοριστής του διαγραμμένου item απελευθερώνεται αμέσως και είναι έτοιμος να ξαναχρησιμοποιηθεί. Αυτή η διαδικασία, βέβαια, αυξάνει κατά πολύ την πολυπλοκότητα της διαγραφής items αλλά φαίνεται ότι είναι ένα απαραίτητο τίμημα προκειμένου να διατηρούμε την *ακεραιότητα αναφορών* (referential integrity) της βάσης δεδομένων και να έχουμε ακριβείς όψεις.

Εκτός βέβαια από αυτόν τον αλγόριθμο άμεσης διαγραφής, υπάρχει και η δυνατότητα να απομακρύνουμε τους προσδιοριστές των διαγραμμένων items κατά τρόπο έμμεσο. Η έμμεση απομάκρυνση στηρίζεται στην παρατήρηση ότι αν ένα item διαγραφεί, τότε το σύστημα δεν πρόκειται ποτέ να το συναντήσει και κατά συνέπεια ποτέ δε θα το αναζητήσει μέσα στα items που εγκλείονται σε ή αποκλείονται από μια συγκεκριμένη όψη. Επομένως, τα διαγραμμένα items μπορούν να παραμένουν στις λίστες των εγκλειόμενων και αποκλειόμενων items των όψεων με τη σύμβαση ότι κάθε φορά που πρόκειται το σύστημα να χρησιμοποιήσει μια όψη, πριν από οτιδήποτε άλλο θα εξετάζει ποια από τα items που εγκλείονται ή αποκλείονται άμεσα έχουν διαγραφτεί και ποια όχι και θα απομακρύνει από τις λίστες όσα βρίσκει ότι έχουν διαγραφτεί. Εδώ βέβαια υπάρχει το πρόβλημα ότι δεν μπορούμε να ξέρουμε πότε τελικά ένα διαγραμμένο item δεν αναφέρεται πλέον στον ορισμό καμιάς όψης ώστε να μπορεί να απελευθερωθεί ο προσδιοριστής του. Γι' αυτό το λόγο, η έμμεση διαδικασία απομάκρυνσης θα πρέπει να χρησιμοποιείται σε συνδυασμό με ένα πρόγραμμα ελέγχου το οποίο θα τρέχει περιοδικά και θα βρίσκει ποιοι προσδιοριστές απελευθερώνονται και ποιοι όχι.

β. Μέθοδοι

Πρώτα από όλα πρέπει να οριστούν μέθοδοι για τους *βασικούς* χειρισμούς των όψεων, δηλαδή για τη δημιουργία, ανάκτηση, αποθήκευση και διαγραφή όψεων. Ορίζονται οι επόμενες μέθοδοι:

Create(Name, Password, Description, IncItems, ExcItems, CriteriaList)

Η μέθοδος αυτή δημιουργεί μια όψη στη μνήμη, της αποδίδει τον πρώτο διαθέσιμο ObjectId και

αποθηκεύει τις τιμές των παραμέτρων στις αντίστοιχες μεταβλητές. Όπως και οι ανάλογες μέθοδοι των άλλων κλάσεων, δεν αποθηκεύει την όψη.

Απαιτούνται τιμές μόνο για τις μεταβλητές Name και Password, και ο δημιουργός της όψης πρέπει να παροτρύνεται να δώσει τουλάχιστον μια μη κενή λίστα εγκλειόμενων items ή ένα κριτήριο που να επιλέγει κάποια items, προκειμένου να μη δημιουργηθεί μια κενή όψη. Η δημιουργία κενών όψεων επιτρέπεται αλλά δε συνίσταται, αφού μια κενή όψη δεν έχει home item και δεν μπορεί να χρησιμοποιηθεί. Όλες οι άλλες μεταβλητές μπορούν να πάρουν προκαθορισμένες τιμές, που θα δοθούν σε επίπεδο user interface.

Η ανάκτηση μιας όψης γίνεται με τη μέθοδο ανάκτησης που κληρονομείται από την κλάση Object.

Οι μέθοδοι

GetName()
GetDescription()
GetPassword()

επιστρέφουν τις τιμές των αντίστοιχων μεταβλητών για μια όψη που έχει ήδη ανακτηθεί. Ακόμα, οι μέθοδοι

SetName(NewName)
SetDescription(NewDescription)
SetPassword(NewPassword)

θέτουν νέες τιμές στις αντίστοιχες μεταβλητές μιας όψης που έχει ήδη ανακτηθεί.

Οι μέθοδοι

GetCriteriaList()
GetIncItems()
GetExcItems()

και

AddCriterion(NewCriterion)
AddIncItem(ItemId)
AddExcItem(ItemId)
DelCriterion(Criterion)
DelIncItem(ItemId)
DelExcItem(ItemId)

χρησιμοποιούνται για να επιστραφούν οι τιμές των αντίστοιχων μεταβλητών μιας όψης που έχει ήδη ανακτηθεί και για να προστεθούν ή να διαγραφτούν στοιχεία από αυτές.

Ένα item δεν μπορεί και να εγκλείεται σε μια όψη και να αποκλείεται από αυτήν. Κατά συνέπεια, η υψηλότερου επιπέδου λειτουργία που τροποποιεί τις λίστες των εγκλειόμενων και αποκλειόμενων items θα πρέπει όταν πρόκειται να προστεθεί ένα στοιχείο στη μια από αυτές να ελέγχει αν ήδη υπάρχει και στην άλλη και, στην περίπτωση σύγκρουσης, να περνάει τον έλεγχο πίσω στο χρήστη με κάποιο διαγνωστικό.

Για την παρουσίαση των (ορισμών των) όψεων υπάρχει η μέθοδος

Present(WinWidth,WinHeight,...)

Οι παράμετροι για το παράθυρο παρουσίασης καθορίζονται όπως και για την παρουσίαση πληροφοριών για τους συγγραφείς.

Τελειώνοντας θα πρέπει να τονιστεί ότι η ιδέα να αναπαρασταθούν οι όψεις ως συνηθισμένες οντότητες του συστήματος πληροφοριών πάνω στο οποίο ορίζονται δεν είναι καινούρια. Στις σχεσιακές βάσεις δεδομένων, οι όψεις ορίζονται και αποθηκεύονται ως ερωτήσεις, ακριβώς όπως και στο μοντέλο που προτείνεται εδώ οι όψεις ορίζονται και αποθηκεύονται ως αντικείμενα. Στη σχετική βιβλιογραφία ([NAN88]), παρουσιάζεται η έννοια των *ιδεατών εγγράφων* (conceptual documents): ένα σύστημα ανάκτησης πληροφοριών στο οποίο τα δεδομένα συγκεντρώνονται σε "σακιά". Τα ιδεατά έγγραφα είναι ορισμοί όψεων πάνω στις διαθέσιμες πληροφορίες και αποθηκεύονται και αυτά σε σακιά. Κάθε ιδεατό έγγραφο αποτελείται από ένα σύνολο από *κανόνες εξαγωγής* (extraction rules), που αντιστοιχούν στα κριτήρια επιλογής που ορίστηκαν προηγουμένως για τις όψεις, ένα σύνολο *κανόνων οργάνωσης* (organisation rules), που αντιστοιχούν στις προτάσεις που γίνονται στην επόμενη παράγραφο για την παρουσίαση του περιεχομένου των όψεων με χρήση συνεκτικών γράφων, και ένα σύνολο *κανόνων παρουσίασης* (presentation rules), που αντιστοιχούν στις μεθόδους για την παρουσίαση των items.

Σημειώνεται ότι ο ορισμός όψεων είναι ένα ανοιχτό και ενδιαφέρον πρόβλημα όχι μόνο από την πλευρά των συστημάτων hypermedia, αλλά και από τη γενικότερη πλευρά του object-oriented περιβάλλοντος. Σημαντική δουλειά έχει ήδη γίνει και συνεχίζεται σχετικά με το πώς είναι δυνατόν να

οριστούν όψεις σε ένα τέτοιο περιβάλλον. Κατά κανόνα, αποφασίζεται να οριστούν όψεις που διαπερνούν πολλές κλάσεις της ιεραρχίας που υπάρχει κάθε φορά (ακριβώς όπως και στο δικό μας μοντέλο η κλάση View διαπερνά τις κλάσεις Item και Link). Αυτή η συσχέτιση των κλάσεων υλοποιείται είτε "επιφανειακά", χρησιμοποιώντας κριτήρια επιλογής που αναφέρονται στις κλάσεις τις οποίες διαπερνούν οι όψεις (ακριβώς όπως τα κριτήρια επιλογής που ορίζονται παραπάνω αναφέρονται στις κλάσεις Item και Link) είτε "βαθύτερα", με επεκτάσεις του object-oriented σχήματος. Ένα πολύ ενδιαφέρον παράδειγμα της δεύτερης περίπτωσης μπορεί να βρεθεί στο [SHI89], όπου παρουσιάζεται ένα συνεπές σύνολο επεκτάσεων που επιτρέπει στο κλασικό object-oriented μοντέλο να υποστηρίξει ευθύς εξαρχής και όψεις.

5.5. Χρήση όψεων και περιπλάνηση μέσα σε μια hypermedia βάση δεδομένων

Οι όψεις δημιουργούνται για να μπορούν οι χρήστες να εκμεταλλευτούν καλύτερα τις πληροφορίες που υπάρχουν μέσα στη hypermedia βάση δεδομένων. Οι ίδιοι οι χρήστες, ωστόσο, αντιλαμβάνονται την ύπαρξη των όψεων μόνο όταν μπαίνουν σε μια όψη ή βγαίνουν από αυτήν. Όλο τον υπόλοιπο καιρό, όταν δηλαδή δουλεύουν μέσα σε συγκεκριμένες όψεις, τους παρέχεται μόνο το τμήμα της βάσης δεδομένων που ανταποκρίνεται στους ορισμούς των όψεων αυτών.

Όπως περιγράφτηκε και σε προηγούμενες παραγράφους, μια όψη αποτελείται από ένα σύνολο από items και ένα σύνολο από συνδέσμους, των οποίων τα αρχικά ή/και τελικά items δεν περιέχονται αναγκαστικά μέσα στην όψη (αν και το τελευταίο δείχνει, γενικά, ότι μια όψη δεν έχει οριστεί και τόσο καλά). Έτσι, μια όψη είναι ένας υπογράφος ολόκληρου του hypermedia γράφου και δεν υπάρχει κανείς λόγος για τον οποίο να είναι μια όψη συνεκτική, τη στιγμή μάλιστα που και ολόκληρος ο hypermedia γράφος δεν είναι αναγκαστικά συνεκτικός. Στο επίπεδο των χρηστών, ωστόσο, είναι πολύ σημαντικό να παρουσιάζεται μια όψη ως ένα συνεκτικό γράφημα, για να διευκολύνεται ο προσανατολισμός και η περιπλάνηση των χρηστών μέσα σε αυτήν ([NAN88]).

Προτείνεται, λοιπόν, από όλο το περιεχόμενο μιας όψης να παρουσιάζονται στους χρήστες

(1) όλα τα items που περιέχονται στην όψη, και

(2) από τους συνδέσμους που περιέχονται στην όψη, μόνο εκείνοι των οποίων το αρχικό ή/και το τελικό item περιέχονται στην όψη

(να μην παρουσιάζονται, δηλαδή, οι σύνδεσμοι που περιέχονται στην όψη αλλά το τελικό και το αρχικό τους item δεν περιέχονται στην όψη). Επιπλέον, θα χρησιμοποιούνται πρόσθετα *τόξα-ΚΑΙ* (AND-arcs) και *τόξα-Η* (OR-arcs) για να εξασφαλίζουν ότι ο γράφος που θα σχηματίζεται από τα περιεχόμενα της όψης θα είναι οπωσδήποτε συνεκτικός. Τα τόξα αυτά θα χρησιμοποιούνται ως εξής:

(1) όλα τα items που επιλέγονται με το ίδιο κριτήριο θα ενώνονται με έναν ελάχιστο αριθμό από τόξα-ΚΑΙ, σχηματίζοντας μία *ομάδα-ΚΑΙ* (AND-group)

(2) κάθε item που εγκλείεται άμεσα στην όψη θα θεωρείται μια εκφυλισμένη ομάδα-ΚΑΙ

(3) όλες οι ομάδες-ΚΑΙ θα ενώνονται με έναν ελάχιστο αριθμό τόξων-Η σε μια *ομάδα-Η* (OR-group).

Έτσι, ολόκληρο το περιεχόμενο της όψης θα εμφανίζεται ενωμένο σε μια ομάδα-Η, η οποία θα είναι συνεκτική και από την ύπαρξη και μόνο των τόξων, ακόμα δηλαδή και αν δεν περιέχει καθόλου συνδέσμους.

Είναι ευνόητο, βέβαια, ότι τα τόξα θα πρέπει να παρουσιάζονται στους χρήστες διαφορετικά από τους συνδέσμους, εφόσον τα πρώτα είναι νοητά αντικείμενα του interface και δεν έχουν καμιά φυσική αναπαράσταση ούτε φέρουν πληροφορία, ενώ οι τελευταίοι είναι υπαρκτά αντικείμενα της βάσης δεδομένων, που αναπαρίστανται φυσικά και φέρουν σημασιολογική πληροφορία. Οι χρήστες θα ξέρουν ότι μπορούν να διασχίσουν τόσο τους συνδέσμους όσο και τα τόξα, αλλά μπορούν να επεξεργαστούν, να δημιουργήσουν και να διαγράψουν μόνο συνδέσμους.

Θα πρέπει να σημειωθεί ότι αυτός ο αλγόριθμος συνεκτικής αναπαράστασης δημιουργεί επίπεδα γραφήματα, εκτός βέβαια και αν ένα κομμάτι της βάσης hypermedia που περιέχεται μέσα σε μια όψη έχει το ίδιο μη επίπεδη δομή. Έτσι, πολλά από τα προβλήματα αναπαράστασης που αναφέρονται στο [TAN89α] απλοποιούνται σημαντικά.

Σύμφωνα με το παραπάνω σχήμα, η περιπλάνηση μέσα σε μια όψη επιτυγχάνεται είτε με τη διάσχιση των συνδέσμων, όταν αυτοί υπάρχουν, είτε με τη διάσχιση τόξων-ΚΑΙ μέσα στην ίδια ομάδα-ΚΑΙ, είτε με τη διάσχιση τόξων-Η ανάμεσα σε διαφορετικές ομάδες-ΚΑΙ. Η συνολική κίνηση τώρα μέσα σε μια όψη μπορεί να επιτευχθεί είτε με περιπλάνηση είτε με ερωτήσεις. Οι ερωτήσεις μπορεί να βασίζονται απλώς σε χαρακτηριστικές των items, ή να περιέχουν πιο σύνθετα κριτήρια, όπως αυτά που χρησιμοποιούνται στους

ορισμούς των όψεων. Ενώ η περιπλάνηση καταλήγει σε εντολές διάσχισης τόξων ή συνδέσμων, οι ερωτήσεις καταλήγουν σε εντολές μεταφοράς στα items που επιστρέφονται ως αποτέλεσμα. Έτσι, μπαίνουν τα ζητήματα που περιγράφονται παρακάτω για την κίνηση μέσα στις όψεις.

Όταν ένας χρήστης μπαίνει σε μια όψη, πρέπει να τοποθετηθεί από το σύστημα σε ένα item που να ανήκει στην όψη αυτή. Αυτό, όπως είπαμε, θα είναι το home item της όψης αυτής. Το ποιο είναι το home item δεν αποθηκεύεται σε καμιά ξεχωριστή μεταβλητή του ορισμού της όψης, διότι μπορεί να αλλάζει όποτε αλλάζει ο ορισμός της όψης ή/και τα items που υπάρχουν σε ολόκληρη τη βάση δεδομένων. Το home item μιας όψης πρέπει να εντοπίζεται κάθε φορά που ένας χρήστης ενεργοποιεί την όψη αυτήν. Ορίζεται, λοιπόν, στην κλάση View η μέθοδος

GetHome()

η οποία θα εντοπίζει το home item μιας όψης. Εφόσον, όμως, μια κενή όψη δεν περιέχει κανένα item και κατά συνέπεια ούτε και home item, η μέθοδος θα αποτυγχάνει για μια κενή όψη. Έτσι, η λειτουργία υψηλότερου επιπέδου που ελέγχει την είσοδο σε όψεις θα πρέπει να απαγορεύει την είσοδο σε κενές όψεις.

Για την περιπλάνηση μέσα από συνδέσμους μπορούν να χρησιμοποιηθούν οι μέθοδοι ανάκτησης των αντικειμένων αυτών. Για την περιπλάνηση μέσω τόξων μπορούν να οριστούν οι μέθοδοι

GetNextAND()

που θα μεταφέρει το χρήστη στο κυκλικά επόμενο item της ομάδας-ΚΑΙ, και

GetNextOR()

που θα μεταφέρει το χρήστη στο πρώτο item της κυκλικά επόμενης ομάδας-ΚΑΙ. Οι ομάδες-ΚΑΙ διατάσσονται με τη σειρά που έχουν στον ορισμό της όψης (α) τα εγκλειόμενα items και (β) τα κριτήρια επιλογής items, δηλαδή το πρώτο εγκλειόμενο item θα είναι η πρώτη (εκφυλισμένη) ομάδα-ΚΑΙ, το δέκατο και τελευταίο (π.χ.) εγκλειόμενο item θα είναι η δέκατη ομάδα-ΚΑΙ, το πρώτο κριτήριο επιλογής items θα παράγει την ενδέκατη ομάδα-ΚΑΙ, κοκ.

Παρόλο που οι δύο τελευταίες μέθοδοι αφορούν καθαρά το user interface είναι σκόπιμο να οριστούν στην κλάση View, εφόσον θα χρειαστούν έτσι κι αλλιώς τον ορισμό μιας όψης για να δουλέψουν.

Το user interface θα συνδυάζει όλα τα εργαλεία περιπλάνησης και ερωτήσεων ώστε να εξασφαλίζει την κίνηση των χρηστών μέσα στις όψεις. Είναι, ωστόσο, αναμενόμενο, να φτάνουν οι χρήστες στα όρια των όψεών τους και να θέλουν να τα ξεπεράσουν. Αυτό μπορεί να συμβεί σε δύο περιπτώσεις:

(1) ένας χρήστης προσπαθεί, διασχίζοντας ένα σύνδεσμο, να φτάσει σε ένα item που δεν περιέχεται στην ενεργή όψη, ή

(2) ένας χρήστης προσπαθεί, μετά από μια ερώτηση, να μεταφερθεί σε ένα item που δεν περιέχεται στην ενεργή όψη (αν και αυτό δεν πρόκειται ποτέ να γίνει αν οι ερωτήσεις αποτιμώνται μόνο ως προς τα items της ενεργής όψης).

Και στις δύο περιπτώσεις το σύστημα μπορεί να δώσει μια προειδοποίηση, να βρει την όψη ή τις όψεις όπου ανήκει το νέο item, και να ζητήσει από το χρήστη να επιβεβαιώσει ότι θέλει να μεταφερθεί σε μια από αυτές τις όψεις δίνοντας το σύνθημά της. Από εκεί και πέρα είναι θέμα του χρήστη αν θα αλλάξει όψη ή θα μείνει εκεί που είναι.

5.6. Βιβλιογραφία

1. [DAT84] C. J. Date, "An Introduction to Database Systems", Addison-Wesley, 1984.
2. [HAL88] Frank G. Halasz, "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems", CACM, τεύχος 31/7, Ιούλιος 1988.
3. [KOR86] H. Korth, A. Silberschatz, "Database System Concepts", 1986.
4. [NAN88] J.Nanard, M.Nanard, H.Richy, "Conceptual Documents: A Mechanism for Specifying Active Views in Hypertext", ACM Conf. on Document Processing Systems, 1988.
5. [PAR89] Kamran Parsaye, M. Chingel, S. Khoshafian και H. Wong, "Intelligent Databases", Wiley, 1989.
6. [SHI89] J.J.Shilling, P.F.Sweeney, "Three Steps to Views: Extending the Object-Oriented Paradigm", πρακτικά OOPSLA'89, Οκτώβριος 1989.
7. [TAN89α] Gary Tanner, "Navigating Through Hyperchaos", SAFE/HYP/HCI-pap/GT240789, Ιούλιος 1989.

6. ΤΗΡΗΣΗ ΕΚΔΟΧΩΝ ΣΕ HYPERMEDIA ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

6.1. Απαιτήσεις για την τήρηση εκδοχών σε βάσεις δεδομένων

Μια λειτουργία που τείνει να καθιερωθεί σε όλα τα περιβάλλοντα που υποστηρίζουν διαχείριση δεδομένων ή ανάπτυξη λογισμικού, είναι η *τήρηση εκδοχών* (versioning). Το να διατηρούνται *εκδοχές* (versions) είναι ένα πολύ χρήσιμο χαρακτηριστικό, αν ληφθεί υπόψη το πόσες φορές αλλάζει ένα έγγραφο κατά την επεξεργασία του, ένα πρόγραμμα μετά από διαδοχικές δοκιμές κλπ. Επιπλέον, από τη στιγμή που υπάρχουν οι προηγούμενες εκδοχές κάποιων αρχείων περιορίζονται οι πιθανότητες να χαθούν πληροφορίες από αβλεψίες των χρηστών, αλλά μειώνονται και οι περιπτώσεις στις οποίες οι χρήστες θέλουν να *ακυρώσουν* (undo) τις ενέργειές τους. Η μεγάλη σπουδαιότητα της τήρησης εκδοχών οδήγησε στο να υποστηριχθεί σε επίπεδο λειτουργικών συστημάτων (π.χ. VMS) αλλά και σε πολλά περιβάλλοντα προγραμματισμού, επεξεργασίας εγγράφων, κ.ά.. Μέσα σε αυτά τα πλαίσια, η τήρηση εκδοχών άρχισε να εμφανίζεται ως επιθυμητό χαρακτηριστικό και για τα συστήματα hypertext και τις βάσεις δεδομένων hypermedia. Δεν μπορούμε να πούμε ότι η τήρηση εκδοχών έχει καθιερωθεί για τα συστήματα αυτά, καθώς πολλά γνωστά προϊόντα hypertext και hypermedia δεν την υποστηρίζουν (πχ.Notecards, HyperCard), από την άλλη πλευρά όμως έχει υλοποιηθεί, έστω και σε περιορισμένη κλίμακα, σε αρκετές περιπτώσεις. (πχ.KMS, Neptune, Intermedia, PIE).

Το να διατηρούμε εκδοχές σε μια hypermedia βάση δεδομένων αυξάνει την πολυπλοκότητά της αφού απαιτεί, εκτός από τις μεθόδους που χρειάζονται για την επίτευξη των στόχων που είδαμε ως τώρα, και μεθόδους που να μπορούν:

(1) να δημιουργήσουν μια καινούρια εκδοχή,

(2) να διαγράψουν από τη βάση δεδομένων μια εκδοχή που δε χρειάζεται πια,

(3) να ανακτήσουν μια εκδοχή από την βάση δεδομένων και

(4) να κάνουν *συλλογή απορριμμάτων* (garbage collection), δηλαδή να απομακρύνουν όσες εκδοχές δε χρειάζονται πλέον.

Λαμβάνοντας υπόψη το ότι η διατήρηση εκδοχών αυξάνει τις απαιτήσεις σε χώρο αποθήκευσης, με αποτέλεσμα να χρειάζονται τεχνικές συμπίεσης, και το ότι η προσπέλαση των εκδοχών πρέπει να είναι γρήγορη, καταλήγουμε στο συμπέρασμα ότι οι προηγούμενες μέθοδοι θα είναι αρκετά πολύπλοκες.

Η πρώτη εκδοχή ενός αντικειμένου θα δημιουργείται όταν το ίδιο το αντικείμενο δημιουργείται για πρώτη φορά. Τροποποιήσεις του αντικειμένου μετά τη δημιουργία του, θα καταλήγουν στην δημιουργία μιας καινούριας εκδοχής. Η αίτηση για διαγραφή ενός αντικειμένου από την βάση δε θα οδηγεί πλέον στη φυσική διαγραφή του αντικειμένου, αλλά στη δημιουργία μιας καινούριας κενής εκδοχής γι' αυτό. Ένα αντικείμενο θα διαγράφεται φυσικά από τη βάση δεδομένων μόνο αν ένας χρήστης διαγράψει όλες τις εκδοχές του, ή καθορίσει κατά τη διάρκεια της συλλογής απορριμμάτων ότι όλες οι εκδοχές του αντικειμένου αυτού αποτελούν απορρίμματα. Αίτηση για ανάκτηση ενός αντικειμένου από τη βάση δεδομένων θα πρέπει να επιστρέφει την τελευταία εκδοχή του αντικειμένου αυτού, εκτός αν ο χρήστης έχει καθορίσει ότι θέλει κάποια συγκεκριμένη εκδοχή. Η συλλογή απορριμμάτων θα έχει στόχο να απομακρύνει από τη βάση δεδομένων όσες εκδοχές αντικειμένων θεωρούνται περιττές, απελευθερώνοντας με αυτόν τον τρόπο χώρο αποθήκευσης.

Δύο ακόμα προβλήματα σχεδιασμού σχετικά με την τήρηση εκδοχών είναι (i) ο μηχανισμός με τον οποίο θα συνδέονται οι εκδοχές μεταξύ τους, και (ii) η δομή την οποία θα έχει το σύνολο των εκδοχών.

Η σύνδεση των εκδοχών μπορεί να γίνει είτε με *συνδέσμους ενός ειδικού τύπου* ([HAN87]), είτε με *δείκτες* (pointers). Στην πρώτη περίπτωση, η χρήση ενός νέου τύπου συνδέσμων, και μάλιστα μη σημασιολογικών, θα είχε επιπτώσεις σε ολόκληρο το σχεδιασμό και θα αύξανε αδικαιολόγητα την πολυπλοκότητα. ;Ετσι, κρίνεται προτιμότερη χρήση δεικτών.

;Οσον αφορά τη δομή που θα σχηματίζουν οι διαδοχικές εκδοχές, στη βιβλιογραφία αναφέρονται τρεις περιπτώσεις ([HAL88]): είτε οι εκδοχές ενώνονται σε μια *στοίβα* (π.χ. KMS ([AKS88]), Neptune), είτε σε ένα *ιεραρχικό δέντρο* (λύση που υιοθετείται κυρίως στην ανάπτυξη λογισμικού), είτε σε ένα *γενικό ακυκλικό γράφο* (π.χ. PIE). Οι δύο τελευταίες επιλογές, ωστόσο, δημιουργούν προβλήματα σχετικά με τις

σημασιολογικές συνδέσεις των εκδοχών και δε θα πρέπει να ακολουθούνται αν δεν υπάρχει συγκεκριμένος λόγος. Για τη γενική περίπτωση μιας hypermedia βάσης δεδομένων, θεωρείται σκόπιμο να σχηματίζουν οι εκδοχές μια συνδεδεμένη λίστα.

6.2. Τεχνικές για τήρηση εκδοχών δεδομένων multimedia

Το να υποστηρίζει κάποια βάση δεδομένων την τήρηση εκδοχών για τα δεδομένα που αποθηκεύονται σε αυτή, γίνεται πιο δύσκολο όταν τα δεδομένα αυτά δεν είναι πλέον *εγγραφές* (records), όπως στις σχεσιακές, ιεραρχικές και δικτυωτές βάσεις δεδομένων, αλλά *δεδομένα multimedia* χωρίς συγκεκριμένη δόμηση ούτε καθορισμένο μέγεθος. Το βασικό πρόβλημα με τα δεδομένα multimedia, και ειδικότερα με τον ήχο, την εικόνα και το video, είναι το ότι τείνουν να καταλάβουν πολύ χώρο αποθήκευσης. 'Ενα κείμενο μιας σελίδας, για παράδειγμα, χρειάζεται μόνο 2 Kbytes περίπου για να αποθηκευτεί σαν κείμενο, αλλά 1 Mbyte για να αποθηκευτεί σαν ψηφιοποιημένη εικόνα ή σα δειγματοληπτημένη ομιλία κάποιου που το διαβάζει σε κανονικό ρυθμό. Εύκολα καταλαβαίνουμε ότι, χωρίς τεχνικές συμπίεσης, σύντομα ο χώρος αποθήκευσης θα εξαντλούνταν.

Μια τεχνική που μπορεί να βοηθήσει στον περιορισμό του χώρου αποθήκευσης που απαιτείται για τις εκδοχές των multimedia δεδομένων είναι η *τεχνική δέλτα* (delta technique). Σύμφωνα με την τεχνική αυτή κρατάμε μόνο μία *πλήρη εκδοχή* (full version) της πληροφορίας και τις *αλλαγές* (deltas) που διαφοροποιούν την εκδοχή αυτή από τις υπόλοιπες. Ανάλογα με το αν κρατάμε την αρχική εκδοχή της πληροφορίας και για κάθε εκδοχή τις αλλαγές ως προς την προηγούμενη χρονικά, ή την τελική εκδοχή της πληροφορίας και για κάθε εκδοχή τις αλλαγές ως προς την επόμενη χρονικά, η τεχνική δέλτα χαρακτηρίζεται ως *δέλτα προς τα εμπρός* (forward delta) ή *δέλτα προς τα πίσω* (backward delta) αντίστοιχα. Η τεχνική δέλτα καταφέρνει να εξοικονομεί χώρο λόγω του ότι συνήθως μόνο ένα μικρό τμήμα των δεδομένων αλλάζει και, συνεπώς, η πληροφορία που απαιτείται για την περιγραφή των αλλαγών είναι πολύ λιγότερη από την πληροφορία που χρειάζεται για να περιγραφούν τα δεδομένα στο σύνολό τους. ('Ενας μηχανισμός τήρησης εκδοχών που χρησιμοποιεί δέλτα προς τα πίσω για αρχεία κειμένου περιγράφεται στο [KER84] και βασίζεται στο εργαλείο diff του περιβάλλοντος Unix. Επίσης, ένα μαθηματικό μοντέλο για τις εκδοχές και τα δέλτα στα πλαίσια των συστημάτων hypertext μπορεί να βρεθεί στο [GARG88]).

Η τεχνική δέλτα χρησιμοποιείται από την πλειονότητα των βάσεων δεδομένων hypermedia που υποστηρίζουν τήρηση εκδοχών. Έχουν τεθεί, μάλιστα, και τα ζητήματα (i) να μπορούν να χρησιμοποιούνται τα δέλτα και σε ερωτήσεις προς τη βάση δεδομένων, και (ii) να μπορούν να δημιουργούνται σύνδεσμοι από και προς τα δέλτα ([HAL88]). Το προτεινόμενο μοντέλο για τήρηση εκδοχών υλοποιεί τα δέλτα ως συνηθισμένα αντικείμενα της βάσης δεδομένων με το σκεπτικό ότι το δέλτα ενός αντικειμένου θα πρέπει να είναι ένα αντικείμενο της ίδιας κλάσης. Θα πρέπει να σημειωθεί, ωστόσο, ότι τα ίδια τα δέλτα δεν μπορούν να προσπελαστούν άμεσα, δηλαδή δεν μπορούν να ανακτηθούν, να τροποποιηθούν ή να διαγραφούν με τη χρήση των μεθόδων της αντίστοιχης κλάσης. Τα δέλτα μπορούν μόνο να χρησιμοποιούνται για τον υπολογισμό των αντίστοιχων εκδοχών, ή να διαγραφούν. Αυτό σημαίνει (i) πως μπορεί να δημιουργηθεί σύνδεσμος από ή προς μια ενδιάμεση εκδοχή ενός item, δημιουργώντας όμως μια καινούρια εκδοχή, η οποία θα προστεθεί τελευταία στη λίστα των εκδοχών του item αυτού, και (ii) πως όταν γίνεται μια ερώτηση για κάποια items δεν εξετάζονται όλες οι εκδοχές όλων των items της βάσης (κάτι τέτοιο θα ήταν απαγορευτικά χρονοβόρο), αλλά μόνο η τελευταία εκδοχή κάθε item της βάσης. Επειδή όμως έτσι, αν η πληροφορία που ζητάμε υπάρχει σε μια παλιότερη εκδοχή κάποιου item δε θα τη βρούμε, θα πρέπει να μπορεί ένας χρήστης να επιλέξει τον έλεγχο όλων των εκδοχών για κάποια ή όλα τα items της βάσης.

Είναι γεγονός ότι σε βάσεις δεδομένων που υποστηρίζουν τήρηση εκδοχών συνήθως ζητείται η ανάκτηση της τελευταίας ή, γενικότερα, των πιο προσφάτων εκδοχών, και πιο σπάνια των παλιότερων. Αυτό κάνει την τεχνική του δέλτα προς τα πίσω πιο ελκυστική για ένα τέτοιο περιβάλλον, αφού η τελευταία εκδοχή θα είναι διαθέσιμη ανά πάσα στιγμή ενώ λίγοι πρόσθετοι υπολογισμοί θα είναι απαραίτητοι για την ανακατασκευή κάποιας πρόσφατης εκδοχής. Επίσης, αν κατά τη διαδικασία της συλλογής απορριμμάτων θέλουμε να απομακρύνουμε την παλιότερη εκδοχή, αρκεί να απομακρύνουμε το αντίστοιχο δέλτα. Αντίθετα, αν χρησιμοποιηθεί η τεχνική του δέλτα προς τα εμπρός είναι απαραίτητη η προσπέλαση τόσο στην αρχική πληροφορία όσο και στο σύνολο των περιγραφών των αλλαγών προκειμένου να κατασκευαστεί η πιο πρόσφατη εκδοχή, πράγμα που συνεπάγεται πολύ σημαντική χρονική επιβάρυνση, τόσο για την ανάκτηση των πληροφοριών που περιγράφουν τις διαφορές, όσο και για τους

υπολογισμούς που πρέπει να γίνουν.

Μια βελτίωση που μπορεί να γίνει στην τεχνική των δέλτα προς τα εμπρός είναι το να αποθηκεύουμε την αρχική εκδοχή, τα δέλτα προς τα εμπρός για όλες τις εκδοχές από τη δεύτερη μέχρι και την προτελευταία, και ολόκληρη την τελευταία εκδοχή. Το σχήμα αυτό, που μπορεί να ονομαστεί *ενισχυμένο δέλτα προς τα εμπρός* , συνεπάγεται μεν κάποια επιπλέον επιβάρυνση σε χώρο μνήμης και χρόνο συντήρησης των εκδοχών, αλλά βελτιώνει σημαντικά την απόδοση στην περίπτωση που ζητάμε συνήθως την πιο πρόσφατη εκδοχή και οι παλιότερες εκδοχές απέχουν πολλά δέλτα από τις τελευταίες.

Η σημερινή τεχνολογία, ωστόσο, δεν προσφέρεται για την εφαρμογή του δέλτα προς τα πίσω σε μια multimedia βάση δεδομένων: λόγω του μεγάλου όγκου της πληροφορίας που απαιτείται για την περιγραφή multimedia αντικειμένων οδηγούμαστε αναγκαστικά στη λύση της *οπτικής αποθήκευσης* (optical storage), και τα σημερινά μέσα οπτικής αποθήκευσης είναι *μέσα ανάγνωσης μόνο* (read only media) . Αυτό σημαίνει ότι είναι αδύνατο να αλλάξουμε τα δεδομένα που βρίσκονται στο οπτικό μέσο αποθήκευσης, δηλαδή δεν μπορούμε να αποθηκεύσουμε νέες εκδοχές ενός αντικειμένου πάνω στο οπτικό μέσο. Συνεπώς, θα πρέπει να αποθηκεύσουμε οπτικά την αρχική εκδοχή του αντικειμένου και να κρατήσουμε τα δέλτα προς τα εμπρός που χρειάζονται για την ανακατασκευή των πιο πρόσφατων εκδοχών σε ένα συμβατικό μαγνητικό μέσο, ώστε να μπορούν, αν χρειαστεί, να σβήνονται κατά τη συλλογή απορριμμάτων. Η δημιουργία στο μέλλον οπτικών μέσων αποθήκευσης με δυνατότητες ανάγνωσης και εγγραφής (read/write media) θα επιτρέψει την εφαρμογή της τεχνικής του δέλτα προς τα πίσω ή του ενισχυμένου δέλτα προς τα εμπρός σε multimedia βάσεις δεδομένων.

6.3. Μοντέλο για τήρηση εκδοχών σε μία hypermedia βάση δεδομένων

Γενικά σε μια βάση δεδομένων hypermedia διακρίνουμε δύο είδη δεδομένων: τα *ακατέργαστα δεδομένα* (raw data), δηλαδή το περιεχόμενο των hypermedia items, και τα *σημασιολογικά δεδομένα* (semantic data), δηλαδή τα μεταδεδομένα που καθορίζουν τα χαρακτηριστικά των hypermedia items και περιγράφουν τις λογικές συνδέσεις μεταξύ τους. Τα ακατέργαστα δεδομένα είναι τύπου multimedia ενώ ο τύπος των σημασιολογικών δεδομένων εξαρτάται από την υλοποίηση του μοντέλου hypermedia (μπορούμε πάντως να περιμένουμε ότι τα σημασιολογικά δεδομένα θα είναι συνήθως κείμενο ή *συμβολοσειρές* (strings)). Η τεχνική που θα χρησιμοποιηθεί για την τήρηση εκδοχών των αντικειμένων hypermedia που αποθηκεύονται στη βάση δεδομένων εξαρτάται σε μεγάλο βαθμό από το είδος των αλλαγών που ο χρήστες μπορούν να κάνουν σε αυτά και πιο συγκεκριμένα από το αν οι χρήστες έχουν το δικαίωμα να αλλάζουν τα ακατέργαστα δεδομένα ή όχι.

Έχοντας μιλήσει για ένα object-oriented μοντέλο είναι απαραίτητο να τονίσουμε ότι όλες οι εκδοχές ενός αντικειμένου θα έχουν τον ίδιο *προσδιοριστή* (identifier)˙ το αντίθετο θα παραβίαζε την object-oriented φιλοσοφία και θα δημιουργούσε προβλήματα στους χρήστες, καθώς ο προσδιοριστής ενός αντικειμένου υποτίθεται ότι προσδιορίζει μοναδικά το αντικείμενο αυτό, και, κατ' επέκταση, όλες τις εκδοχές αυτού του αντικειμένου. Στο μοντέλο της hypermedia βάσης δεδομένων που αναπτύξαμε ως τώρα διακρίνουμε τέσσερις βασικούς τύπους (κλάσεις) αντικειμένων: τα items (και όλες τις υποκλάσεις τους), τους συνδέσμους (links), τους συγγραφείς (authors) και τις όψεις (views). Εύκολα βλέπουμε ότι τα στιγμιότυπα της κλάσης Author δεν απαιτούν τήρηση εκδοχών αφού τα στοιχεία τους συνήθως δε μεταβάλλονται, αλλά, και αν μεταβληθούν, δε μας ενδιαφέρουν οι προηγούμενες τιμές τους. Τα στιγμιότυπα της κλάσης View έχουν σκοπό να παρουσιάζουν στους χρήστες μια άποψη της βάσης δεδομένων που κάποιος συγγραφέας θεωρεί χρήσιμη ή ενδιαφέρουσα. Ανάλογα με το αν μας ενδιαφέρει το ιστορικό των αλλαγών στις όψεις ή όχι, μπορούμε να ορίσουμε το αν θα κρατάμε ή όχι εκδοχές των όψεων.

Με μια προσεκτική εξέταση φαίνεται ότι για τα στιγμιότυπα της κλάσης Link δε χρειάζεται τήρηση εκδοχών. Στην πράξη, ένας χρήστης δεν μπορεί να αλλάξει άλλα δεδομένα ενός συνδέσμου, εκτός από τον τύπο, το βάρος και τις άγκυρές του. Όμως η αλλαγή των τιμών για τις άγκυρες ενός συνδέσμου δε συνεπάγεται τη δημιουργία ενός νέου φυσικού αντιγράφου του συνδέσμου, αφού οι τιμές αυτές φυλάσσονται στα αντίστοιχα items, και όσον αφορά τις αλλαγές στον τύπο ενός συνδέσμου και στο βάρος του, μπορούμε να θεωρήσουμε ότι μια τέτοια αλλαγή αφορά όλες τις εκδοχές των items που συνδέει ο σύνδεσμος αυτός, και κατά συνέπεια δε χρειάζεται ούτε τώρα να κρατήσουμε τις παλιές τιμές. Περιπτώσεις που αυτή η παραδοχή δεν ισχύει θα αντιμετωπίζονται με τη διαγραφή του συνδέσμου από τις εκδοχές του αρχικού και του τελικού του item και τη δημιουργία ενός νέου συνδέσμου με τον επιθυμητό τύπο (διαφορετικές εκδοχές του ίδιου item μπορούν να έχουν διαφορετικές τιμές στη μεταβλητή

AnchoredLinks).

Σύμφωνα με το μοντέλο αυτό, οι διάφορες λογικές λειτουργίες της τήρησης εκδοχών απεικονίζονται στο φυσικό επίπεδο ως εξής:

(1) Οποιαδήποτε μεταβολή σε στιγμιότυπο της κλάσης Author απεικονίζεται φυσικά στη βάση δεδομένων. Για τα αντικείμενα αυτής της κλάσης δεν κρατάμε τις παλιότερες εκδοχές. Με τον όρο μεταβολή , εδώ, εννοούμε τις αλλαγές μετά από μια ολόκληρη *σύνοδο τροποποιήσεων* (editing session), και όχι μια απλή αλλαγή σε κάποια από τις μεταβλητές ενός αντικειμένου της κλάσης Author˙ έτσι οι αλλαγές αυτές περνάνε στη βάση μετά τη λήξη της συνόδου. Η σύνοδος τροποποιήσεων τερματίζεται είτε έμμεσα, όταν ο χρήστης ζητήσει να βγει από το περιβάλλον του editor που χρησιμοποιεί, είτε άμεσα, όταν ο χρήστης ζητήσει να σωθούν στη βάση οι αλλαγές που έκανε. Η ίδια τακτική για το πέρασμα των αλλαγών στη βάση των δεδομένων ισχύει και για όσα αναφέρονται παρακάτω.

(2) Αν μας ενδιαφέρει το ιστορικό των μεταβολών στις όψεις, τότε κάθε μεταβολή στον ορισμό μιας όψης θα δημιουργεί μια καινούρια εκδοχή της όψης. Διαγραφή κάποιας όψης θα σημαίνει τη δημιουργία μιας καινούριας εκδοχής της όψης στην οποία οι λίστες CriteriaList, IncItems και ExcItems θα είναι κενές. Αν δε μας ενδιαφέρει το ιστορικό των μεταβολών τότε, όταν οι μεταβολές που γίνονται στον ορισμό μιας όψης περνάνε στη βάση, ο παλιός ορισμός της όψης θα αντικαθίσταται από τον καινούριο, ενώ σε περίπτωση διαγραφής κάποιας όψης, ο ορισμός της όψης θα διαγράφεται φυσικά από την βάση δεδομένων.

(3) Μεταβολές σε στιγμιότυπο της κλάσης Item θα δημιουργούν μια καινούρια εκδοχή του item αυτού. Η διαγραφή ενός item θα συνεπάγεται τη δημιουργία μιας κενής εκδοχής του item αυτού, χωρίς περιεχόμενα, ιδιότητες και συνδέσμους, ενώ η μεταβλητή ContentType του item αυτού θα παίρνει την τιμή dummy . Μπορούμε δηλαδή να πούμε ότι η διαγραφή ενός non-dummy item θα δημιουργεί ένα dummy item.

(4) Η μεταβολή του τύπου ή του βάρους κάποιου συνδέσμου θα απεικονίζεται άμεσα στη βάση και, ως εκ τούτου, θα επηρεάζει σημασιολογικά τα items τα οποία ο σύνδεσμος αυτός ενώνει (με τον όρο items, εδώ, εννοούμε όλες τις εκδοχές του αρχικού και του τελικού item που περιλαμβάνουν στη μεταβλητή AnchoredLinks τον προσδιοριστή του συνδέσμου που τροποποιήθηκε). Αίτηση για διαγραφή ενός συνδέσμου από ένα item θα καταλήγει στη δημιουργία μιας νέας εκδοχής του item που στη μεταβλητή AnchoredLinks δε θα περιλαμβάνει τον προσδιοριστή του συνδέσμου του οποίου ζητήθηκε η διαγραφή. Ο σύνδεσμος θα εξακολουθεί να υπάρχει φυσικά και να είναι προσπελάσιμος από τις προηγούμενες εκδοχές του item καθώς και από το item που βρισκόταν στο άλλο άκρο του, αν είναι αμφικατευθυνόμενος.

(5) Η διάσχιση ενός συνδέσμου ανάγεται τώρα στην ανάκτηση της πρώτης (από την περισσότερο προς τη λιγότερο πρόσφατη) εκδοχής του item που βρίσκεται στο άλλο άκρο του συνδέσμου και περιέχει τον προσδιοριστή του συνδέσμου στη μεταβλητή AnchoredLinks.

(6) Ο έλεγχος για το αν κάποιο item ανήκει σε μια όψη μπορεί να περιλαμβάνει μόνο έλεγχο για την τελευταία εκδοχή του item, ή έλεγχο για όλες τις εκδοχές του item ή, ως ενδιάμεση λύση, έλεγχο για την πιο πρόσφατη εκδοχή από κάθε συγγραφέα.

6.4. Συλλογή απορριμμάτων σε μια hypermedia βάση δεδομένων

Ο τρόπος εργασίας που αναπτύχθηκε προηγουμένως, σε συνδυασμό με το ότι υπάρχει η δυνατότητα να απομακρύνονται από τη βάση δεδομένων όσες εκδοχές θεωρούνται περιττές, μπορεί να οδηγήσει στη δημιουργία συνδέσμων που να μην είναι προσπελάσιμοι από κανένα item, ή που ένα από τα δύο items που συνέδεαν να μην υπάρχει πλέον. Επιπλέον, καθώς το σύστημα χρησιμοποιείται και δημιουργούνται, με κάποιο ρυθμό, καινούριες εκδοχές για διάφορα items, όλο και λιγότεροι χρήστες θα εξακολουθούν να αναφέρονται στις παλιότερες εκδοχές οι οποίες, από ένα σημείο και μετά, θα είναι εντελώς άχρηστες. Τους συνδέσμους που έχουν χάσει το ένα ή και τα δύο items που συνέδεαν, και τις άχρηστες εκδοχές των items, θα αναλάβει να απομακρύνει από την βάση δεδομένων η λειτουργία της συλλογής απορριμμάτων. Μια μέθοδος που μπορεί να χρησιμοποιηθεί μοιάζει με την τεχνική "σημάδεψε και σκούπισε " ("mark and sweep") ([COX86]) , και προσαρμόζεται στο μοντέλο που έχει περιγραφτεί, ως εξής:

Αρχικά ελέγχεται ποιες εκδοχές των items πρέπει να κρατηθούν. Ο έλεγχος γίνεται σύμφωνα με ορισμένα κριτήρια που περιγράφουν τις χρήσιμες εκδοχές (ποιος είναι ο συγγραφέας, πότε έγινε η

τελευταία αλλαγή, πόσες εκδοχές έχει αποφασιστεί να κρατούνται ανά item, πόσες εκδοχές έχει αποφασιστεί να κρατούνται ανά συγγραφέα για ένα item, κλπ.). Συγκεκριμένα, για την επιλογή των εκδοχών των items που θα σβηστούν θα χρησιμοποιείται ο εξής αλγόριθμος:

- για κάθε item σβήνουμε όσες εκδοχές έχουν δημιουργηθεί πριν από το χρονικό κατώφλι που έχει τεθεί, αλλά αν όλες οι εκδοχές ενός συγκεκριμένου συγγραφέα έχουν δημιουργηθεί πριν από το κατώφλι αυτό κρατάμε την πιο πρόσφατη εκδοχή του συγγραφέα αυτού
- αν οι εκδοχές που απομένουν είναι περισσότερες από το μέγιστο πλήθος εκδοχών που θέλουμε να υπάρχουν για κάθε item, σβήνουμε τις λιγότερο πρόσφατες μέχρι να φτάσουμε στα επιθυμητά όρια, αλλά αν κάποια εκδοχή είναι η τελευταία που απομένει για τον αντίστοιχο συγγραφέα την κρατάμε (έτσι, βέβαια, θα διατηρείται πάντα μία τουλάχιστον εκδοχή ανά συγγραφέα˙ αν αυτό δημιουργεί προβλήματα, μπορούν να σβήνονται οι πολύ παλιές εκδοχές όχι με συλλογή απορριμμάτων αλλά με άμεσες εντολές διαγραφής)
- αν στις εκδοχές που απομένουν υπάρχουν για κάποιο συγγραφέα περισσότερες εκδοχές από το μέγιστο πλήθος εκδοχών που θέλουμε να υπάρχουν σε κάθε item για ένα συγγραφέα, σβήνουμε τις λιγότερο πρόσφατες εκδοχές αυτού του συγγραφέα μέχρι να φτάσουμε στα επιθυμητά όρια.

Αφού γίνει η επιλογή των χρήσιμων και των άχρηστων εκδοχών, για κάθε εκδοχή που πρέπει να κρατηθεί σημαδεύονται οι προσδιοριστές των συνδέσμων που ξεκινούν από αυτήν ή καταλήγουν σε αυτήν. Τέλος, απομακρύνονται από τη βάση δεδομένων όσες εκδοχές θεωρήθηκαν περιττές, καθώς και κάθε σύνδεσμος του οποίου ο προσδιοριστής δεν περιλαμβάνεται σε καμιά εκδοχή του αρχικού ή του τελικού του item. Στην περίπτωση που κάποιο item απομακρυνθεί εντελώς από την βάση δεδομένων (δηλαδή η βάση δεν περιλαμβάνει πλέον καμία εκδοχή του item αυτού), θα πρέπει να ελέγχονται οι ορισμοί των όψεων για την περίπτωση να υπάρχει ο προσδιοριστής του item αυτού στις μεταβλητές IncItems και ExcItems. Στην περίπτωση αυτή, ο προσδιοριστής θα πρέπει να απομακρύνεται, η απομάκρυνση αυτή όμως δε θα δημιουργεί καινούρια εκδοχή της αντίστοιχης όψης. Οι όψεις θα χαρακτηρίζονται ως απορρίμματα με βάση το ποιος είναι ο συγγραφέας τους και το πότε έγινε η τελευταία αλλαγή στον ορισμό τους, και ίσως τον αριθμό των συνολικών εκδοχών και των εκδοχών ανά συγγραφέα που θέλουμε να κρατάμε για κάθε όψη), ενώ στιγμιότυπα της κλάσης Author δε θα απομακρύνονται κατά τη διάρκεια της συλλογής απορριμμάτων. Αν πρέπει να διαγραφτεί κάποιο στιγμιότυπο της κλάσης Author από τη βάση δεδομένων θα χρειάζεται ξεχωριστή αίτηση διαγραφής.

Η συλλογή απορριμμάτων είναι μια πολύπλοκη και χρονοβόρα διαδικασία, η οποία θα πρέπει να ενεργοποιείται σε τακτά χρονικά διαστήματα, αν θέλουμε να αποφύγουμε την υπερφόρτωση της βάσης δεδομένων, και μόνο από το διαχειριστή του συστήματος, αν θέλουμε να αποφύγουμε την καταστροφή χρήσιμων πληροφοριών που περιλαμβάνονται στη βάση δεδομένων.

Τα σημασιολογικά μεταδεδομένα που αφορούν κάποιο item έχουν συνήθως πολύ μικρότερο όγκο από τα ακατέργαστα δεδομένα του ίδιου του item. Εκτός αυτού, τα σημασιολογικά μεταδεδομένα αποθηκεύονται κατ' ανάγκην σε κάποιο μέσο που επιτρέπει και ανάγνωση και εγγραφή, αφού δημιουργούνται μετά την αποθήκευση των multimedia δεδομένων σε CD-ROMs. Μία νέα εκδοχή ενός multimedia item θα δημιουργείται πλέον αν αλλάξουν είτε τα ακατέργαστα δεδομένα του, είτε τα σημασιολογικά δεδομένα. Η τεχνική δέλτα είναι δυνατό να εφαρμόζεται και στις δύο κατηγορίες των δεδομένων ενός item ή μόνο στα ακατέργαστα δεδομένα του, οπότε τα σημασιολογικά δεδομένα θα κρατούνται στο σύνολό τους για κάθε εκδοχή. Η δεύτερη επιλογή δεν αναμένεται να δημιουργήσει προβλήματα χώρου αποθήκευσης, αφού, όπως είπαμε, ο σχετικός όγκος των σημασιολογικών δεδομένων είναι μικρός.

6.5. Επέκταση του μοντέλου δεδομένων για την τήρηση εκδοχών

Για την υποστήριξη των εκδοχών σε ένα object-oriented hypermedia περιβάλλον είναι σκόπιμο να ορίσουμε μια νέα κλάση, τη VersionedObject, η οποία θα περιέχει δεδομένα και μεθόδους που χρειάζονται για την τήρηση των εκδοχών, και να κάνουμε όλες τις κλάσεις που μας ενδιαφέρουν για την τήρηση εκδοχών παιδιά της VersionedObject, ώστε να κληρονομούν τα δεδομένα και τις μεθόδους της κλάσης αυτής. Φυσικά είναι αδύνατο να υλοποιηθούν όλες οι μέθοδοι που θα χρησιμοποιηθούν για την τήρηση εκδοχών στο επίπεδο της κλάσης VersionedObject, γιατί αρκετές από αυτές εξαρτώνται άμεσα από τα ακατέργαστα multimedia δεδομένα (π.χ. η μέθοδος υπολογισμού των διαφορών μεταξύ δύο εκδοχών

μπορεί να είναι εντελώς διαφορετική σε items τύπου text και items τύπου video). Τέτοιες μέθοδοι θα υλοποιηθούν στις κλάσεις που περιέχουν τα αντίστοιχα δεδομένα. Επίσης οι κλάσεις-παιδιά της VersionedObject μπορούν να επανακαθορίσουν τις μεθόδους που κληρονόμησαν, αν για κάποιο λόγο ο αρχικός ορισμός των μεθόδων αυτών δεν εξυπηρετεί τις ανάγκες τους.

Η κλάση VersionedObject αρκεί να έχει δύο μεταβλητές, τις PreviousVersion και NextVersion, που θα περιέχουν ανά πάσα στιγμή τους προσδιοριστές δύο αντικειμένων της VersionedObject. Έτσι, μέσω των μεταβλητών αυτών θα προσπελάζονται όλες οι εκδοχές ενός αντικειμένου. Η αρχική εκδοχή ενός αντικειμένου θα έχει μια ειδική τιμή για την PreviousVersion και η τελική εκδοχή του θα έχει μια ειδική τιμή για τη NextVersion. Η τήρηση εκδοχών χρειάζεται βέβαια και άλλες πληροφορίες όπως το συγγραφέα της κάθε εκδοχής και τη χρονική στιγμή κατά την οποία η εκδοχή δημιουργήθηκε, τόσο για την πληροφόρηση των χρηστών, όσο και για τη διαχείριση της βάσης δεδομένων (π.χ. για τη συλλογή απορριμμάτων). Οι πληροφορίες αυτές περιέχονται, στο μοντέλο που περιγράφουμε, στον ορισμό της κλάσης AuthoredObject, της οποίας η κλάση VersionedObject θα πρέπει να είναι παιδί.

Όσον αφορά τις μεθόδους που θα οριστούν στην κλάση VersionedObject, οι λειτουργίες που πρέπει να υποστηρίζονται είναι, όπως προαναφέρθηκε, τέσσερις: δημιουργία εκδοχών, ανάκτηση εκδοχών, διαγραφή εκδοχών και συλλογή απορριμμάτων. Από την προηγούμενη συζήτηση, είναι φανερό το ότι η συλλογή απορριμμάτων θα πρέπει να υλοποιηθεί ως δοσοληψία, αφενός γιατί θα χρησιμοποιεί μεθόδους από πολλές κλάσεις αντικειμένων, και αφετέρου γιατί δε θα πρέπει να διακόπτεται.

Όταν ζητάμε ανάκτηση ή διαγραφή εκδοχών, θα πρέπει να προσδιορίζουμε σε ποια ή σε ποιες εκδοχές αναφερόμαστε. Εφόσον όλες οι εκδοχές έχουν τον ίδιο προσδιοριστή, μια εκδοχή θα ορίζεται μοναδικά από το συνδυασμό του προσδιοριστή του αντικειμένου για το οποίο πρόκειται, του συγγραφέα της και της χρονικής στιγμής κατά την οποία δημιουργήθηκε. Αν θέλουμε πολλές εκδοχές ενός αντικειμένου μπορούμε να δώσουμε πολλές εναλλακτικές τιμές για το συγγραφέα, ή/και ένα χρονικό διάστημα για τη στιγμή της δημιουργίας. Έτσι, για ανάκτηση και διαγραφή εκδοχών θα έχουμε τις μεθόδους

GetVersion(Criteria,ObjectId)

DeleteVersion(Criteria,ObjectId)

που θα ανακτούν ή θα διαγράφουν πολλές, στη γενική περίπτωση, εκδοχές. Η παράμετρος Criteria θα είναι ένας συνδυασμός από συνθήκες για το όνομα του συγγραφέα ή/και τη χρονική στιγμή δημιουργίας των εκδοχών και μπορεί να έχει ειδικές τιμές για την τελευταία εκδοχή που έφτιαξε ένας συγκεκριμένος συγγραφέας ή την τελευταία εκδοχή γενικά. Στην ειδική περίπτωση που οι μέθοδοι αυτές χρησιμοποιούνται για items οποιουδήποτε τύπου, τα κριτήρια θα μπορούν επίσης να αφορούν την ύπαρξη κάποιου συγκεκριμένου συνδέσμου στη μεταβλητή AnchoredLinks του item ή/και την ύπαρξη κάποιου συνδυασμού ιδιοτήτων στη μεταβλητή Properties. Ειδικότερα στην περίπτωση της DeleteVersion, τα κριτήρια μπορούν να περιλαμβάνουν πληροφορίες για το πλήθος των εκδοχών ανά συγγραφέα που θέλουμε να κρατήσουμε.

Εφόσον, όπως είπαμε και αρχικά, υποθέτουμε ότι στο φυσικό επίπεδο της βάσης δεν αποθηκεύουμε όλες τις εκδοχές, αλλά μόνο την αρχική και ένα δέλτα προς τα εμπρός για κάθε νεώτερη, οι μέθοδοι αυτές θα κάνουν τα εξής, για κάθε εκδοχή στην οποία θα εφαρμόζονται:

(1) Η GetVersion θα ξεκινά από την εκδοχή αυτή και θα βρίσκει όλες τις προηγούμενες εκδοχές ως και την αρχική μέσω της μεταβλητής PreviousVersion. Στη συνέχεια, η GetVersion θα διασχίζει προς τα εμπρός την αλυσίδα των εκδοχών εφαρμόζοντας ένα-ένα τα δέλτα στην αρχική εκδοχή και, όταν εφαρμόζεται και το τελευταίο δέλτα, θα επιστρέφει το αποτέλεσμα, το οποίο θα είναι όλες οι εκδοχές που ικανοποιούν τα κριτήρια.

(2) Αν η DeleteVersion εφαρμόζεται σε μια ενδιάμεση εκδοχή, τότε θα βρίσκει την προηγούμενη και την επόμενη εκδοχή μέσω των μεταβλητών PreviousVersion και NextVersion, θα εφαρμόζει στην ενδιάμεση εκδοχή το δέλτα της επόμενης εκδοχής παράγοντας μια νέα επόμενη εκδοχή, θα σώζει τη νέα επόμενη εκδοχή, θα συνδέει την προηγούμενη εκδοχή με τη νέα επόμενη εκδοχή, και θα διαγράφει την ενδιάμεση εκδοχή φυσικά. Αν η DeleteVersion εφαρμόζεται στη λιγότερο πρόσφατη εκδοχή, τότε θα εφαρμόζει σε αυτήν το πρώτο δέλτα προς τα εμπρός παράγοντας τη νέα αρχική εκδοχή, θα σώζει τη νέα αρχική εκδοχή και θα διαγράφει την παλιά αρχική εκδοχή φυσικά. Αν, τέλος, η DeleteVersion εφαρμόζεται στην περισσότερο πρόσφατη εκδοχή, τότε αρκεί να απομακρυνθεί το δέλτα της εκδοχής αυτής.

Τέλος, θα πρέπει να υλοποιηθεί και η δημιουργία νέων εκδοχών. Εδώ θα πρέπει να παρατηρήσουμε ότι για να δημιουργηθεί μια νέα εκδοχή ενός αντικειμένου θα πρέπει (i) να υπολογιστεί το δέλτα από την

προηγούμενη εκδοχή στην καινούρια, και (ii) να αποθηκευτεί το δέλτα αυτό στη βάση. Υποτίθεται βέβαια, όπως αναφέρθηκε και στην αρχή, ότι το δέλτα ενός αντικειμένου είναι ένα αντικείμενο της ίδιας κλάσης. το δέλτα ενός item τύπου text, για παράδειγμα, είναι επίσης ένα item τύπου text. Κατά συνέπεια, η αποθήκευση του δέλτα θα γίνει σε κάθε περίπτωση με τη μέθοδο Save της αντίστοιχης κλάσης. Η μόνη μέθοδος, λοιπόν, που μένει να οριστεί εδώ είναι η

ComputeDelta(OldValues,NewValues)

η οποία θα παίρνει όλες τις παλιές και όλες τις καινούριες τιμές των μεταβλητών ενός αντικειμένου και θα υπολογίζει τα αντίστοιχα δέλτα. Στο επίπεδο της VersionedObject η ComputeDelta θα πρέπει να οριστεί εντελώς γενικά, αφού είναι βέβαιο το ότι κατά την υλοποίηση θα επανακαθοριστεί στις περισσότερες από τις κλάσεις-απογόνους.

Μετά την επέκταση της ιεραρχίας κλάσεων σύμφωνα με τα παραπάνω, το προτεινόμενο μοντέλο δεδομένων θα μπορεί να υποστηρίξει και την τήρηση εκδοχών.

6.6. Βιβλιογραφία

1. [AKS88] Robert M. Akscyn, Donald L. McCracken και Elise A. Yoder, "KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations", CACM, τεύχος 31/7, Ιούλιος 1988.
2. [COX86] Brad J. Cox, "Object Oriented Programming: An Evolutionary Approach", Addison-Wesley, 1986.
3. [GARG88] Pankaj K. Garg, "Abstraction Mechanisms in Hypertext", CACM, τεύχος 31/7, Ιούλιος 1988.
4. [HAL88] Frank G. Halasz, "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems", CACM, τεύχος 31/7, Ιούλιος 1988.
5. [HAN87] Robin Hanson, "Toward Hypertext Publishing. Issues and Choices in Database Design", Μάρτιος 1987.
6. [KER84] Brian W. Kernighan και Rob Pike, "The UNIX Programming Environment", Prentice Hall, 1984.

7. ΔΙΑΧΕΙΡΙΣΗ HYPERMEDIA ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΔΟΣΟΛΗΨΙΩΝ

Μια βάση δεδομένων είναι ένα σύνθετο σύστημα υλικού και λογισμικού, για το οποίο οι χρήστες δε θέλουν και δεν μπορούν να έχουν σφαιρικό έλεγχο. Έτσι είναι απαραίτητη η ύπαρξη ενός ή κάποιων ειδικών που αναλαμβάνουν να παρακολουθούν, να ελέγχουν και να βελτιώνουν τη λειτουργία της βάσης δεδομένων, με μια λέξη να διαχειρίζονται τη βάση. Επιπλέον, είναι απαραίτητο κάποιες επεξεργασίες που γίνονται στη βάση να μη διακόπτονται, ώστε να διατηρούνται τα αποθηκευμένα δεδομένα σε συνεπή κατάσταση. Έτσι, εισάγεται η έννοια των δοσοληψιών και μπαίνει το ζήτημα με ποιο τρόπο θα γίνεται η εξυπηρέτηση και η ακύρωσή τους.

Το κεφάλαιο αυτό ασχολείται ακριβώς με τα προβλήματα της διαχείρισης και της επεξεργασίας δοσοληψιών σε hypermedia βάσεις δεδομένων. Καθώς οι βάσεις hypermedia δε διαφοροποιούνται ριζικά στον τομέα αυτό από τις συμβατικές βάσεις, τα όσα αναφέρονται παρακάτω ισχύουν κατ' αρχήν για συμβατικές βάσεις δεδομένων. Σε όσα σημεία κάποια συγκεκριμένα χαρακτηριστικά των hypermedia βάσεων υπαγορεύουν διαφορετικές επιλογές, αυτό αναφέρεται ρητά.

7.1. Απαιτήσεις για τη διαχείριση hypermedia βάσεων δεδομένων

Οι ανάγκες για *διαχείριση* (administration) των βάσεων δεδομένων είναι ήδη γνωστές από τις συμβατικές βάσεις δεδομένων.

Μεταξύ των αρμοδιοτήτων του *διαχειριστή μιας βάσης δεδομένων* (database administrator, συντομ. DBA) περιλαμβάνονται και οι ακόλουθες ([KOR86]):

(1) ορισμός του *σχήματος της βάσης δεδομένων* (database schema)

(2) ορισμός των *δομών αποθήκευσης* (storage structures) και των *μεθόδων προσπέλασης* (access methods) για τα αποθηκευμένα δεδομένα

(3) αλλαγές στο σχήμα της βάσης δεδομένων, καθώς και στη φυσική οργάνωση των αποθηκευμένων δεδομένων (π.χ. αλλαγή της οργάνωσης από σειριακή σε B-δέντρο, δημιουργία *ευρετηρίων* (indices), κλπ.)˙ οι αλλαγές αυτές σε καμία περίπτωση δεν πρέπει να επηρεάζουν τον τρόπο με τον οποίο οι χρήστες βλέπουν τη βάση δεδομένων

(4) εκχώρηση προνομίων στους χρήστες και αφαίρεση των προνομίων αυτών˙ τα προνόμια αφορούν το ποια τμήματα της βάσης δεδομένων έχουν το δικαίωμα οι χρήστες να προσπελάσουν και με ποιον τρόπο (για ανάγνωση μόνο, για τροποποίηση, για διαγραφή, κλπ.)

(5) ορισμός *περιορισμών ακεραιότητας* (integrity constraints) που έχουν σκοπό να διατηρήσουν την *ορθότητα* (correctness), τη *συνέπεια* (consistency) και την *ακεραιότητα* (integrity) των αποθηκευμένων δεδομένων. Τέτοιοι περιορισμοί για παράδειγμα θα εξασφαλίζουν ότι όλα τα δεδομένα έχουν έγκυρες τιμές, ότι δεν υπάρχουν αναφορές σε ανύπαρκτα στοιχεία, κλπ.

Όλα αυτά τα καθήκοντα ισχύουν και για το διαχειριστή μιας hypermedia βάσης δεδομένων. Στα καθήκοντα αυτά θα πρέπει να προσθέσουμε

(6) τη δημιουργία μηχανισμών για *τήρηση αντιγράφων* (backup) και *επανόρθωση* (recovery) της βάσης δεδομένων, δηλαδή την περιοδική αποθήκευση σε μαγνητική ταινία αντιγράφων των περιεχομένων της βάσης που βρίσκονται σε μαγνητικούς δίσκους, και τη χρήση των αντιγράφων αυτών για την επανόρθωση της βάσης σε περίπτωση *βλάβης του υλικού* ή του *λογισμικού* (hardware ή software failure), και

(7) τη συλλογή απορριμμάτων.

Θα πρέπει να σημειώσουμε εδώ ότι η σωστή διαχείριση μιας βάσης δεδομένων είναι απαραίτητη όχι μόνο για τη διατήρηση της ακεραιότητάς της και την αποδοτική λειτουργία της, αλλά και για την ίδια την υλοποίηση των σκοπών για τους οποίους η βάση είναι σχεδιασμένη. Αν, για παράδειγμα, μια βάση δεδομένων έχει αναπτυχθεί για ένα εκπαιδευτικό περιβάλλον, τότε η εκχώρηση στους χρήστες προνομίων για προσπέλαση σε όλη τη βάση μπορεί να τους επιτρέψει τελικά να ασχολούνται με αντικείμενα άσχετα με την εργασία τους, ή να τους αποπροσανατολίσει πλήρως μέσα στο χώρο των συσχετισμένων δεδομένων. Ακόμα, αν η βάση καθυστερεί πολύ να αποκριθεί στις ερωτήσεις ή εντολές των χρηστών, οι

τελευταίοι θα φαίνονται, από ένα σημείο και μετά, απρόθυμοι να τη χρησιμοποιήσουν. Έτσι, η ορθή διαχείριση μιας βάσης δεδομένων πρέπει να εξασφαλίζει πρώτα απ' όλα ότι η βάση πραγματοποιεί τους σκοπούς, και μετά ότι τους πραγματοποιεί αποδοτικά.

Ο διαχειριστής μιας βάσης δεδομένων θα πρέπει να έχει στη διάθεσή του εργαλεία που να τον βοηθούν στην υλοποίηση των στόχων της εργασίας του. Ανάμεσα στα εργαλεία αυτά πρέπει να συμπεριλαμβάνονται ([DAT84]):

(1) εργαλεία για την αρχική δημιουργία της βάσης δεδομένων

(2) εργαλεία τα οποία θα κάνουν αναδιοργάνωση των δεδομένων σε φυσικό επίπεδο, προκειμένου να είναι πιο γρήγορη η προσπέλασή τους (π.χ. τοποθέτηση των δεδομένων μιας εγγραφής ή ενός αντικειμένου (προκειμένου για βάσεις προσανατολισμένες στις εγγραφές ή στα αντικείμενα, αντίστοιχα) σε συνεχόμενους τομείς του δίσκου)

(3) *μηχανισμούς τήρησης ημερολογίων* (journaling mechanisms), δηλαδή ρουτίνες που θα καταγράφουν τις λειτουργίες που γίνονται πάνω στη βάση δεδομένων, καθώς και την κατάσταση της βάσης πριν και μετά από κάθε λειτουργία

(4) *μηχανισμούς επανόρθωσης* της βάσης (revovery mechanisms), δηλαδή ρουτίνες για την επαναφορά της βάσης δεδομένων σε μια παλιότερη, σωστή κατάσταση, μετά από κάποια βλάβη υλικού ή λογισμικού

(5) εργαλεία για στατιστική ανάλυση των λειτουργιών που γίνονται πάνω στη βάση δεδομένων· τα αποτελέσματα μιας τέτοιας ανάλυσης θα χρησιμοποιηθούν για να εξαχθούν συμπεράσματα σχετικά με το πώς μπορεί να βελτιωθεί η απόδοση της βάσης δεδομένων σε σχέση με τις τυπικές επεξεργασίες.

Στον κατάλογο αυτό θα πρέπει να προσθέσουμε το συλλέκτη απορριμμάτων (garbage collector). Θα πρέπει να σημειωθεί ότι εδώ η συλλογή απορριμμάτων δεν έχει μόνο την έννοια που της δίναμε μιλώντας για τήρηση εκδοχών, αλλά σκοπός της είναι και η απελευθέρωση του χώρου που κατείχαν τα δεδομένα που έχουν διαγραφτεί. Αυτός είναι και ο λόγος που η συλλογή απορριμμάτων συνδυάζεται πολλές φορές με τη φυσική αναδιοργάνωση των δεδομένων, και χρησιμοποιείται ένας ενιαίος μηχανισμός και για τα δύο καθήκοντα.

Όλα τα εργαλεία που αναφέρθηκαν θα αποτελούν αναπόσπαστο τμήμα του συστήματος διαχείρισης της βάσης δεδομένων.

7.2. Επεξεργασία δοσοληψιών σε hupermedia βάσεις δεδομένων

Στο [KOR86] βρίσκουμε τον εξής ορισμό για τις δοσοληψίες σε σχεσιακά συστήματα βάσεων δεδομένων:

> "Μια δοσοληψία είναι ένα τμήμα προγράμματος του οποίου η εκτέλεση διατηρεί τη συνέπεια της βάσης δεδομένων. Αν, πριν εκτελεστεί μια δοσοληψία, η βάση δεδομένων βρισκόταν σε *συνεπή κατάσταση* (consistent state), τότε και μετά την εκτέλεση της δοσοληψίας η βάση δεδομένων θα βρίσκεται πάλι σε συνεπή κατάσταση. Για να διασφαλιστεί κάτι τέτοιο, θέλουμε η δοσοληψία να είναι *αδιαίρετη* (atomic), δηλαδή είτε να εκτελούνται όλες οι εντολές της, είτε να μην εκτελείται καμία εντολή της."

Παρατηρούμε ότι ο ορισμός αυτός δεν αναφέρεται σε στοιχεία υλοποίησης της βάσης δεδομένων (δηλαδή στο σχήμα της βάσης, στις λογικές και φυσικές οργανώσεις των δεδομένων, κλπ.), παρά μόνο στη συνέπειά της, η οποία είναι ένα εξωτερικό, ως προς τα προηγούμενα στοιχεία, χαρακτηριστικό. Ωστόσο, η έννοια των δοσοληψιών χρησιμοποιείται για να διατηρείται από το σύστημα διαχείρισης της βάσης δεδομένων η συνέπεια των δεδομένων που βρίσκονται μέσα στη βάση, και όχι των δεδομένων που μπορούν να αντλήσουν από τη βάση οι χρήστες. Το να εμφανίζονται τα τελευταία κατά τρόπο συνεπή στους χρήστες (να μην εκτυπώνονται, για παράδειγμα, τα αποτελέσματα μιας *ερώτησης* (query) της οποίας η εκτέλεση διακόπηκε πριν ολοκληρωθεί), είναι συνήθως ευθύνη του user interface. Αυτό είναι απόλυτα δεκτό στα κλασικά συστήματα βάσεων δεδομένων, αλλά, σε ένα μοντέλο δεδομένων στο οποίο υπάρχουν και σημασιολογικά στοιχεία, θα πρέπει να προστατεύεται από το σύστημα διαχείρισης της βάσης δεδομένων και η συνέπεια των δεδομένων που οι χρήστες εξάγουν από την βάση. Έτσι, θα πρέπει να

επεκτείνουμε τον πιο πάνω ορισμό των δοσοληψιών, ως εξής:

> "Δοσοληψία είναι ένα τμήμα προγράμματος το οποίο πρέπει να εκτελείται είτε ολόκληρο είτε καθόλου. Δοσοληψίες που εκτελούνται πάνω σε μια συνεπή κατάσταση μιας βάσης δεδομένων καταλήγουν σε μια επίσης συνεπή κατάσταση της βάσης δεδομένων, και τα αποτελέσματα που προκύπτουν από αυτές και παρουσιάζονται στους χρήστες είναι επίσης συνεπή."

Για την υποστήριξη των δοσοληψιών η γλώσσα χειρισμού δεδομένων (data manipulation language, συντομ. DML) θα πρέπει να διαθέτει εντολές ανάλογες με τις "Begin Transaction" (για να σημειωθεί η έναρξη μιας δοσοληψίας), "Commit" (για να περάσουν όσες αλλαγές έχουν γίνει στη βάση δεδομένων), "Abort Transaction" (για να τερματιστεί μια δοσοληψία χωρίς να περάσουν στη βάση δεδομένων οι αλλαγές που έχουν γίνει) και "End Transaction" (για να σημειωθεί το τέλος μιας δοσοληψίας). Παρόλα αυτά, οι δοσοληψίες δε θα είναι ορατές στους χρήστες· οι αιτήσεις των τελευταίων προς τη βάση δεδομένων πρέπει να διατυπώνονται σε κάποια γλώσσα υψηλού επιπέδου, και κατόπιν να μεταφράζονται στις αντίστοιχες δοσοληψίες. Οι χρήστες θα ειδοποιούνται μόνο στην περίπτωση που κάποια δοσοληψία αποτύχει.

Οι μέθοδοι που έχουμε ορίσει για κάθε κλάση αντικειμένων δεν αντιστοιχούν σε δοσοληψίες, παρά μόνο σε τμήματά τους. Στο μοντέλο που προτείνουμε, ουσιαστικά κάθε αίτηση που υποβάλλει ένας χρήστης στη βάση δεδομένων αντιστοιχεί σε μία δοσοληψία, και χρησιμοποιηθεί ένα σύνολο από μεθόδους διαφορετικών, γενικά, κλάσεων, για να διεκπεραιωθεί διατηρώντας τη βάση σε καλή κατάσταση. Τα δύο βασικά σημεία που πρέπει να εξασφαλίσουμε είναι:

(1) το ότι όπου χρησιμοποιείται ο προσδιοριστής ενός αντικειμένου, το αντικείμενο αυτό υπάρχει, και

(2) το ότι όπου δίνεται τιμή για μια μεταβλητή η οποία πρέπει να έχει μοναδικές τιμές, η νέα τιμή δεν υπάρχει ήδη σε κάποιο άλλο στιγμιότυπο της αντίστοιχης κλάσης.

Για να εξασφαλίσουμε αυτά τα δύο σημεία, καθώς και μερικά ακόμα, έχουμε τις παρακάτω δοσοληψίες (αναφέρονται ομαδοποιημένες για απλότητα):

(1) Δημιουργία ενός αντικειμένου. Η λειτουργία αυτή είναι απαραίτητο να υλοποιηθεί ως δοσοληψία ειδικά όταν δημιουργούνται αντικείμενα που αναφέρονται σε άλλα αντικείμενα. Έτσι:

(1α) Η δημιουργία ενός item με ContentType διαφορετικό από dummy, απαιτεί έλεγχο για την ύπαρξη του αρχείου στο οποίο βρίσκονται τα περιεχόμενα που ο χρήστης ορίζει.

(1β) Η δημιουργία ενός συνδέσμου μεταξύ δύο items απαιτεί έλεγχο για την ύπαρξη των items, τη δημιουργία του link και τη δημιουργία καινούριων εκδοχών των items οι οποίες θα περιλαμβάνουν στη μεταβλητή AnchoredLinks τον προσδιοριστή του νέου συνδέσμου.

(1γ) Η δημιουργία μιας όψης απαιτεί ελέγχους για την ύπαρξη των items των οποίων οι προσδιοριστές περιλαμβάνονται στις μεταβλητές ExcItems και IncItems.

(1δ) Η δημιουργία ενός συγγραφέα απαιτεί έλεγχο για την ύπαρξη και άλλου συγγραφέα με το ίδιο όνομα.

(2) Τροποποίηση ενός αντικειμένου. Επιπλέον έλεγχοι απαιτούνται στις παρακάτω περιπτώσεις:

(2α) Η τροποποίηση της μεταβλητής ContentType ενός item απαιτεί έλεγχο για το αν η αρχική τιμή της είναι dummy, και αν υπάρχει το αρχείο που δήλωσε ο χρήστης για το περιεχόμενο του item.

(2β) Στην ειδική περίπτωση που ο χρήστης μπορεί να τροποποιήσει το περιεχόμενο ενός item, πρέπει να εξασφαλίζεται ότι οι άγκυρες των συνδέσμων που ξεκινάνε από το περιεχόμενο του item ή καταλήγουν σε αυτό παραμένουν έγκυρες.

(2γ) Η τροποποίηση μιας άγκυρας ενός συνδέσμου θα δημιουργεί καινούρια εκδοχή του αντίστοιχου item.

(2δ) Οποιαδήποτε πρόσθεση στις μεταβλητές IncItems και ExcItems μιας όψης απαιτεί ελέγχους για την ύπαρξη των items που προστίθενται.

(2ε) Τροποποίηση του ονόματος κάποιου συγγραφέα απαιτεί έλεγχο για το αν υπάρχει άλλος συγγραφέας με το ίδιο όνομα, καθώς και την ενημέρωση όλων των στιγμιοτύπων της κλάσης VersionedObject (και των κλάσεων-παιδιών της) που αναφέρονται στο συγγραφέα αυτόν.

Οι υπόλοιπες περιπτώσεις τροποποιήσεων δεν απαιτούν επιπλέον ελέγχους.

(3) Διαγραφή ενός αντικειμένου. Έλεγχοι απαιτούνται στις παρακάτω περιπτώσεις:

(3α) Διαγραφή της τελευταίας εκδοχής κάποιου item (δηλαδή αίτηση για απομάκρυνση της τελευταίας εκδοχής ενός item που υπάρχει στη βάση) θα πρέπει να συμπεριλαμβάνει και την απομάκρυνση των συνδέσμων των οποίων οι προσδιοριστές περιλαμβάνονται στη μεταβλητή

AnchoredLinks του item αυτού, καθώς επίσης και την ενημέρωση των items που βρίσκονται στο άλλο άκρο των συνδέσμων αυτών. Οι τελευταίες ενημερώσεις **δε** θα δημιουργούν καινούριες εκδοχές των items που ενημερώνονται. Θα πρέπει επίσης να ελέγχονται οι ορισμοί των όψεων για την περίπτωση να υπάρχει ο προσδιοριστής του item αυτού στις μεταβλητές IncItems και ExcItems. Ο έλεγχος αυτός στις όψεις μπορεί, εναλλακτικά, να μη γίνεται κατά τη διαγραφή ενός item, αλλά όταν πρόκειται να χρησιμοποιηθεί ο ορισμός τους. Θα πρέπει εδώ να προσέξουμε το ότι αν οι προσδιοριστές των items που έχουν διαγραφτεί χρησιμοποιούνται ξανά για οικονομία, τότε πρέπει να υπάρχει ένας μηχανισμός που να εξασφαλίζει ότι ένας τέτοιος προσδιοριστής δε θα χρησιμοποιηθεί πριν ενημερωθούν οι ορισμοί όλων των όψεων που τον περιέχουν στις μεταβλητές IncItems και ExcItems.

(3β) Η αίτηση για φυσική διαγραφή ενός συνδέσμου οδηγεί στην ενημέρωση των items που συνδέονται με το σύνδεσμο αυτό, χωρίς να δημιουργούνται καινούριες εκδοχές τους.

(3γ) Διαγραφή κάποιου συγγραφέα δε θα καταλήγει στη διαγραφή των στιγμιοτύπων της κλάσης VersionedObject (και των κλάσεων-παιδιών της) που αναφέρονται στο συγγραφέα αυτόν.

Η διαγραφή μιας όψης δεν απαιτεί επιπλέον ελέγχους.

(4) Ανάκτηση ενός αντικειμένου.

(4α) Η ανάκτηση ενός item θα ακολουθείται από την ανάκτηση των συνδέσμων που συνδέονται με αυτό.

(4β) Κατά την ανάκτηση ενός συνδέσμου θα ελέγχεται αν υπάρχουν τα items τα οποία συνδέει (και, ίσως, αν υπάρχουν ειδικά κάποιες εκδοχές των items που αναφέρονται στον προσδιοριστή του συνδέσμου αυτού).

(4γ) Η ανάκτηση του ορισμού κάποιας όψης θα περιλαμβάνει ελέγχους για την ύπαρξη των items που εμφανίζονται στις μεταβλητές IncItems και ExcItems, αν οι έλεγχοι αυτοί δε γίνονται κατά τη διαγραφή των items.

Η ανάκτηση ενός συγγραφέα δεν απαιτεί επιπλέον ελέγχους. Έτσι, και η ανάκτηση αντικειμένων θα πρέπει να θεωρηθεί δοσοληψία, γιατί στην αντίθετη περίπτωση είναι δυνατό να παρουσιάζονται ελλιπείς πληροφορίες στους χρήστες ή κάποιες από τις ενημερώσεις (που ίσως αποφασίζονται μετά τους ελέγχους) να γίνονται, και κάποιες όχι.

7.3. Ακύρωση δοσοληψιών σε hypermedia βάσεις δεδομένων

Κατά τη λειτουργία της βάσης δεδομένων, υπάρχει η περίπτωση να διατυπώσει κάποιος χρήστης από αβλεψία ή άγνοια μια αίτηση που θα τροποποιήσει τη βάση διαφορετικά από ότι ήθελε, ή να μετανιώσει για μια αλλαγή που μόλις έκανε. Στην περίπτωση αυτή, θα πρέπει να έχει ο χρήστης τη δυνατότητα να αντιστρέψει την προηγούμενή του ενέργεια. Φυσικά, αυτό μπορεί πάντα να γίνει χρησιμοποιώντας τα αντίγραφα της βάσης και τα ημερολόγια που θα διατηρεί ο DBA, αλλά είναι προφανές ότι δεν μπορεί να καλείται ο διαχειριστής της βάσης κάθε φορά που ένας χρήστης θέλει να ακυρώσει κάποια δοσοληψία. Έτσι, χρειάζεται κάποιος πιο άμεσος τρόπος για να πραγματοποιείται η ακύρωση των ενεργειών των χρηστών, ή, πιο συγκεκριμένα, χρειάζεται να διαθέτουν οι χρήστες κάποια επιλογή *ακύρωσης* (undo).

Οι αιτήσεις των χρηστών προς τη βάση δεδομένων αφορούν τη δημιουργία, διαγραφή, τροποποίηση και ανάκτηση αντικειμένων. Δεν έχει νόημα να ακυρώνεται η ανάκτηση ενός αντικειμένου, και έτσι υπάρχουν τρεις περιπτώσεις ακύρωσης:

(1) να ακυρωθεί η δημιουργία ενός αντικειμένου, με τη διαγραφή του

(2) να ακυρωθεί η διαγραφή ενός αντικειμένου, με τη δημιουργία του

(3) να ακυρωθεί η τροποποίηση ενός αντικειμένου, με την ακριβώς αντίθετη τροποποίησή του.

Αν η δημιουργία, διαγραφή ή τροποποίηση ενός αντικειμένου οδήγησε σε δημιουργία, διαγραφή ή τροποποίηση και άλλων αντικειμένων και στη συνέχεια ακυρώθηκε, εννοείται ότι θα πρέπει να ακυρωθεί και η δημιουργία, διαγραφή ή τροποποίηση και αυτών των αντικειμένων.

Είναι γεγονός, βέβαια, ότι όσο μεγαλύτερο είναι το πλήθος των αλλαγών στη βάση δεδομένων που κάνει μία δοσοληψία, τόσο περισσότερες είναι οι πληροφορίες που πρέπει να φυλάσσονται προκειμένου να μπορεί να επανέλθει η βάση σε συνεπή κατάσταση αν η δοσοληψία αυτή αποτύχει ή πρέπει να ακυρωθεί. Δεδομένου, μάλιστα, ότι μιλάμε για ένα περιβάλλον hypermedia, ο όγκος των απαραίτητων πληροφοριών αυξάνει δραματικά. Γι' αυτούς τους λόγους καταλήγουμε στο συμπέρασμα ότι το μέγεθος και ο χρόνος εκτέλεσης κάθε δοσοληψίας θα πρέπει να κρατηθεί σε χαμηλά επίπεδα, ώστε να είναι μικρά και το μέγεθος

και η χρονική διάρκεια της διαδικασίας που θα ακυρώνει τη δοσοληψία αυτή (η οποία θα πρέπει επίσης να υλοποιηθεί ως δοσοληψία) (μια ειδική τεχνική για τη διάσπαση μιας δοσοληψίας σε μια ιεραρχία από υποδοσοληψίες, ώστε να μειώνεται δραστικά το μέγεθος των δοσοληψιών-φύλλων, περιγράφεται στο [NEU88]).

Όσον αφορά το πώς θα φυλάσσονται οι πληροφορίες που είναι απαραίτητες για να μπορεί να ακυρωθεί μια δοσοληψία, μπορούμε να χρησιμοποιήσουμε την τεχνική της κρυφής σελιδοποίησης, που περιγράφεται στην επόμενη παράγραφο για την επαναφορά των δοσοληψιών (άλλωστε, και η ακύρωση μιας δοσοληψίας είναι μια ειδική περίπτωση επαναφοράς).

Έτσι, έχοντας ακόμα υπόψη το ότι η βάση δεδομένων τηρεί, για κάποια αντικείμενα, εκδοχές, με αποτέλεσμα τα δεδομένα που υπήρχαν πριν τις αλλαγές των αντικειμένων αυτών να βρίσκονται ακόμα στη βάση δεδομένων σε παλιότερες εκδοχές τους, και το ότι η διαδικασία των δοσοληψιών θα πρέπει να είναι διαφανής στους χρήστες, καταλήγουμε στον ακόλουθο μηχανισμό:

> Όλες οι ενέργειες που είναι απαραίτητες για να εξυπηρετηθεί μια αίτηση προς τη βάση δεδομένων, μπαίνουν σε μια δοσοληψία. Στο τέλος της δοσοληψίας οι αλλαγές δεν περνάνε στη βάση δεδομένων ως μόνιμες (ο τερματισμός της δοσοληψίας, δηλαδή, δεν ακολουθείται από ένα αυτόματο commit). Αν τώρα ο χρήστης διατυπώσει μια αίτηση ακύρωσης της προηγούμενης δοσοληψίας, η αίτηση αυτή μεταφράζεται σε μια εντολή "Abort Transaction", με την οποία η βάση δεδομένων γυρνά στην κατάσταση που είχε πριν διατυπωθεί η αρχική αίτηση. Αν ο χρήστης διατυπώσει οποιαδήποτε άλλη αίτηση προς τη βάση δεδομένων εκτός από αίτηση για ακύρωση, τότε εκτελείται αυτόματα πριν από την εξυπηρέτηση της νέας αίτησης ένα η *μονιμοποίηση* της προηγούμενης δοσοληψίας (transaction commitment) και με αυτόν τον τρόπο όποιες αλλαγές έχουν γίνει περνάνε στη βάση δεδομένων ως μόνιμες.

7.4. Προβλήματα επανόρθωσης σε hypermedia βάσεις δεδομένων

Στην προηγούμενη παράγραφο αναφέρθηκε ότι μια δοσοληψία είναι ένα αδιαίρετο κομμάτι προγράμματος που είτε θα εκτελεστεί ολόκληρο είτε καθόλου, και ότι ο μηχανισμός των δοσοληψιών είναι επαρκής για τη διατήρηση της συνέπειας των δεδομένων. Αυτό είναι αληθές σε λογικό επίπεδο, αλλά σε φυσικό επίπεδο μία δοσοληψία απεικονίζεται σε πολλές προσπελάσεις στη βάση δεδομένων (αναγνώσεις ή/και εγγραφές) και είναι πιθανό κάποιες από αυτές να εκτελεστούν και κάποιες άλλες όχι. Αυτό μπορεί να οφείλεται στους εξής λόγους :

(1) Παρουσιάζεται πτώση του συστήματος λόγω διακοπής ρεύματος, σφάλματος του λειτουργικού συστήματος, κλπ.

(2) Παρουσιάζεται σφάλμα στους μαγνητικούς δίσκους όπου φυλάσσονται τα σημασιολογικά δεδομένα και ίσως τα δεδομένα που αφορούν τις ενημερώσεις που έγιναν στα multimedia δεδομένα. Σφάλματα στους οπτικούς δίσκους όπου φυλάσσονται τα multimedia δεδομένα θεωρούνται απίθανα, αλλά και αν συμβούν μπορούν να αντιμετωπισθούν μόνο με συνολική αντικατάσταση του οπτικού δίσκου, αφού δεν μπορούμε να γράψουμε πάνω σε αυτόν που υπέστη τη ζημιά, τουλάχιστον με την ως τώρα τεχνολογία. Κατά συνέπεια, σφάλματα αυτού του τύπου δε θα ληφθούν υπόψη.

(3) Παρουσιάζεται λάθος (bug) σε κάποιο πρόγραμμα, ή κάποιο σφάλμα στα δεδομένα που έδωσε ο χρήστης, με αποτέλεσμα να μην μπορεί να συνεχιστεί η εκτέλεση της δοσοληψίας.

(4) Η διεργασία που ανέλαβε να εξυπηρετήσει τη δοσοληψία εμπλέκεται σε *αδιέξοδο* (deadlock) (πράγμα πιθανό σε περιβάλλον πολυπρογραμματισμού) και αναστέλλεται.

Σε όλες αυτές τις περιπτώσεις είναι πολύ πιθανό να απομείνει η βάση δεδομένων σε *ασυνεπή κατάσταση* (inconsistent state). Θα πρέπει, λοιπόν, να υπάρχουν μηχανισμοί που να την επαναφέρουν σε συνεπή κατάσταση με τη μικρότερη δυνατή απώλεια ενημερώσεων και με εξασφαλισμένη τη διαθεσιμότητα των δεδομένων τα οποία δεν έχουν υποστεί ζημιά. Η διαδικασία αυτή ονομάζεται *επανόρθωση* (recovery).

Οι περιπτώσεις (3) και (4) αντιμετωπίζονται σχετικά εύκολα γιατί μπορούμε να εξετάσουμε όλη την πορεία του προγράμματος ή των προγραμμάτων που προκάλεσαν το πρόβλημα μέσω των *ημερολογίων* (journals) που κρατάει το σύστημα διαχείρισης της βάσης δεδομένων, και να κάνουμε *επαναφορά* (rollback) των δοσοληψιών που έμειναν ημιτελείς, ώστε να αναιρεθούν οι αλλαγές που έγιναν στη βάση δεδομένων από τις δοσοληψίες αυτές. Μια διαφορετική τεχνική που μπορεί να χρησιμοποιηθεί για την αντιμετώπιση των περιπτώσεων αυτών είναι να μη γίνονται οι αλλαγές απευθείας στα δεδομένα της βάσης,

αλλά σε αντίγραφά τους. Αν οι δοσοληψίες ζητήσουν να περάσουν οι αλλαγές στη βάση ή τερματίσουν επιτυχώς (με εντολές "Commit" ή "End Transaction" αντίστοιχα), τότε τα αντίγραφα γράφονται στους δίσκους και έτσι οι αλλαγές γίνονται μόνιμες. Αν όχι, τότε τα αντίγραφα αγνοούνται. Η τεχνική αυτή ονομάζεται *κρυφή σελιδοποίηση* (shadow paging) και χρησιμοποιείται σε αρκετά εμπορικά συστήματα διαχείρισης βάσεων δεδομένων. Σε περιβάλλον multimedia, βέβαια, τα δεδομένα της βάσης θα έχουν αρκετά μεγάλο όγκο και έτσι δεν είναι πρακτικό να κρατάμε αντίγραφά τους. Γι' αυτό το λόγο, η κρυφή σελιδοποίηση θα πρέπει να βελτιωθεί έτσι ώστε να κρατούνται αντίγραφα μόνο για τις τιμές που αλλάζουν, ή να κρατούνται μόνο οι μεταβολές των τιμών που αλλάζουν, αν αυτό είναι δυνατό.

Στην περίπτωση (1) είναι επίσης δυνατό να επαναφέρουμε τις δοσοληψίες, υπό την προϋπόθεση όμως ότι οι αλλαγές στα ημερολόγια γράφονται απευθείας στο δίσκο, δε χρησιμοποιείται δηλαδή ενδιάμεση μνήμη (buffer) για τις ενημερώσεις των αρχείων αυτών.

Η περίπτωση (2) είναι αυτή που θα προκαλέσει τα περισσότερα προβλήματα, αφού ίσως χαθούν δεδομένα και από κάποια ημερολόγια. Στην περίπτωση αυτή αρχικά θα πρέπει να εξασφαλίσουμε ότι είναι διαθέσιμα τα δεδομένα που δεν έχουν υποστεί ζημιά. Αν χρησιμοποιούμε δέλτα προς τα εμπρός, τότε απώλεια οποιασδήποτε εκδοχής συνεπάγεται απώλεια και όλων των νεώτερων εκδοχών, και στην ακραία περίπτωση που θα χαθεί η πρώτη εκδοχή χάνεται ολόκληρο το αντικείμενο. Αν χρησιμοποιούμε δέλτα προς τα πίσω, τότε απώλεια οποιασδήποτε εκδοχής συνεπάγεται απώλεια και όλων των παλιότερων εκδοχών, και στην ακραία περίπτωση που θα χαθεί η τελευταία εκδοχή χάνεται ολόκληρο το αντικείμενο. (Εδώ, εφόσον συνήθως χρειαζόμαστε περισσότερο τις πιο πρόσφατες εκδοχές, φαίνεται ακόμα ένα πλεονέκτημα των δέλτα προς τα πίσω). Αφού βρούμε τα δεδομένα που είναι διαθέσιμα, χρησιμοποιούμε το πιο πρόσφατο αντίγραφο της βάσης για να ανακτήσουμε τα αντικείμενα και τις εκδοχές αντικειμένων που έχουν καταστραφεί. Αν τα ημερολόγια δεν έχουν καταστραφεί τότε μπορούμε να επαναλάβουμε (redo) τις δοσοληψίες εκείνες που έγιναν μετά το πιο πρόσφατο αντίγραφο και που αφορούν τα αντικείμενα που υπέστησαν ζημιές, ώστε τελικά να χάσουμε όσο το δυνατό λιγότερες ενημερώσεις. Σε κάθε περίπτωση όμως, μετά από οποιαδήποτε βλάβη υλικού ή λογισμικού, συνίσταται ένας *έλεγχος συνέπειας* (consistency check) στη βάση δεδομένων, έτσι ώστε να εντοπιστούν και να διορθωθούν τυχόν ασυνέπειες που έχουν προκύψει.

7.5. Προβλήματα ελέγχου σύγχρονης προσπέλασης σε hypermedia βάσεις δεδομένων

Εφόσον μια hypermedia βάση δεδομένων θα χρησιμοποιηθεί σε περιβάλλον πολυπρογραμματισμού, θα πρέπει να υπάρχουν μηχανισμοί οι οποίοι θα αναλάβουν να διαμοιράσουν τους πόρους της βάσης ανάμεσα σε *ανταγωνιστικές διεργασίες* (competing processes), έτσι ώστε να μην προκύπτουν ασυνέπειες και να μειώνεται η πιθανότητα αδιεξόδων, αλλά και να μην περιορίζεται σημαντικά η *διαθεσιμότητα των δεδομένων* (data availability).

Θα πρέπει να σημειωθεί εδώ, ότι ο έλεγχος της σύγχρονης προσπέλασης είναι ένα ανοιχτό πρόβλημα για τα περισσότερα εμπορικά συστήματα hypermedia. Το Notecards, για παράδειγμα, είχε σχεδιαστεί αρχικά για *περιβάλλον ενός χρήστη* (single-user environment). Άλλα προϊόντα, όπως τα KMS, Intermedia και Neptune, λειτουργούν σε *περιβάλλον πολλών χρηστών* (multi-user environment) αλλά δεν υποστηρίζουν ιδιαίτερα εξεζητημένους μηχανισμούς για τον έλεγχο της σύγχρονης προσπέλασης (για παράδειγμα, κανένα σύστημα δεν παρέχει ειδικό μηχανισμό για να ειδοποιεί ένας χρήστης τους υπόλοιπους όταν προτίθεται να τροποποιήσει κάποιες πληροφορίες. αυτό μπορεί να γίνει μόνο έμμεσα, με τα γενικής χρήσης εργαλεία που παρέχει το user interface, ή έξω από το σύστημα ([HAL88])). Στα επόμενα, θα αγνοήσουμε αυτές τις απαιτήσεις για *υποστήριξη συνεργασίας* (collaboration support) και θα περιοριστούμε στις τεχνικές για *διαμοιρασμό δεδομένων* (data sharing).

Στις συμβατικές βάσεις δεδομένων οι μηχανισμοί που χρησιμοποιούνται για διαμοιρασμό δεδομένων βασίζονται στα *διαμοιραζόμενα κλειδώματα* (shared locks ή S-locks), και στα *αποκλειστικά κλειδώματα* (exclusive locks ή X-locks). Ένα αντικείμενο στο οποίο έχει γίνει διαμοιραζόμενο κλείδωμα από μια δοσοληψία είναι δυνατό να διαβαστεί από άλλες δοσοληψίες, αλλά όχι και να τροποποιηθεί από αυτές, ενώ αν το κλείδωμα είναι αποκλειστικό, δεν μπορεί να προσπελαστεί από τις άλλες δοσοληψίες, ούτε για διάβασμα ούτε για τροποποίηση. Μια δοσοληψία χρησιμοποιεί διαμοιράσιμα κλειδώματα για τα αντικείμενα που θέλει μόνο να διαβάσει και αποκλειστικά κλειδώματα για τα αντικείμενα που θέλει να τροποποιήσει. Ανάλογα με το μηχανισμό που χρησιμοποιείται, η δέσμευση και η απελευθέρωση των

κλειδωμάτων μπορεί να γίνεται σε διάφορες φάσεις μιας δοσοληψίας: ο πιο συντηρητικός μηχανισμός είναι το *κλείδωμα σε μία φάση* (one phase locking), που απαιτεί όλα τα κλειδώματα να δεσμεύονται στην αρχή και να απελευθερώνονται στο τέλος, ενώ ένα πιο ευέλικτο σχήμα είναι το *κλείδωμα σε δύο φάσεις* (two phase locking), όπου μια δοσοληψία σε μια πρώτη φάση ζητά κλειδώματα, αλλά δεν μπορεί να απελευθερώσει κανένα από όσα έχει πάρει, ενώ στην επόμενη φάση μπορεί να απελευθερώσει κλειδώματα αλλά δεν μπορεί να ζητήσει καινούρια.

Είναι, ωστόσο, φανερό, ότι ένας τέτοιος μηχανισμός δεν μπορεί να εφαρμοστεί σε περιβάλλοντα όπως αυτό για το οποίο συζητάμε, γιατί μειώνει δραματικά τη διαθεσιμότητα των δεδομένων. Πράγματι, σε μια συμβατική βάση δεδομένων ένα αποκλειστικό κλείδωμα θα διαρκέσει για μερικά κλάσματα του δευτερολέπτου ή για κάποια δευτερόλεπτα, όσο χρειάζεται δηλαδή μια δοσοληψία για να διαβάσει δεδομένα, να κάνει όποια επεξεργασία απαιτείται και να γράψει τα καινούρια δεδομένα. Αντίθετα, σε μια βάση δεδομένων hypermedia, μια σύνοδος τροποποιήσεων σε ένα αντικείμενο μπορεί να διαρκέσει από μερικά λεπτά μέχρι αρκετές ώρες και, αν υιοθετήσουμε το μηχανισμό των διαμοιραζόμενων και αποκλειστικών κλειδωμάτων, το αντικείμενο δε θα είναι διαθέσιμο στις άλλες δοσοληψίες κατά το χρονικό διάστημα αυτό. Για το λόγο αυτό, στα hypermedia συστήματα χρησιμοποιούνται γενικά *μαλακά κλειδώματα* (soft lockings), δηλαδή κλειδώματα που επιτρέπουν όσο το δυνατόν μεγαλύτερο ταυτοχρονισμό στις προσπελάσεις, αυξάνοντας βέβαια τον κίνδυνο να έχουμε *συγκρούσεις ενημερώσεων* (update conflicts) ([HAL88]).

Στην κατηγορία των μαλακών κλειδωμάτων θα βρούμε το μηχανισμό κλειδώματος του KMS ([AKS88]) που ονομάζεται *αισιόδοξος έλεγχος σύγχρονης προσπέλασης* (optimistic concurrency control). Σύμφωνα με το μηχανισμό αυτό είναι δυνατό κάποιο αντικείμενο (*πλαίσιο* (frame) στην ορολογία του KMS) να τροποποιείται ταυτόχρονα από δύο ή περισσότερους χρήστες, αλλά μόνο οι αλλαγές του ενός χρήστη θα σωθούν στο αρχικό αντικείμενο, ενώ αυτές των άλλων χρηστών θα σωθούν σε κάποια προσωρινά αντικείμενα, προκειμένου οι χρήστες αυτοί να μπορούν να απεικονίσουν στην καινούρια μορφή του αντικειμένου τις αλλαγές που έκαναν. Μπορεί επίσης κάποιος χρήστης να "κλειδώσει" ένα αντικείμενο με το να σημειώσει πάνω σ' αυτό ότι επιθυμεί να το τροποποιήσει, να σώσει το αντικείμενο με την προειδοποίηση αυτή, και να συνεχίσει κάνοντας όποιες τροποποιήσεις αυτός θέλει. Υποτίθεται ότι ένας άλλος χρήστης που θα ανακτήσει αργότερα το αντικείμενο και θα δει την προειδοποίηση δε θα αρχίσει να το τροποποιεί κι αυτός, και ότι ο χρήστης που έβαλε στο αντικείμενο την προειδοποίηση, θα τη σβήσει όταν κάνει όσες αλλαγές ήθελε.

Ο μηχανισμός αυτός όμως δε συνίσταται σε καμία περίπτωση σε ένα περιβάλλον που τηρεί εκδοχές, αφού οι αλλαγές καταλήγουν στην δημιουργία καινούριων εκδοχών τις οποίες ο χρήστης θα πρέπει αργότερα να απομακρύνει από την βάση. Εκτός αυτού, ο προηγούμενος μηχανισμός δεν είναι απόλυτα ασφαλής, μια και είναι δυνατό κάποιος χρήστης να ξεχάσει να κάνει το ανεπίσημο κλείδωμα, ή κάποιος άλλος χρήστης να το αγνοήσει, ή ακόμη ο χρήστης που έκανε το κλείδωμα να ξεχάσει να το σβήσει. Είναι σαφές ότι η ευθύνη για το διαμοιρασμό των αντικειμένων θα πρέπει να περνάει στο σύστημα διαχείρισης της βάσης δεδομένων.

Μια λύση σ' αυτό το πρόβλημα είναι το να υπάρχει ένα *κλειδί εγγραφής* (write key) για κάθε αντικείμενο της βάσης δεδομένων και, σε κάθε δοσοληψία που ζητάει ανάγνωση ή τροποποίηση κάποιων αντικειμένων, να δίνονται αντίγραφα των αντικειμένων αυτών. Μια δοσοληψία θα ζητάει το κλειδί εγγραφής ενός αντικειμένου όταν πρόκειται να τροποποιήσει το αντικείμενο αυτό, και το κλειδί θα απελευθερώνεται στο τέλος της δοσοληψίας ή όταν η δοσοληψία εγκαταλειφθεί (με μια εντολή "Abort Transaction"). Όσο μια δοσοληψία έχει το κλειδί εγγραφής ενός αντικειμένου, οι άλλες δοσοληψίες μπορούν να διαβάσουν το αντικείμενο (να πάρουν δηλαδή ένα αντίγραφό του), αλλά όχι να το τροποποιήσουν. Για να τροποποιήσει μια δοσοληψία ένα αντικείμενο, θα πρέπει να βρει ελεύθερο το κλειδί εγγραφής του αντικειμένου αυτού και να το δεσμεύσει.

Στο επίπεδο των χρηστών, η απόκτηση του κλειδιού εγγραφής ενός αντικειμένου θα γίνεται μόνο μετά από σαφή αίτηση τροποποίησης προς το σύστημα διαχείρισης της βάσης δεδομένων, αλλά η ευθύνη αυτή μπορεί να μεταφερθεί στο user interface, το οποίο θα κάνει μια τέτοια αίτηση όταν, για παράδειγμα, ένας χρήστης καλεί κάποιον editor. Μια τέτοια αίτηση θα απορρίπτεται αν κάποια άλλη δοσοληψία έχει εκείνη τη στιγμή υπό την κατοχή της το κλειδί εγγραφής. Αν μια δοσοληψία (π.χ. η δημιουργία ενός συνδέσμου) απαιτεί την τροποποίηση περισσοτέρων από ένα αντικειμένων, και διαπιστωθεί ότι κάποια από αυτά δεν είναι διαθέσιμα, τότε θα πρέπει να εγκαταλείπεται, αφού δεν μπορούμε να προβλέψουμε τι αποτελέσματα θα έχει η τροποποίηση αντικειμένων των οποίων τα κλειδιά εγγραφής είναι δεσμευμένα από άλλες

δοσοληψίες, και ούτε μπορούμε να αφήσουμε τη δοσοληψία σε αναμονή, αφού έτσι θα μειωνόταν η διαθεσιμότητα των δεδομένων της βάσης και θα μπορούσαν να δημιουργηθούν αδιέξοδα.

Η χρήση των κλειδιών εγγραφής μπορεί να εκλεπτυνθεί αν έχουμε ένα κλειδί όχι για κάθε αντικείμενο αλλά για κάθε μεταβλητή κάθε αντικειμένου. Υπάρχουν ακόμα και μικτές περιπτώσεις, όπου για κάποιες κλάσεις έχουμε ένα κλειδί ανά αντικείμενο, και για κάποιες άλλες κλάσεις έχουμε ένα κλειδί ανά μεταβλητή αντικειμένου.

Όπως κι αν έχουν τα πράγματα, το σχήμα των κλειδιών εγγραφής απαιτεί να υπάρχουν *διαμοιράσιμα* (shareable) και αποκλειστικά κλειδώματα στο επίπεδο των λειτουργιών ανάγνωσης και εγγραφής αντικειμένων από τους και στους δίσκους· υπονοείται, δηλαδή, πως όταν μια δοσοληψία διαβάζει από το δίσκο ένα αντικείμενο δεν μπορεί κάποια άλλη να το τροποποιήσει, ενώ όταν κάποια δοσοληψία γράφει στο δίσκο ένα αντικείμενο, καμία προσπέλαση από άλλες δοσοληψίες δεν επιτρέπεται πάνω στο αντικείμενο αυτό. Το σχήμα αυτό εγγυάται ότι είναι αδύνατο δύο χρήστες να τροποποιούν το ίδιο αντικείμενο ταυτόχρονα, ενώ παράλληλα εξασφαλίζει τη διαθεσιμότητα των αντικειμένων, αφού τα μη διαμοιράσιμα κλειδώματα έχουν πολύ μικρή διάρκεια.

Και στις τρεις μεθόδους που είδαμε, η ανάγνωση πληροφοριών από τη hypermedia βάση δεδομένων θεωρείται δραστηριότητα που μπορεί να γίνεται ταυτόχρονα από πολλούς χρήστες μαζί για τα ίδια δεδομένα, ενώ η τροποποίηση των πληροφοριών θεωρείται "αποκλειστική" δραστηριότητα, δηλαδή δραστηριότητα που μόνο ένας χρήστης θα μπορεί να έχει κάθε φορά. Υπάρχουν, ωστόσο, περιπτώσεις, που είναι σκόπιμο να κάνουμε ένα λεπτότερο χωρισμό δραστηριοτήτων σε αποκλειστικές και μη. Δύο σχήματα που μπορούν να χρησιμοποιηθούν ([HAL88] και [GARR86]) είναι:

(1) Η ανάγνωση θεωρείται μη αποκλειστική δραστηριότητα. Η τροποποίηση διακρίνεται σε *σχολιασμό* (annotation) και *εγγραφή* (write). Ο σχολιασμός, δηλαδή η προσθήκη σχολίων ή/και σημειώσεων στις αποθηκευμένες πληροφορίες (πράγμα που στο δικό μας μοντέλο μεταφράζεται σε δημιουργία καινούριων items με σχόλια ή/και σημειώσεις και καινούριων συνδέσμων προς τα items αυτά), θεωρείται μη αποκλειστική δραστηριότητα. Η εγγραφή, που σημαίνει αλλαγή της αποθηκευμένης πληροφορίας, θεωρείται αποκλειστική δραστηριότητα.

(2) Η ανάγνωση θεωρείται μη αποκλειστική δραστηριότητα. Η τροποποίηση διακρίνεται σε *δημιουργική τροποποίηση* (constructive modification) και σε *καταστροφική τροποποίηση* (destructive modification). Η δημιουργική τροποποίηση, δηλαδή η προσθήκη πληροφοριών και η δημιουργία νέων αντικειμένων, θεωρείται μη αποκλειστική δραστηριότητα. Η καταστροφική τροποποίηση, δηλαδή η αλλαγή ή διαγραφή πληροφοριών και η διαγραφή αντικειμένων, θεωρείται αποκλειστική δραστηριότητα.

Με τα σχήματα αυτά αυξάνεται η πολυπλοκότητα του μηχανισμού που ελέγχει τη σύγχρονη προσπέλαση, ταυτόχρονα όμως αυξάνεται και η διαθεσιμότητα των αντικειμένων και, κατά συνέπεια, μειώνεται ο χρόνος που θα χρειαστεί να αναμένουν οι χρήστες για να εκτελέσουν μια επεξεργασία.

Τέλος, μιλώντας για σύγχρονη προσπέλαση θα πρέπει να κάνουμε ειδική μνεία στο [NEU88], όπου περιγράφεται ένα πλήρες μοντέλο για δοσοληψίες σε object-oriented βάσεις δεδομένων, και προτείνονται διάφοροι μηχανισμοί για τον έλεγχο της σύγχρονης προσπέλασης.

7.6. Βιβλιογραφία

1. [AKS88] Robert M. Akscyn, Donald L. McCracken και Elise A. Yoder, "KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations", CACM, τεύχος 31/7, Ιούλιος 1988.
2. [DAT84] C. J. Date, "An Introduction to Database Systems", Addison-Wesley, 1984.
3. [GARR86] L. Garrett, K. Smith και N. Meyrowitz, "Intermedia: Issues, Strategies and Tactics in the Design of the Hypermedia Document System", πρακτικά CSCW'86, Δεκέμβριος 1986.
4. [HAL88] Frank G. Halasz, "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems", CACM, τεύχος 31/7, Ιούλιος 1988.
5. [KOR86] H. Korth, A. Silberschatz, "Database System Concepts", 1986.
6. [NEU88] Erich J. Neuhold, Junzhong Gu και Thomas C. Rakow (Institute for Integrated Publication and Information Systems), "An Object-Oriented Transaction Model", 1988.

8. USER INTERFACE ΓΙΑ HYPERMEDIA ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

8.1. Απαιτήσεις για το user interface σε περιβάλλον hypermedia

Το user interface είναι ένα από τα πιο σημαντικά στοιχεία ενός υπολογιστικού συστήματος, αφού αποτελεί τον διάδρομο επικοινωνίας του συστήματος με τους χρήστες. Αν και αυτό είναι κοινά παραδεκτό, μόνο τα τελευταία χρόνια έχει εκδηλωθεί ενδιαφέρον προς την κατεύθυνση της βελτίωσης του user interface. Ως τώρα, κυρίως στα μεγάλα υπολογιστικά συστήματα, συναντούσαμε interfaces που κάθε άλλο παρά φιλικά προς τους χρήστες ήταν, με αποτέλεσμα να αποθαρρύνονται οι τελευταίοι από τη χρήση των υπολογιστών. Τα προβλήματα που θα αντιμετωπίσουμε αν προσεγγίσουμε το user interface με τον τρόπο που ως τώρα έχει ακολουθηθεί, είναι περισσότερα σε ένα περιβάλλον hypermedia, αφενός λόγω του τεράστιου όγκου πληροφορίας που υπάρχει μέσα στη βάση δεδομένων, και αφετέρου λόγω της πολυπλοκότητας των λειτουργιών που θα πρέπει να υποστηρίζονται. Είναι σαφές ότι θα πρέπει να υιοθετήσουμε μία εναλλακτική προσέγγιση, προσανατολισμένη στις ανάγκες των χρηστών. Έτσι, κατά το σχεδιασμό του user interface θα πρέπει να λάβουμε υπόψη τα ακόλουθα:

(1) Στο επίπεδο του υλικού, θα πρέπει να υπάρχει ο κατάλληλος εξοπλισμός για την υποστήριξη της δημιουργίας, παρουσίασης και τροποποίησης κάθε τύπου δεδομένων που μπορεί να περιέχει η hypermedia βάση δεδομένων. Στη κατηγορία αυτή υπάγονται *δειγματολήπτες* (samplers), *σαρωτές εικόνας* (scanners), ειδικές οθόνες για παρουσίαση γραφικών, ηχεία κλπ. Θα πρέπει επίσης να υπάρχουν περιφερειακά με τα οποία ο χρήστης θα αλληλεπιδρά με το υπολογιστικό σύστημα, όπως πληκτρολόγια, *ποντίκια* (mice), *οπτικά μολύβια* (light pens), κ.ά. Θα πρέπει να χρησιμοποιούνται οθόνες bitmapped, που να επιτρέπουν την παρουσίαση όχι μόνο κειμένου αλλά και γραφικών.

(2) Το user interface θα πρέπει να έχει τη *δυνατότητα αλλαγών* (modifiability), τη *δυνατότητα επεκτάσεων* (extensibility) προκειμένου να καλύπτει καινούριες ανάγκες, και τη *δυνατότητα να προσαρμόζεται* (tailorability) στις ανάγκες ή τις προτιμήσεις του κάθε χρήστη.

(3) Θα πρέπει να παρέχονται εργαλεία και εποπτικά μέσα που να επιτρέπουν τόσο την *περιπλάνηση* (navigation) ενός χρήστη μέσα στη βάση δεδομένων, όσο και την *υποβολή ερωτήσεων* (querying).

(4) Ο χειρισμός των αντικειμένων θα πρέπει να είναι όσο το δυνατόν πιο *ομοιόμορφος* (uniform). Αυτό θα βοηθήσει τους χρήστες να εξοικειωθούν πιο γρήγορα με το interface, και θα τους επιτρέψει να εκτελέσουν πιο αποδοτικά τις εργασίες τους ([TAN89α]).

(5) Οι λειτουργίες που εκτελούνται από το σύστημα είναι πολύπλοκες, οι χρήστες όμως πρέπει να προφυλάσσονται από αυτή την πολυπλοκότητα και να μπορούν να εκτελέσουν όλες τις λειτουργίες με απλό τρόπο. Οι βασικές λειτουργίες θα πρέπει να εκτελούνται σε ένα κατά κανόνα και δύο το πολύ βήματα ([TAN89α]).

(6) Το user interface θα πρέπει να είναι όσο πιο "φυσικό" γίνεται. Η "φυσικότητα" δεν είναι κάτι που μπορεί να προσδιοριστεί με τυπικά κριτήρια, αλλά μόνο διαισθητικά. Μεταξύ των χαρακτηριστικών που προάγουν την φυσικότητα του user interface είναι ([PAR89]):

(α) η χρήση όχι μόνο κειμένου, αλλά ακόμα ήχου και εικόνων,

(β) η δυνατότητα να βλέπουν οι χρήστες πιο γενικές και πιο ειδικές απόψεις των αντικειμένων,

(γ) η δυνατότητα να εκτελούν οι χρήστες *συσχετιστικές αναζητήσεις* (associative search) (δηλαδή να αναζητούν αντικείμενα σχετικά με αυτό που εξετάζουν),

(δ) η χρήση αντικειμένων και νοημάτων με τα οποία οι χρήστες έχουν εξοικειωθεί ([TAN89α]),

(ε) η παρουσίαση της πληροφορίας σύμφωνα με τις ανάγκες των χρηστών και

(στ) η δυνατότητα να αλληλεπιδρούν οι χρήστες με τα αντικείμενα του interface απευθείας, και όχι μόνο μέσω κάποιας *γλώσσας εντολών* (command language).

Υπάρχει ακόμα η άποψη ότι ένα interface γίνεται πιο φυσικό αν ο χρήστης μπορεί να δώσει εντολές τόσο με ένα ποντίκι (mouse), όσο και χρησιμοποιώντας κάποια γλώσσα, φυσική ή μη ([POG85]).

(7) Η εκμάθηση του user interface από τους χρήστες θα πρέπει να είναι εύκολη. Ας σημειωθεί ότι αυτό είναι ένα ξεχωριστό σημείο από τα προηγούμενα, καθώς είναι δυνατό να διαθέτει ένα interface όλα τα προηγούμενα χαρακτηριστικά και η εκμάθησή του να είναι παρόλα αυτά δύσκολη, λόγω

κακού σχεδιασμού ή άλλων ατελειών.

(8) Το user interface θα πρέπει να παρέχει εργαλεία που θα βοηθούν τους χρήστες σε περιπτώσεις *αποπροσανατολισμού* (disorientation), δηλαδή όταν συμβεί, κατά τη διάρκεια της περιπλάνησης, να μην ξέρουν οι χρήστες σε ποιο σημείο της βάσης δεδομένων βρίσκονται ή πως βρέθηκαν εκεί ([CON87]).

(9) Οι χρήστες δεν μπορούν με κανέναν τρόπο να διαχειριστούν όλη την πληροφορία που υπάρχει στη βάση δεδομένων. Το user interface θα πρέπει να φιλτράρει την πληροφορία αυτή και να παρουσιάζει στους χρήστες μόνο ένα κομμάτι της. Στην ιδανική περίπτωση θα παρουσιάζεται σε κάθε χρήστη η πληροφορία που τον ενδιαφέρει και μόνο αυτή. Βέβαια, στο προτεινόμενο μοντέλο, η αρμοδιότητα για το φιλτράρισμα της πληροφορίας έχει μεταφερθεί από το επίπεδο του user interface στο επίπεδο της βάσης δεδομένων, όπου ορίζονται και οι όψεις.

(10) Οι σημασιολογικοί δεσμοί μεταξύ των αντικειμένων θα πρέπει να φαίνονται καθαρά. Έτσι, κατά την παρουσίαση ενός item στον χρήστη, θα πρέπει να φαίνονται τα σημεία απ' όπου ξεκινούν ή καταλήγουν σύνδεσμοι. Ο τρόπος της παρουσίασης θα εξαρτάται από την φύση των items (text, image, κλπ.), και ενδεχομένως και από τους τύπους των συνδέσμων.

(11) Το user interface θα πρέπει να παρέχει προσπέλαση σε αυτά μέσω παραθύρων ή άλλων μηχανισμών σε εργαλεία υψηλού επιπέδου που θα αποτελούν αναπόσπαστο τμήμα της βάσης δεδομένων, όπως εργαλεία για τον έλεγχο της ακεραιότητας και της συνέπειας των δεδομένων, πακέτα λογισμικού για την επεξεργασία των διαφόρων τύπων δεδομένων, εργαλεία που να βοηθούν τους χρήστες να ανακαλύψουν δεσμούς μεταξύ αντικειμένων, εργαλεία για υποστήριξη αποφάσεων και ανάλυση καταστάσεων, κλπ.

(12) Θα πρέπει να μπορεί ο χρήστης να βλέπει ταυτόχρονα περισσότερα από ένα αντικείμενα στην οθόνη. Για να γίνει κάτι τέτοιο θα πρέπει το σύστημα να υποστηρίζει παράθυρα, τα οποία κατά προτίμηση θα ακολουθούν κάποια καθιερωμένα *πρότυπα* (standards) (πχ. X-Windows).

(13) Θα πρέπει να υπάρχει *βοήθεια κατά τη χρήση* (on-line help), η οποία θα εξηγεί στους χρήστες το τι μπορούν ή πρέπει να κάνουν στις διάφορες περιστάσεις. Η βοήθεια αυτή θα πρέπει να είναι σαφής και εύληπτη, αλλά όχι τόσο εκτεταμένη ώστε να γίνεται δύσχρηστη ή κουραστική.

Η κατασκευή ενός user interface με όλα τα πιο πάνω χαρακτηριστικά, σίγουρα δεν είναι απλή υπόθεση. Θα πρέπει, ωστόσο, να γίνει κατανοητό ότι η υποστήριξη των χαρακτηριστικών αυτών, στο σύνολό τους ή κατά ένα μεγάλο μέρος, είναι απολύτως απαραίτητη αν θέλουμε να εκμεταλλευτούμε τις δυνατότητες που μας παρέχουν οι hypermedia βάσεις δεδομένων.

8.2. Μοντέλο για user interface σε περιβάλλον hypermedia

Βασισμένοι στις απαιτήσεις για το user interface που παρουσιάσαμε στην προηγούμενη παράγραφο, θα προχωρήσουμε εδώ στην περιγραφή ενός μοντέλου για user interface που θα καλύπτει τις απαιτήσεις αυτές. Υποθέτουμε ότι είναι διαθέσιμα, από την πλευρά του υλικού ο εξοπλισμός που περιγράψαμε στην προηγούμενη παράγραφο, και από την πλευρά του λογισμικού κάποιο σύστημα παραθύρων.

Κάθε αντικείμενο που ο χρήστης επιλέγει θα παρουσιάζεται σε ένα χωριστό παράθυρο της οθόνης. Οι χρήστες θα μπορούν να κινήσουν, να μικρύνουν ή να μεγαλώσουν τα παράθυρα που υπάρχουν οποιαδήποτε στιγμή στην οθόνη, φτιάχνοντας με αυτόν τον τρόπο την οθόνη σύμφωνα με τις προσωπικές τους προτιμήσεις.

Επειδή το πλήθος των παραθύρων που μπορούν να απεικονιστούν ταυτόχρονα στην οθόνη, καθώς και το πλήθος των αντικειμένων που μπορούν κάποια δεδομένη στιγμή να βρίσκονται στη μνήμη, είναι περιορισμένο, θα πρέπει να υπάρχει κάποια πολιτική αντικατάστασης, θα πρέπει δηλαδή να ορίζεται πότε και πώς ένα αντικείμενο θα παύει να απεικονίζεται στην οθόνη, προκειμένου να παραχωρήσει τη θέση του σε κάποιο άλλο. Μια απλή αλλά αρκετά αποδοτική τεχνική που μπορεί να χρησιμοποιηθεί εδώ είναι η τεχνική FIFO, σύμφωνα με την οποία το αντικείμενο που ανακτήθηκε πρώτο θα είναι αυτό που πρώτο θα παραχωρεί τη θέση του σε κάποιο καινούριο, όταν το πλήθος των διαθέσιμων παραθύρων εξαντλείται. Ο κάθε χρήστης, βέβαια, θα μπορεί να ορίζει αν η αντικατάσταση θα γίνεται με το σχήμα FIFO ή οποιοδήποτε άλλο σχήμα τελικά χρησιμοποιηθεί ή αν, όταν πρέπει να γίνει αντικατάσταση, θα αποφασίζει ο ίδιος για το ποιο αντικείμενο θα αντικατασταθεί.

Όταν το αντικείμενο που επιλέγεται προς παρουσίαση δε χρησιμοποιεί την οθόνη για απεικόνιση (π.χ. όταν είναι ένα item που περιέχει ήχο ή όταν χρησιμοποιείται για την απεικόνισή του κάποια ειδικό

περιφερειακό), τότε το αντίστοιχο παράθυρο θα καταλαμβάνεται εξ ολοκλήρου από τα σημασιολογικά δεδομένα που περιλαμβάνει το αντικείμενο, ενώ στην αντίθετη περίπτωση ο χώρος του παραθύρου θα μοιράζεται ανάμεσα στα σημασιολογικά και μη δεδομένα.

Θα πρέπει ακόμη να υπάρχει ένα παράθυρο από το οποίο οι χρήστες θα διατυπώνουν ερωτήσεις προς τη βάση δεδομένων, και στο οποίο θα παρουσιάζονται τα αποτελέσματα των ερωτήσεων αυτών, και ένα δεύτερο παράθυρο από το οποίο οι χρήστες θα μπορούν να δίνουν εντολές του λειτουργικού συστήματος. Το να υπάρχουν ξεχωριστά παράθυρα για τη βοήθεια κατά τη χρήση και τα διαγράμματα, τους χάρτες και τα άλλα εποπτικά μέσα δεν είναι απαραίτητο, αφού μπορούν να χρησιμοποιηθούν για τους σκοπούς αυτούς τα παράθυρα που χρησιμοποιούνται και για την παρουσίαση των αντικειμένων.

Η επεκτασιμότητα και η τροποποιησιμότητα του user interface, καθώς και η ομοιόμορφη αντιμετώπιση των αντικειμένων, εξασφαλίζεται από τον σχεδιασμό του σύμφωνα με το object-oriented μοντέλο, αφού τα προαναφερθέντα χαρακτηριστικά είναι εγγενή στο μοντέλο αυτό. Η δυνατότητα συσχετιστικής αναζήτησης παρέχεται από τους συνδέσμους και τις ιδιότητες, ή ακόμη και από εργαλεία *αναζήτησης κειμένου* (text searching). Η προσαρμοστικότητά του user interface στις ανάγκες κάθε χρήστη μπορεί να εξασφαλιστεί από την ύπαρξη παραμέτρων που θα έχουν αρχικά κάποιες *προκαθορισμένες τιμές* (default values), τις οποίες όμως οι χρήστες θα μπορούν να αλλάζουν. Οι τιμές αυτές μπορεί να αφορούν το μέγιστο πλήθος παραθύρων που θα βρίσκονται στην οθόνη, το μέγεθος και τη θέση τους, τα χρώματα που θα χρησιμοποιούνται, το σχήμα των διαφόρων κουμπιών, κ.ά.

8.3. Περιπλάνηση σε hypermedia βάσεις δεδομένων

Η περιπλάνηση σε μια βάση δεδομένων είναι μια από τις πιο σημαντικές λειτουργίες που μπορούν οι χρήστες να επιτελέσουν. Ειδικότερα σε ένα εκπαιδευτικό περιβάλλον, για παράδειγμα, η περιπλάνηση μπορεί να αποδειχθεί σημαντικό βοήθημα για να ανακαλύπτουν οι εκπαιδευόμενοι καινούρια γνώση και να τη συνδέουν με τη γνώση που ήδη κατέχουν ([KIB89]). Για την υποστήριξη της περιπλάνησης το user interface θα πρέπει να περιλαμβάνει έναν browser ([TAN89β]), και άλλα βοηθητικά εργαλεία και εποπτικά μέσα, μερικά από τα οποία θα είναι:

(1) *Τοπικοί* και *γενικοί χάρτες* (local και global maps) της βάσης δεδομένων ([GARR86]). Ένας τοπικός χάρτης θα δείχνει τα items που συνδέονται με το item το οποίο εκείνη τη στιγμή ο χρήστης εξετάζει, παρέχοντας αρκετές πληροφορίες για τους εισερχόμενους και εξερχόμενους συνδέσμους (ίσως τις άγκυρες και τους τύπους τους), ενώ ένας γενικός χάρτης θα δείχνει ένα μεγαλύτερο μέρος της βάσης δεδομένων, δίνοντας όμως λιγότερη πληροφορία για τις συνδέσεις.

(2) *Γενικά διαγράμματα* (overview diagrams) που θα απεικονίζουν ολόκληρη τη hypermedia βάση δεδομένων ως ένα δίκτυο από items και συνδέσμους, ή θα λειτουργούν ως ένα κυλιόμενο παράθυρο πάνω από το δίκτυο αυτό και θα μπορούν να *εστιάζουν από μακριά* ή *από κοντά* (zoom-in ή zoom-out, αντίστοιχα) ([HAR89α], [KIB89] και [SHU89]). Ανάλογα με το πόσο πολύπλοκο θα αποφασιστεί να είναι ένα τέτοιο διάγραμμα, μπορεί να αποδοθεί σημασία στις αποστάσεις μεταξύ των κόμβων του δικτύου, στο είδος των γραμμών με τις οποίες συνδέονται οι κόμβοι, στην προοπτική (αν το σχήμα είναι τρισδιάστατο) και σε άλλα χαρακτηριστικά. Αν πρόκειται για μεγάλα δίκτυα, ένα τέτοιο διάγραμμα μπορεί να έχει και τη δομή ενός *υπεργράφου* (hypergraph), δηλαδή ενός γράφου του οποίου οι κόμβοι είναι επίσης γράφοι κοκ., μέχρι 7ρ2 επίπεδα ([SHU89]). Υπάρχουν, ωστόσο, εδώ, αρκετά προβλήματα αναπαράστασης ([TAN89β]).

(3) Ειδική *εντολή οπισθοχώρησης* (backtrack command) που θα ανακαλεί το αμέσως προηγούμενο item που επισκέφθηκε ένας χρήστης ([AKS88]). Αρκετές φορές είναι δυνατό να ακολουθήσει κάποιος χρήστης ένα σύνδεσμο που δεν καταλήγει στις ζητούμενες πληροφορίες, ή να βρεθεί για οποιοδήποτε λόγο σε κάποιο item που δεν τον ενδιαφέρει· για τις περιπτώσεις αυτές είναι καλό να παρέχεται κάποιος άμεσος τρόπος να γυρίσει ο χρήστης πίσω στο προηγούμενο item ([HAR89β]).

(4) Ένα αρχείο για κάθε χρήστη με το ιστορικό της πορείας του μέσα στη βάση δεδομένων, έτσι ώστε αυτός να μπορεί να δει τα items τα οποία εξέτασε, και ενδεχομένως να ξαναγυρίσει σε ένα από αυτά ([APP87] και [HAR89β]).

(5) *Όψεις προσανατολισμού* (landmark views) και *όψεις προοπτικής* (fisheye views) ([PAR89], [HAR89β] και [FUR86]). Μια όψη προσανατολισμού θα δείχνει τη σχετική θέση κάποιων αντικειμένων που έχουν οριστεί ως σημαντικά από ένα χρήστη, ως προς το αντικείμενο το οποίο ο χρήστης αυτή τη στιγμή εξετάζει. Αυτό γίνεται κατ' αναλογία με τη συνήθεια των ανθρώπων να

προσανατολίζονται μέσα στις πόλεις με το να απομνημονεύουν κάποια σημαντικά σημεία, και κατόπιν να εντοπίζουν τα υπόλοιπα σημεία σε σχέση με αυτά. Μια όψη προοπτικής θα δείχνει μια γενική άποψη της βάσης δεδομένων με μεγαλύτερη όμως λεπτομέρεια στα κοντινά σημεία.

(6) Προκαθορισμένες διαδρομές. Όταν κάποιος χρήστης εντοπίζει κάποια ενδιαφέρουσα διαδρομή θα πρέπει να μπορεί να την αποθηκεύει, έτσι ώστε να χρησιμοποιηθεί αργότερα ([HAR89β]).

(7) Home items για όλους τους χρήστες (ανάλογα με το "home item" που συναντάμε στο HyperCard ([APP87])). Κάθε χρήστης θα μπορεί οποιαδήποτε στιγμή να μεταφερθεί στο δικό του home item με μια συγκεκριμένη εντολή.

(8) Δυνατότητα για αντιστοίχηση συμβολικών ονομάτων σε items και μια ειδική εντολή "Πήγαινε" ("Go to"), η οποία θα ανακαλεί κάποιο συγκεκριμένο item που ο χρήστης θα ορίζει ([HAR89β]). Η δυνατότητα αυτή μπορεί να εκλεπτυνθεί με το να μη μαρκάρονται απλώς ολόκληρα items αλλά και τμήματα multimedia πληροφοριών μέσα σε items ([TAN89β]).

Η περιπλάνηση θα γίνεται στο μεγαλύτερο μέρος της με τη βοήθεια ενός ποντικιού, ενώ, αν ο χρήστης το επιθυμεί, θα μπορεί να επιτυγχάνεται μέσω εντολών κάποιας ειδικής γλώσσας. Οι άγκυρες των συνδέσμων μέσα σε κάθε item θα ορίζουν *κουμπιά* (buttons)· αν κάποιος χρήστης πατήσει ένα κουμπί οδηγώντας τον κέρσορα του ποντικιού πάνω του και πατώντας ένα από τα κουμπιά του ποντικιού, θα διασχίσει αυτόματα τον αντίστοιχο σύνδεσμο και θα ανακτήσει το item που βρίσκεται στο άλλο άκρο του συνδέσμου. Τα κουμπιά που ορίζονται από τις άγκυρες των συνδέσμων θα πρέπει να είναι εμφανή στον χρήστη. Έτσι, σε items τύπου text, μπορεί το κείμενο που υπάρχει μέσα στην άγκυρα να εμφανίζεται υπογραμμισμένο ή με αντεστραμμένα χρώματα, σε items τύπου image μπορεί η άγκυρα και το αντίστοιχο κουμπί να ορίζονται από ένα πλαίσιο, κοκ. Οι σύνδεσμοι που ξεκινάνε από, ή καταλήγουν σε, ένα ολόκληρο item, και τα αντίστοιχα κουμπιά, μπορούν να παρουσιάζονται όχι στο παράθυρο του item αλλά στο παράθυρο όπου βρίσκονται τα σημασιολογικά δεδομένα.

8.4. Υποβολή ερωτήσεων σε hypermedia βάσεις δεδομένων

Μια εναλλακτική προσέγγιση για την ανάκτηση πληροφορίας από μια βάση δεδομένων, είναι η υποβολή ερωτήσεων. Τόσο η περιπλάνηση όσο και η υποβολή ερωτήσεων έχουν δοκιμαστεί στα κλασικά συστήματα διαχείρισης βάσεων δεδομένων (η υποβολή ερωτήσεων στο σχεσιακό μοντέλο και περιπλάνηση στο ιεραρχικό και το δικτυωτό), και έχουν διαπιστωθεί πλεονεκτήματα και μειονεκτήματα σε κάθε μία από αυτές (για παράδειγμα, η SQL δεν είναι αρκετά εκφραστική για να υπολογίσει το *μεταβατικό κλείσιμο* (transitive closure) κάποιας σχέσης, ενώ η *αποτίμηση* (evaluation) μιας ερώτησης στο περιβάλλον μιας δικτυωτής ή ιεραρχικής βάσης δεδομένων απαιτεί πολύ περισσότερη προσπάθεια από την πλευρά του χρήστη απ' ότι θα απαιτούσε, π.χ., με SQL). Η δυνατότητα υποβολής ερωτήσεων είναι απαραίτητο συμπλήρωμα στην περιπλάνηση σε ένα περιβάλλον hypermedia, αφού η περιπλάνηση μέσα σε μια βάση με πάνω από 500 περίπου αντικείμενα γίνεται προβληματική· επίσης, δεν είναι σπάνιο το φαινόμενο να μπορούν οι χρήστες να περιγράψουν επακριβώς την πληροφορία που θέλουν, αλλά να μην ξέρουν που βρίσκεται ([HAL88] και [CON87]). Πέρα από αυτά, ανάλογα με την φύση της εφαρμογής, η μία ή η άλλη από αυτές τις προσεγγίσεις μπορεί να φαίνεται πιο "φυσική", ενώ θα πρέπει να ληφθεί υπόψη ότι άλλοι χρήστες προτιμούν, γενικά, την υποβολή ερωτήσεων, και άλλοι την περιπλάνηση. Από αυτά καθίσταται φανερό ότι η βάση δεδομένων θα πρέπει να υποστηρίζει ένα μηχανισμό υποβολής ερωτήσεων, δηλαδή κάποια γλώσσα στην οποία οι ερωτήσεις αυτές θα διατυπώνονται. Η γλώσσα αυτή θα πρέπει να είναι απλή στην εκμάθησή της, αλλά παράλληλα εκφραστική και κοντινή από άποψη μορφής στη φυσική γλώσσα, και θα πρέπει να επιτρέπει στο χρήστη να εκμεταλλευτεί το σύνολο των δυνατοτήτων της βάσης δεδομένων. Στο παράρτημα Β περιγράφεται η HYQL (Hypermedia Query Language), η οποία πιστεύουμε ότι ανταποκρίνεται στις προδιαγραφές αυτές. Η HYQL υποστηρίζει και τους δύο τύπους αναζήτησης που αναφέρονται στο [HAL88], δηλαδή την *αναζήτηση ως προς το περιεχόμενο* (content search) και την *αναζήτηση ως προς τη δομή* (structure search). Στην αναζήτηση ως προς το περιεχόμενο ο χρήστης μπορεί να χρησιμοποιήσει κριτήρια που αφορούν την περιγραφή ενός αντικειμένου, τις ιδιότητές του, το συγγραφέα του, τη χρονική στιγμή που δημιουργήθηκε, τον τύπο του και το περιεχόμενό του (σ' αυτό το σημείο, ωστόσο, υπάρχουν ακόμη ανοιχτά ζητήματα, όπως, για παράδειγμα, η αναζήτηση ενός τμήματος εικόνας μέσα σε ένα αντικείμενο τύπου image, ή ενός τμήματος ήχου μέσα σε ένα αντικείμενο τύπου audio). Στην αναζήτηση ως προς τη δομή οι χρήστες μπορούν να χρησιμοποιήσουν κριτήρια που αφορούν τη συσχέτιση των items μεταξύ τους μέσω συνδέσμων (δηλαδή το αν κάποια items συσχετίζονται με

σύνδεσμο ο οποίος έχει έναν καθορισμένο τύπο ή βάρος που να βρίσκεται σε κάποιο διάστημα, ή το αν το αρχικό ή τελικό item κάποιου συνδέσμου πληρούν κάποιες προϋποθέσεις, κτλ.). Θα πρέπει να σημειώσουμε ότι οι δύο αυτές μορφές αναζήτησης μπορούν να συνδυαστούν, καθώς κριτήρια και των δύο τύπων μπορούν να χρησιμοποιηθούν στην ίδια ερώτηση.

Το γεγονός αυτό, σε συνδυασμό με την υποστήριξη *φωλιασμένων ερωτήσεων* (nested queries), κάνει την HYQL μια πολύ ισχυρή γλώσσα αναζήτησης. Στην HYQL εισάγεται η έννοια της *λίστας επικοινωνίας* (communication list). Η λίστα επικοινωνίας έχει σκοπό να αποθηκεύει τα αποτελέσματα των ερωτήσεων που οι χρήστες υποβάλουν στην βάση δεδομένων (τους προσδιοριστές των αντικειμένων ή, στην ειδική περίπτωση που πρόκειται για items, και την περιγραφή που αντιστοιχεί σε κάθε προσδιοριστή, τον τύπο του item και τη λίστα των ιδιοτήτων), και από αυτήν οι χρήστες μπορούν να διαλέγουν πλέον τα αντικείμενα τα οποία θα επεξεργαστούν. Η ύπαρξη της λίστας επικοινωνίας επιβάλλεται για δύο λόγους:

(1) Αντίθετα με ότι συμβαίνει στις κλασικές βάσεις δεδομένων, σε μια hypermedia βάση δεδομένων είναι πρακτικά αδύνατο να παρουσιαστούν ταυτόχρονα όλα τα αντικείμενα που επιστρέφονται στο αποτέλεσμα μιας ερώτησης, αφού αφενός το πλήθος των παραθύρων είναι περιορισμένο, και αφετέρου η ταυτόχρονη παρουσίαση περισσοτέρων του ενός αντικειμένων τύπου audio, κινούμενων γραφικών ή video δεν έχει νόημα γιατί ξεπερνά τις δυνατότητες αντίληψης των χρηστών.

(2) Σε αρκετές περιπτώσεις είναι απαραίτητο να επιλέξει ο χρήστης ακριβώς ένα αντικείμενο (π.χ. στην ερώτηση που προσδιορίζει το αρχικό item ενός συνδέσμου), και έτσι, με τη λίστα επικοινωνίας, απαλλάσσεται ο χρήστης από την υποχρέωση να επαναδιατυπώσει την αρχική ερώτηση με πιο περιοριστικά κριτήρια, προκειμένου το αποτέλεσμά της να είναι ένα μόνο αντικείμενο. Η διαδικασία αυτή θα ήταν άλλωστε χρονοβόρα, αφού θα έπρεπε να αποτιμηθεί ξανά το αποτέλεσμα της ερώτησης.

Το συντακτικό, τέλος, της HYQL, βρίσκεται αρκετά κοντά σε αυτό της φυσικής γλώσσας, έτσι ώστε η εκμάθησή της να μην απαιτεί μεγάλη προσπάθεια.

8.5. Βιβλιογραφία

1. [AKS88] Robert M. Akscyn, Donald L. McCracken και Elise A. Yoder, "KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations", CACM, τεύχος 31/7, Ιούλιος 1988.
2. [APP87] Apple Computers, "HyperCard: User's Guide", Apple Computers, 1987.
3. [CON87] Jeff Conklin, "A Survey of Hypertext", MCC Technical Report, τεύχος STP-356-86/2.
4. [FUR86] G.W. Furnas, "Generalised Fisheye Views", πρακτικά ACM Conference on Human Factors in Computing Systems, 1986.
5. [GARR86] L. Garrett, K. Smith και N. Meyrowitz, "Intermedia: Issues, Strategies and Tactics in the Design of a Hypermedia Document System", πρακτικά CSCW'86, Δεκέμβριος 1986.
6. [HAL88] Frank G. Halasz, "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems", CACM, τεύχος 31/7, Ιούλιος 1988.
7. [HAR89α] Lynda Hardman, "Navigation in a Hypermedia Environment", SAFE/HYP/OWL-pap/elts_of_nav, Σεπτέμβριος 1989.
8. [HAR89β] Lynda Hardman, "Requirements of Navigation Metaphors in a Hypermedia Learning Environment", SAFE/HYP/OWL-pap/rqts_of_nav_met, Σεπτέμβριος 1989.
9. [KIB89] Mike Kibby, "The Learner's View of Hypermedia", SAFE/HYP/wp/mk080989, Σεπτέμβριος 1989.
10. [PAR89] Kamran Parsaye, M. Chingel, S. Khoshafian και H. Wong, "Intelligent Databases", Wiley, 1989.
11. [POG85] A. Poggio, J. Garcia Luna Aceves, E. Craighill, D. Morgan, L. Aguilar, D. Worthington και J. Hight, SRI International, "CCWS: A Computer-Based, Multimedia Information System", IEEE Computer, Οκτώβριος 1985.
12. [SHU89] Simon Shum, "Enriching the Spatial Environment for Hypertext Browsers: Developing an Approach to Browser Design", PACIS Workshop, Ανοιχτό Πανεπιστήμιο, Μάιος 1989.
13. [TAN89α] Gary Tanner, "Navigating Through Hyperchaos", SAFE/HYP/HCI-pap/GT240789, Ιούλιος 1989.
14. [TAN89β] Gary Tanner, "On the Use of Metaphor", SAFE/HYP/HCI-pap/GT010989, Σεπτέμβριος 1989.

9. ΑΞΙΟΛΟΓΗΣΗ ΤΟΥ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

Στο καταληκτικό αυτό κεφάλαιο παρουσιάζονται συνοπτικά τα βασικά χαρακτηριστικά του σχεδιασμού που προτείνεται για hypermedia βάσεις δεδομένων και υποδεικνύονται οι λειτουργίες διαχείρισης βάσεων δεδομένων που μπορεί ο σχεδιασμός αυτός να υποστηρίξει. Στη συνέχεια αναφέρονται ορισμένες δυνατότητες επέκτασης του προτεινόμενου σχεδιασμού και τέλος παρουσιάζονται κάποια γενικά συμπεράσματα.

9.1. Συνοπτική παρουσίαση του σχεδιασμού και σημαντικά χαρακτηριστικά του

Ο προτεινόμενος σχεδιασμός για hypermedia βάσεις δεδομένων ξεκινά από την κεντρική αντίληψη ότι μια hypermedia βάση έχει τη δομή ενός γράφου του οποίου οι κόμβοι περιέχουν multimedia πληροφορία και οι ακμές φέρουν σημασιολογική αξία, και δημιουργεί ένα object-oriented μοντέλο δεδομένων στο οποίο εντάσσονται όλες οι οντότητες της βάσης δεδομένων. Οι βασικές κλάσεις δεδομένων είναι τα items, οι σύνδεσμοι, οι συγγραφείς και οι όψεις. Τα items διαμερίζονται σε υποκλάσεις ανάλογα με τη μορφή των πληροφοριών που αποθηκεύουν, και ανεβαίνοντας την ιεραρχία των κλάσεων, τα items και οι σύνδεσμοι ομαδοποιούνται σε μια κλάση αντικειμένων με συγγραφέα, και τα items και οι όψεις ομαδοποιούνται σε μια κλάση αντικειμένων για τα οποία τηρούνται εκδοχές. Τέλος, όλες οι κλάσεις του μοντέλου δεδομένων έχουν ως κοινό πρόγονο μια αφηρημένη κλάση Object.

Ο σχεδιασμός αυτός έχει τα εξής σημαντικά χαρακτηριστικά:

(1) Παρέχει σε όλα τα αντικείμενα της βάσης δεδομένων ένα μοναδικό προσδιοριστή, δημιουργώντας έτσι από την αρχή ένα μηχανισμό μονοσήμαντης αναφοράς για όλα τα αντικείμενα, ανεξάρτητα από την κλάση όπου ανήκουν.

(2) Διαχωρίζει τα items από τους συνδέσμους σε διαφορετικές κλάσεις, πράγμα που είναι σημασιολογικά προτιμότερο και επιτρέπει να γίνονται επιμέρους τροποποιήσεις χωρίς να είναι ανάγκη να αλλάζει συνολικά το μοντέλο δεδομένων.

(3) Χρησιμοποιεί τόσο συνδέσμους όσο και ιδιότητες για την ανάκτηση και τη διασύνδεση των πληροφοριών, πράγμα που επιτρέπει να χρησιμοποιηθούν αφενός η περιπλάνηση και αφετέρου οι ερωτήσεις για την προσπέλαση της βάσης δεδομένων.

(4) Επιτρέπει να αποθηκευτούν στη βάση δεδομένων τόσο οι ορισμοί των όψεων όσο και οι πληροφορίες για τους συγγραφείς, πράγμα που εξασφαλίζει ομοιομορφία και διευκολύνει την κατανόηση από τρίτους της λογικής του μοντέλου δεδομένων.

(5) Αντιμετωπίζει κατά τον ίδιο τρόπο όλες τις μορφές πληροφοριών multimedia, δίνοντας μια ομοιογενή εικόνα της βάσης δεδομένων στους χρήστες.

(6) Υποστηρίζει κατά τρόπο φυσικό σημαντικές και μη καθιερωμένες λειτουργίες των βάσεων δεδομένων, όπως για παράδειγμα την τήρηση εκδοχών, αλλά και κλασικά χαρακτηριστικά, όπως η επεξεργασία δοσοληψιών.

(7) Επιτρέπει την ανάπτυξη ενός απλού και εύχρηστου user interface, ιδιαίτερα σε συνδυασμό με τη χρήση μιας δηλωτικής γλώσσας χειρισμούς δεδομένων με ισχυρά κριτήρια επιλογής.

Τα χαρακτηριστικά αυτά αποτελούν σημαντικά πλεονεκτήματα και προοιωνίζουν ότι, εφόσον γίνει εφικτό κατά την υλοποίηση να κρατηθεί η απόδοση του συστήματος σε ικανοποιητικά επίπεδα, το αποτέλεσμα θα είναι απόλυτα θετικό.

9.2. Δυνατότητες επέκτασης του σχεδιασμού και συμπεράσματα

Ο προτεινόμενος σχεδιασμός υλοποιήθηκε με την object-oriented και όχι με τη σχεσιακή ή κάποια άλλη φιλοσοφία, ακριβώς για να μπορεί να επεκτείνεται εύκολα. Ήδη κατά την περιγραφή του μοντέλου δεδομένων υποδείχθηκαν δυνατότητες για επέκταση, όπως, για παράδειγμα, στις ακόλουθες περιπτώσεις:

(1) Μπορούν να προστεθούν νέες υποκλάσεις της Item αν πρόκειται να υποστηρίξει η hypermedia βάση και άλλες μορφές δεδομένων.

(2) Μπορούν να προστεθούν σε κάθε υποκλάση της Item κάποιες κλάσεις-παιδιά, αν πρόκειται να

υποστηρίξει η hypermedia βάση και δομημένα multimedia δεδομένα.

(3) Μπορούν να οριστούν υποκλάσεις της Link, σε περίπτωση που αποφασιστεί ότι σύνδεσμοι διαφορετικής σημασιολογίας πρέπει να περιέχουν και διαφορετικά μεταδεδομένα.

Εννοείται, βέβαια, ότι εκτός από την επέκταση του μοντέλου δεδομένων με την προσθήκη καινούριων κλάσεων, μπορούν να προστεθούν όπου χρειαστεί στις κλάσεις που ήδη υπάρχουν καινούριες μεταβλητές και οι αντίστοιχες μέθοδοι.

Μπορούμε, επομένως, να πούμε συμπερασματικά τα εξής:

Ο προτεινόμενος σχεδιασμός ακολουθεί την object-oriented προσέγγιση, πράγμα που εξασφαλίζει την επεκτασιμότητά του, και προορίζεται για hypermedia βάσεις δεδομένων με την ακριβή έννοια του όρου. Έτσι,

- *αντιμετωπίζει με τον ίδιο τρόπο όλες τις μορφές πληροφοριών, έτσι ώστε να μπορεί πραγματικά να θεωρηθεί ότι αφορά συστήματα multimedia,*
- *αποδίδει μεγάλη σημασία στις συνδέσεις μεταξύ των πληροφοριών και δημιουργεί ένα γράφο όπου μπορούν οι χρήστες να περιπλανηθούν, έτσι ώστε να έχουμε όντως ένα μοντέλο hypermedia, και τέλος*
- *επιτρέπει να γίνουν όλες οι τυπικές λειτουργίες μιας βάσης δεδομένων, έτσι ώστε να έχουμε όχι απλώς ένα σύστημα hypermedia αλλά μια hypermedia βάση δεδομένων.*

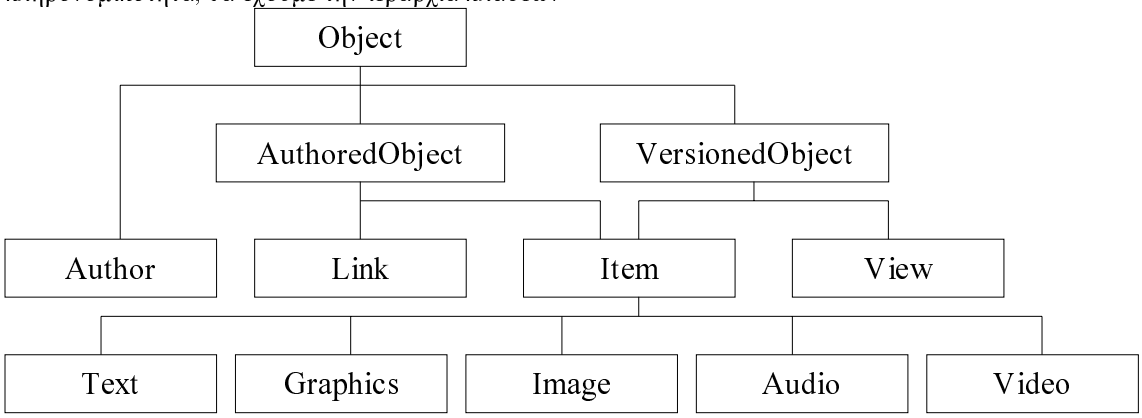
Είναι πεποίθησή μας ότι ο προτεινόμενος σχεδιασμός θα οδηγήσει στην υλοποίηση μιας ισχυρής και λειτουργικής hypermedia βάσης δεδομένων.

Α. ΙΕΡΑΡΧΙΑ ΚΛΑΣΕΩΝ ΓΙΑ ΤΟ ΜΟΝΤΕΛΟ ΔΕΔΟΜΕΝΩΝ

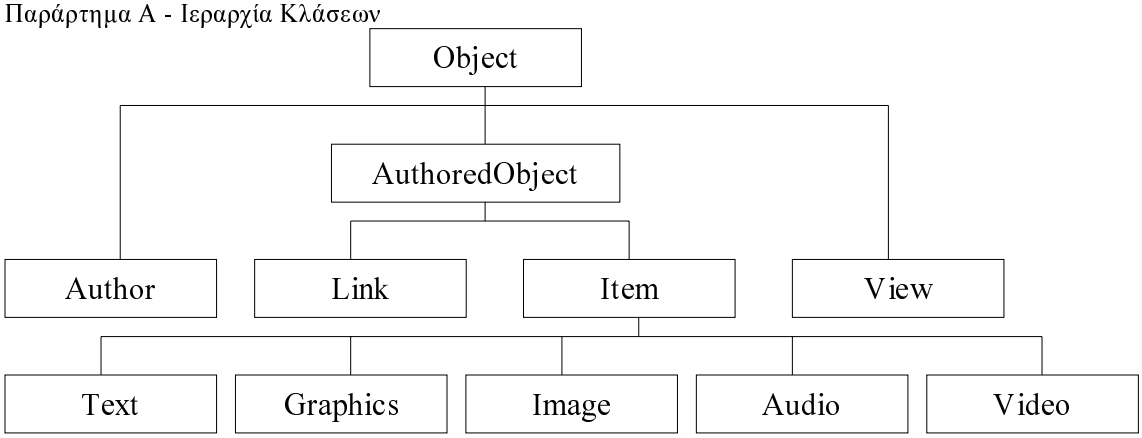
Α.1. Διαγραμματική παρουσίαση

Η ιεραρχία των κλάσεων που σχηματίζεται από το προτεινόμενο μοντέλο δεδομένων μπορεί να παρασταθεί με ένα δέντρο, στην περίπτωση που επιτρέπεται μόνο απλή κληρονομικότητα, ή με έναν ακυκλικό γράφο, στην περίπτωση που επιτρέπεται πολλαπλή κληρονομικότητα. Κάθε κόμβος του δέντρου ή του ακυκλικού γράφου θα αντιστοιχεί σε μια κλάση του μοντέλου δεδομένων και δύο κόμβοι με σχέση γονέα-παιδιού θα αντιστοιχούν σε δύο κλάσεις με σχέση υπερκλάσης-υποκλάσης αντίστοιχα. Θα υπάρχει και στις δύο περιπτώσεις ακριβώς ένας κόμβος χωρίς γονείς, ο οποίος θα αντιστοιχεί στην κλάση Object που θα είναι η ρίζα ολόκληρης της ιεραρχίας κλάσεων. Κάθε κλάση θα έχει δικές της μεταβλητές και μεθόδους, και θα κληρονομεί και τις μεταβλητές και μεθόδους όλων των υπερκλάσεών της μέχρι και την κλάση Object, τις οποίες ενδεχόμενα θα επανορίζει. Αν έχουμε απλή κληρονομικότητα, κάθε κλάση εκτός της Object θα έχει ακριβώς μία υπερκλάση, και για αυτό το λόγο η ιεραρχία των κλάσεων θα έχει τη δομή ενός δέντρου. Αν έχουμε πολλαπλή κληρονομικότητα, κάθε κλάση διαφορετική από την Object θα έχει μία ή περισσότερες υπερκλάσεις, και για αυτό το λόγο η ιεραρχία των κλάσεων θα έχει τη δομή ενός ακυκλικού γράφου.

Υποθέτουμε ότι τηρούμε εκδοχές για τα items και για τις όψεις, οπότε και οι δύο κλάσεις Item και View θα πρέπει να είναι παιδιά της κλάσης VersionedObject. Από την άλλη πλευρά όμως, οι κλάσεις Item και Link θα πρέπει να είναι παιδιά της AuthoredObject. Έτσι, αν επιτρέπεται η πολλαπλή κληρονομικότητα, θα έχουμε την ιεραρχία κλάσεων



Αν, αντίθετα, επιτρέπεται μόνο η απλή κληρονομικότητα, επειδή η προηγούμενη ιεραρχία κλάσεων δεν μπορεί να μετατραπεί σε δέντρο, θα χρειαστεί να την αλλάξουμε. Μια πιθανή λύση είναι να αφήσουμε την ιεραρχία κλάσεων όπως ήταν πριν την εισαγωγή της κλάσης VersionedObject και να περάσουμε τις μεταβλητές και τις μεθόδους της κλάσης αυτής σε όλα τα παιδιά της, δηλαδή εδώ στις κλάσεις Item και View. Έτσι, θα έχουμε την επόμενη ιεραρχία:



Για τη συνέχεια θα θεωρήσουμε ότι έχουμε πολλαπλή κληρονομικότητα και κατά συνέπεια σχηματίζεται η πρώτη ιεραρχία κλάσεων.

Α.2. Κατάλογος μεταβλητών και μεθόδων

Στην παράγραφο αυτή απαριθμούνται οι μεταβλητές και οι μέθοδοι κάθε κλάσης του μοντέλου δεδομένων. Οι κλάσεις αναφέρονται με τη σειρά που δίνει η προδιατεταγμένη σάρωση του ακυκλικού γράφου. Έτσι, ορίζουμε με τη σειρά τις κλάσεις Object, Author, AuthoredObject, VersionedObject, Link, View, Item, Text, Graphics, Image, Audio, Video. Οι μεταβλητές και μέθοδοι που κληρονομούνται αναφέρονται για απλότητα μόνο στην κλάση που τις ορίζει για πρώτη φορά.

Κλάση Object
μεταβλητές
ObjectId
μέθοδοι
Retrieve(ObjectId)
Save()
Delete()
GetObjectId()

Κλάση Author
μεταβλητές
οι μεταβλητές της Object
Name
Password
μέθοδοι
οι μέθοδοι της Object
Create(Name,Password)
GetName()
SetName(NewName)
GetPassword()
SetPassword(NewPassword)
Present(WinWidth,WinHeight,...)

Κλάση AuthoredObject
μεταβλητές
οι μεταβλητές της Object
AuthorId
TimeStamp
μέθοδοι
οι μέθοδοι της Object
GetAuthorId()
GetTimeStamp()
SetAuthorId(NewAuthorId)
SetTimeStamp(NewTimeStamp)

Κλάση VersionedObject
μεταβλητές
οι μεταβλητές της Object
PreviousVersion
NextVersion
μέθοδοι
οι μέθοδοι της Object
GetVersion(Criteria,ObjectId)
DeleteVersion(Criteria,ObjectId)
ComputeDelta(OldValues, NewValues)

Κλάση Link
μεταβλητές
οι μεταβλητές της Object
οι μεταβλητές της AuthoredObject
Source
Target
Type
MirrorType
Weight
μέθοδοι
οι μέθοδοι της Object
οι μέθοδοι της AuthoredObject
Create(AuthorId,TimeStamp,
Source,Target,Type,MirrorType,
Weight)
GetSource()
GetTarget()
GetType()
GetMirrorType()
SetTypes(NewType,NewMirrorType)
GetWeight()
SetWeight(NewWeight)
Present(WinWidth,WinHeight,...)

Κλάση View
μεταβλητές
οι μεταβλητές της Object
οι μεταβλητές της VersionedObject
Name
Password
Description
IncItems
ExcItems
CriteriaList
μέθοδοι
οι μέθοδοι της Object
οι μέθοδοι της VersionedObject
Create(Name,Password,
Description,IncItems,ExcItems,
CriteriaList)
GetName()
SetName(NewName)
GetDescription()
SetDescription(NewDescription)
GetPassword()
SetPassword(NewPassword)
GetIncItems()
AddIncItem(ItemId)
DelIncItem(ItemId)
GetExcItems()
AddExcItem(ItemId)
DelExcItem(ItemId)
GetCriteriaList()
AddCriterion(Criterion)
DelCriterion(Criterion)
Present(WinWidth,WinHeight,...)

GetHome()
GetNextAND()
GetNextOR()

Κλάση Item

μεταβλητές
- οι μεταβλητές της Object
- οι μεταβλητές της AuthoredObject
- οι μεταβλητές της VersionedObject
- Description
- ContentType
- Properties
- AnchoredLinks

μέθοδοι
- οι μέθοδοι της Object
- οι μέθοδοι της AuthoredObject
- οι μέθοδοι της VersionedObject
- Create(AuthorId,TimeStamp, Description, Properties, ContentType)
- GetDescription()
- SetDescription(NewDescription)
- GetContentType()
- GetProperties()
- AddProperty(Property)
- DelProperty(Property)
- GetAnchoredLinks()
- AddAnchoredLink(LinkId,Anchor)
- DelAnchoredLink(LinkId)
- GetAnchor(LinkId)
- SetAnchor(LinkId,NewAnchor)

Κλάση Text

μεταβλητές
- οι μεταβλητές της Object
- οι μεταβλητές της AuthoredObject
- οι μεταβλητές της VersionedObject
- οι μεταβλητές της Item
- Content

μέθοδοι
- οι μέθοδοι της Object
- οι μέθοδοι της AuthoredObject
- οι μέθοδοι της VersionedObject
- οι μέθοδοι της Item
- GetContent()
- SetContent(NewContent)
- Present(WinWidth,WinHeight,...)

Κλάση Graphics

μεταβλητές
- οι μεταβλητές της Object
- οι μεταβλητές της AuthoredObject
- οι μεταβλητές της VersionedObject
- οι μεταβλητές της Item
- Content

μέθοδοι
- οι μέθοδοι της Object
- οι μέθοδοι της AuthoredObject
- οι μέθοδοι της VersionedObject
- οι μέθοδοι της Item
- GetContent()
- SetContent(NewContent)
- Present(WinWidth,WinHeight,...)

Κλάση Image

μεταβλητές
- οι μεταβλητές της Object
- οι μεταβλητές της AuthoredObject
- οι μεταβλητές της VersionedObject
- οι μεταβλητές της Item
- Content

μέθοδοι
- οι μέθοδοι της Object
- οι μέθοδοι της AuthoredObject
- οι μέθοδοι της VersionedObject
- οι μέθοδοι της Item
- GetContent()
- SetContent(NewContent)
- Present(WinWidth,WinHeight,...)

Κλάση Audio

μεταβλητές
- οι μεταβλητές της Object
- οι μεταβλητές της AuthoredObject
- οι μεταβλητές της VersionedObject
- οι μεταβλητές της Item
- Content

μέθοδοι
- οι μέθοδοι της Object
- οι μέθοδοι της AuthoredObject
- οι μέθοδοι της VersionedObject
- οι μέθοδοι της Item
- GetContent()
- SetContent(NewContent)
- Present(WinWidth,WinHeight,...)

Κλάση Video

μεταβλητές
- οι μεταβλητές της Object
- οι μεταβλητές της AuthoredObject
- οι μεταβλητές της VersionedObject
- οι μεταβλητές της Item
- Content

μέθοδοι
οι μέθοδοι της Object
οι μέθοδοι της AuthoredObject
οι μέθοδοι της VersionedObject
οι μέθοδοι της Item
GetContent()
SetContent(NewContent)
Present(WinWidth,WinHeight,...)

Β. BNF ΟΡΙΣΜΟΣ ΜΙΑΣ ΓΛΩΣΣΑΣ ΧΕΙΡΙΣΜΟΥ ΔΕΔΟΜΕΝΩΝ

Β.1. Επεξεργασία δεδομένων με την HYQL

Στο παράρτημα αυτό παρουσιάζεται ο BNF ορισμός της HYQL (Hypermedia Query Language), η οποία είναι μια *δηλωτική γλώσσα χειρισμού δεδομένων* (declarative data manipulation language, συντομ. DDML), σχεδιασμένη για να χρησιμοποιηθεί μέσα από το user interface μιας hypermedia βάσης δεδομένων.

Η HYQL δίνει τη δυνατότητα να δημιουργηθούν, να ανακτηθούν, να τροποποιηθούν, να διαγραφούν και να παρουσιαστούν αντικείμενα, και ακόμα να δηλώνουν οι χρήστες ποιες όψεις θα χρησιμοποιήσουν, να (απο)ενεργοποιούν τους ορισμούς των όψεων αυτών, να ζητάνε βοήθεια από το σύστημα και να δηλώνουν ότι θέλουν να αποσυνδεθούν.

Εφόσον πρόκειται για μια γλώσσα δηλωτική, κεντρικό ρόλο στην HYQL παίζουν τα *κριτήρια* (criteria), δηλαδή οι λογικές συνθήκες τις οποίες μπορούν να ικανοποιούν ή όχι τα αντικείμενα της βάσης δεδομένων. Τα κριτήρια που παρέχει η HYQL είναι τα γενικότερα δυνατά, μπορούν να είναι φωλιασμένα το ένα μέσα στο άλλο, και σε πρώτο επίπεδο έχουν πάντα τη μορφή μιας *διάζευξης συζεύξεων* (disjunction of conjuctions), ώστε να μπορεί να γίνει παράλληλη και, ενδεχομένως, βέλτιστη, αποτίμηση. Τα κριτήρια της HYQL αναφέρονται είτε σε ιδιότητες αντικειμένων μιας συγκεκριμένης κλάσης, είτε σε ιδιότητες αντικειμένων πολλών κλάσεων τα οποία συσχετίζονται μεταξύ τους μέσω κάποιων μεταβλητών (π.χ. σύνδεσμοι που συσχετίζονται με συγγραφείς, όψεις που συσχετίζονται με items, κλπ.).

Για να γίνει οποιαδήποτε λειτουργία (δηλαδή δημιουργία, ανάκτηση, τροποποίηση, διαγραφή ή παρουσίαση) πάνω σε αντικείμενα οποιασδήποτε κλάσης, πρέπει ο χρήστης να σχηματίσει ένα κριτήριο το οποίο να ικανοποιούν τα αντικείμενα που τον ενδιαφέρουν και να επιλέξει τα αντικείμενα που ικανοποιούν το κριτήριο αυτό. Επειδή, ωστόσο, μπορεί το κριτήριο που σχηματίζεται να μην ικανοποιείται μόνο από τα αντικείμενα που ενδιαφέρουν αλλά και από άλλα, η HYQL παρέχει και μια *λίστα επικοινωνίας* (communication list), στην οποία μπορούν κατ' αρχήν να ανακτηθούν όλα τα αντικείμενα που ικανοποιούν ένα κριτήριο και μετά να επιλεγούν από αυτά όσα αντικείμενα ενδιαφέρουν (υποτίθεται ότι θα υπάρχει ένα μέσον, όπως ποντίκι, μενού, κλπ., με το οποίο θα μπορούν να μαρκαριστούν αντικείμενα από τη λίστα επικοινωνίας). Έτσι, η επεξεργασία δεδομένων μπορεί να γίνει με δύο τρόπους:

(1) Ο χρήστης σχηματίζει ένα κριτήριο το οποίο ικανοποιούν όλα τα αντικείμενα που τον ενδιαφέρουν και μόνο αυτά, και χρησιμοποιεί το κριτήριο αυτό μέσα στην εντολή HYQL που αντιστοιχεί στην επεξεργασία που θέλει. Έτσι δε γίνεται χρήση της λίστας επικοινωνίας.

(2) Ο χρήστης σχηματίζει ένα κριτήριο το οποίο ικανοποιούν τουλάχιστον όλα τα αντικείμενα που τον ενδιαφέρουν, και χρησιμοποιεί το κριτήριο αυτό σε μια εντολή HYQL για ανάκτηση. Αφού τα αντικείμενα που ενδιαφέρουν ανακτηθούν και οι προσδιοριστές τους μπουν στη λίστα επικοινωνίας, ο χρήστης σημαδεύει όσα αντικείμενα της λίστας τον ενδιαφέρουν και δίνει την εντολή HYQL που αντιστοιχεί στην επεξεργασία που θέλει με αναφορά στα αντικείμενα αυτά.

Η λίστα επικοινωνίας θα ανανεώνεται έτσι κι αλλιώς μετά από κάθε ανάκτηση αντικειμένων και θα περιέχει τουλάχιστον τους προσδιοριστές των αντικειμένων που ανακτώνται. Αν κάποιο από τα αντικείμενα των οποίων οι προσδιοριστές υπάρχουν στη λίστα επικοινωνίας σβήνεται, θα σβήνεται και το αντίστοιχο στοιχείο της λίστας για να διατηρείται η συνέπεια των πληροφοριών που βλέπει ο χρήστης. Αν κάποιο αντικείμενο δημιουργείται, θα προστίθεται αυτόματα ένα αντίστοιχο στοιχείο στη λίστα επικοινωνίας, διότι είναι πολύ πιθανό να θέλει ο χρήστης να αναφερθεί στο αντικείμενο αυτό στη συνέχεια.

Β.2. Χρήση των όψεων μέσα από την HYQL

Η HYQL επιτρέπει στους χρήστες να μη βλέπουν ολόκληρη τη hypermedia βάση δεδομένων, αλλά μόνο μια όψη της. Θα πρέπει να σημειωθεί, ότι αυτό αφορά μόνο δύο από τις τέσσερεις κλάσεις αντικειμένων της βάσης, δηλαδή τα items και τους συνδέσμους. Τις όψεις και τους συγγραφείς που περιέχονται στη βάση μπορούν οι χρήστες να τους ανακτήσουν πάντοτε, έστω κι αν δεν μπορούν να τους

χρησιμοποιήσουν ή να τους επεξεργαστούν.

Για να μην μπορεί ένας χρήστης να προσπελάζει με έμμεσο τρόπο κομμάτια της βάσης δεδομένων που βρίσκονται πέρα από τις όψεις που είναι εξουσιοδοτημένος να χρησιμοποιήσει, αλλά και για να δουλεύει το user interface όχι με ολόκληρη τη βάση δεδομένων, αλλά μόνο με το τμήμα της που θα χρησιμοποιηθεί από το συγκεκριμένο χρήστη, υποχρεώνονται οι χρήστες να δηλώσουν στην αρχή της εργασίας τους με μια ειδική εντολή της HYQL ποιες όψεις θα χρησιμοποιήσουν. Έτσι, κάθε χρήστης δηλώνει αρχικά ότι θα χρησιμοποιήσει όσες όψεις θέλει, δίνοντας, φυσικά, και τα αντίστοιχα συνθήματα, και στη συνέχεια μπορεί να χρησιμοποιήσει ή να αναφέρει μέσα σε κριτήρια κάθε μια από τις όψεις αυτές, καθώς και κάθε καινούρια όψη που δημιουργεί. Εννοείται, φυσικά, ότι, όσο πιο μικρό είναι το κομμάτι της βάσης δεδομένων που θα καλύπτουν οι όψεις που δηλώνονται, τόσο πιο γρήγορη θα είναι η απόκριση του συστήματος στις εντολές του χρήστη, αφού τόσο λιγότερα θα είναι τα αντικείμενα που πρέπει να εξεταστούν.

Εφόσον ο κάθε χρήστης έχει στη διάθεσή του κάποιες όψεις, μπορεί να διαλέξει με ποια όψη θα δουλεύει κάθε φορά., δηλαδή ποια όψη θα είναι ενεργοποιημένη (μόνο μία όψη μπορεί να είναι ενεργοποιημένη κάθε φορά). Αν ένας χρήστης θέλει να χρησιμοποιήσει την ένωση ή/και την τομή κάποιων όψεων, μπορεί να δημιουργήσει μια καινούρια όψη με αυτές τις ιδιότητες. Έτσι, μέσα από την HYQL μπορεί ο χρήστης να μπει σε μια όψη, δηλαδή να ενεργοποιήσει τον ορισμό της, ή να βγει από μια όψη, δηλαδή να αποενεργοποιήσει τον ορισμό της. Όταν ένας χρήστης μπαίνει σε μια όψη, υπάρχουν δύο ενδεχόμενα:

(1) αν είχε προηγουμένως ανακτήσει κάποιο item που ανήκει στην καινούρια όψη, παραμένει σε αυτό, ενώ

(2) αν είχε προηγουμένως ανακτήσει κάποιο item που δεν ανήκει στην καινούρια όψη ή δεν είχε ανακτήσει κανένα item, μεταφέρεται στο home item της καινούριας όψης.

Το ότι ένας χρήστης έχει, κάποια στιγμή, ενεργοποιήσει τον ορισμό μιας όψης σημαίνει πως τα κριτήρια που χρησιμοποιεί αποτιμώνται πλέον ως προς όλους τους συγγραφείς της βάσης hypermedia, ως προς τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και ως προς τα items και τους συνδέσμους μόνο της ενεργοποιημένης όψης. Έτσι εξασφαλίζεται το ότι μπορεί να γίνει ανάκτηση, τροποποίηση, διαγραφή και παρουσίαση μόνο των items και των συνδέσμων που υπάρχουν μέσα στις όψεις τις οποίες χρησιμοποιεί κάθε χρήστης.

Όταν ένας χρήστης δεν ενεργοποιεί από την αρχή τον ορισμό καμιάς όψης ή απενεργοποιεί έναν ορισμό και δεν ενεργοποιεί κανέναν άλλον, τότε θα πρέπει να θεωρείται ότι δεν έχει προσπέλαση σε κανένα item και σύνδεσμο της hypermedia βάσης δεδομένων (μια εναλλακτική λύση θα ήταν να προσπελάζει ολόκληρη τη βάση, έτσι, όμως, θα παραβιάζονταν οι έννοες της ασφάλειας και της καθοδήγησης για τις οποίες ορίζονται, γενικά, οι όψεις). Εδώ, ωστόσο, η απόφαση που λαμβάνεται εντάσσεται στη γενικότερη πολιτική διαχείρισης της βάσης των δεδομένων και είναι φυσικό να ποικίλλει σε διαφορετικές περιπτώσεις.

Β.3. Χειρισμός των εκδοχών μέσα από τη HYQL

Ο ορισμός της HYQL που παρουσιάζεται εδώ χειρίζεται εκδοχές μόνο για τα items της hypermedia βάσης δεδομένων. Στην περίπτωση που υποστηρίζονται εκδοχές και για άλλες κλάσεις αντικειμένων (για τους συνδέσμους, για παράδειγμα) ο ορισμός της γλώσσας μπορεί να επεκταθεί ανάλογα.

Τα κριτήρια που επιλέγουν items έχουν συνθήκες που αναφέρονται στο συγγραφέα και στη χρονική στιγμή της τροποποίησης κάθε item. Με τις συνθήκες αυτές μπορούμε έμμεσα να ζητήσουμε την τελευταία εκδοχή ενός item που έφτιαξε ένας συγκεκριμένος συγγραφέας, ή την τελευταία εκδοχή ενός item γενικά. Σε οποιοδήποτε κριτήριο επιλέγει items μπορεί να καθοριστεί αν ο χρήστης ζητά να εξεταστεί μόνο η τελευταία εκδοχή κάθε ενός από τα items της ενεργοποιημένης όψης, ή όλες οι εκδοχές κάθε ενός από τα items της ενεργοποιημένης όψης. Αν ζητείται να εξεταστεί μόνο η τελευταία εκδοχή κάθε item τότε, αν η τελευταία εκδοχή ενός item ικανοποιεί ένα κριτήριο επιστρέφεται αυτή, αλλιώς δεν επιστρέφεται καμία. Αν ζητείται να εξεταστούν όλες οι εκδοχές κάθε item τότε, αν κάποιες εκδοχές ενός item ικανοποιούν ένα κριτήριο επιστρέφεται η πιο πρόσφατη από αυτές, αλλιώς δεν επιστρέφεται καμία. Σε κάθε περίπτωση, από τη στιγμή που θα επιστραφεί μια εκδοχή ενός item είναι δυνατόν να ανακτηθούν και οι υπόλοιπες εκδοχές του, με κατάλληλα κριτήρια που αναφέρονται στο item αυτό.

Επίσης θα πρέπει να σημειωθεί το ότι, εφόσον όλες οι εκδοχές ενός item έχουν τον ίδιο προσδιοριστή,

από τη στιγμή που ένα item βρίσκεται σε μια όψη μπορούν να προσπελαστούν μέσα από την όψη και όλες οι εκδοχές του item αυτού, ακόμα και αν δεν ικανοποιούν καμιά συνθήκη που να τους επιτρέπει να βρίσκονται μέσα στην όψη.

Το γεγονός ότι υποστηρίζονται εκδοχές καθιστά απαραίτητη την περιοδική συλλογή απορριμμάτων, και έτσι η HYQL παρέχει εντολή η οποία ενεργοποιεί τη διαδικασία αυτή. Το ποια items και σύνδεσμοι θα σβηστούν κατά τη συλλογή απορριμμάτων θα καθορίζεται με κριτήρια τα οποία θα αφορούν (i) τη χρονική στιγμή της τελευταίας τροποποίησης, (ii) το πόσες εκδοχές επιτρέπεται να κρατάμε ανά item και (iii) το πόσες εκδοχές επιτρέπεται να κρατάμε ανά item και ανά συγγραφέα, με τον αλγορίθμου που περιγράφεται στο κεφάλαιο για την τήρηση εκδοχών.

Β.4. Συμβάσεις για το BNF ορισμό της HYQL

Για τα *διακριτικά* (tokens) του BNF ορισμού της HYQL ισχύουν οι ακόλουθες συμβάσεις:

(1) Οι *δεσμευμένες λέξεις* (reserved words) της HYQL γράφονται με ΚΕΦΑΛΑΙΑ.

(2) Οι *συντακτικές μεταβλητές* (syntax variables) των BNF κανόνων γράφονται με Κεφαλαίο αρχικό γράμμα.

(3) Οι *τιμές που καθορίζονται από το σύστημα* (system-specified values) και οι *τιμές που καθορίζονται από τους χρήστες* (user-specified values) γράφονται με πεζά.

Επιπλέον, οι BNF κανόνες που ακολουθούν, χρησιμοποιούν τα επόμενα *μετασύμβολα* (metasymbols):

(1) **:-** ενώνει την κεφαλή και το σώμα ενός κανόνα BNF και έχει την έννοια "προσδιορίζεται ως"

(2) **,** ενώνει με διάζευξη πολλούς όρους στην κεφαλή ενός BNF κανόνα (οι κανόνες με το ίδιο σώμα σημειώνονται, για συντομία, ως ένας)

(3) **;** τερματίζει έναν κανόνα BNF

(4) | ενώνει με αποκλειστική διάζευξη πολλούς όρους στο σώμα ενός BNF κανόνα (οι κανόνες με την ίδια κεφαλή σημειώνονται, για συντομία, ως ένας)

(5) /* ανοίγει ένα σχόλιο

(6) */ κλείνει ένα σχόλιο

(7) /* */ κενός όρος στο σώμα ενός BNF κανόνα (η κεφαλή του κανόνα δεν αντικαθίσταται με κανέναν όρο)

(8) ' ' περικλείουν μη αλφαριθμητικούς χαρακτήρες (συμπεριλαμβανομένων και των μετασυμβόλων της BNF), που πρέπει να εννοηθούν όπως είναι, χωρίς δηλαδή καμιά αποτίμηση.

Η εφαρμογή των BNF κανόνων καταλήγει σε εντολές HYQL που αποτελούνται από τις δεσμευμένες λέξεις της γλώσσας και από τιμές για ακέραιους, συμβολοσειρές, κλπ., που δίνονται είτε από το σύστημα είτε από τους χρήστες. Επειδή η ακριβής μορφή των τιμών αυτών εξαρτάται από πολλούς παράγοντες και δεν μπορεί να αποφασιστεί πριν από την υλοποίηση, εδώ οι τιμές απλώς αναφέρονται χωρίς να αναλύονται περισσότερο.

Β.5. BNF ορισμός της HYQL

```
/*********************************************************/
/* 1. Εντολές της HYQL                                   */
/*********************************************************/
Request :-
        Operation '.'
        /* όλες οι εντολές της HYQL τελειώνουν με μια τελεία (.) */
    ;

Operation :-
        ObjectOperation
        /* λειτουργία για το χειρισμό αντικειμένων της βάσης δεδομένων */
    | OtherOperation
        /* βοηθητική λειτουργία */
    ;
```

ObjectOperation :-
CREATE ObjectCreation
/* δημιουργεί ένα αντικείμενο */
| MODIFY ObjectModification
/* τροποποιεί ένα αντικείμενο */
| DELETE ObjectDeletion
/* διαγράφει ένα αντικείμενο */
| RETRIEVE ObjectRetrieval
/* ανακτά ένα αντικείμενο */
| PRESENT ObjectPresentation
/* παρουσιάζει ένα αντικείμενο */
;

OtherOperation :-
DECLARE '[' ListOfViews ']'
/* δηλώνει ποιες όψεις θα χρησιμοποιηθούν στη συνέχεια */
| ENTER VIEW OneViewCriterion
/* ενεργοποιεί τον ορισμό μιας όψης */
| EXIT VIEW
/* απενεργοποιεί τον ορισμό μιας όψης */
| COLLECT GARBAGE GarbageCriteria
/* ενεργοποιεί τη λειτουργία συλλογής απορριμμάτων */
| UNDO
/* αναιρεί τα αποτελέσματα της προηγούμενης εντολής */
| HELP
/* δίνει στο χρήστη βοήθεια */
| QUIT
/* αποσυνδέει το χρήστη από το σύστημα */
;

ListOfViews :-
SingleView
| SingleView ',' ListOfViews
/* λίστα με τις όψεις που δηλώνει ότι θα χρησιμοποιήσει ο χρήστης */
;

SingleView :-
ViewName '/' ViewPassword
/* για κάθε όψη που δηλώνει ο χρήστης πρέπει να δώσει το όνομα και το σύνθημά της */
;

GarbageCriteria :-
MODIFIED BEFORE TimeStamp KEEPING AT MOST Number VERSIONS PER ITEM AND AT MOST Number VERSIONS PER AUTHOR PER ITEM
;

ObjectCreation :-
AUTHOR WITH AuthorCreateSpecs
/* μόνο ένας συγγραφέας μπορεί να δημιουργείται κάθε φορά */
| ITEM WITH ItemCreateSpecs
/* μόνο ένα item μπορεί να δημιουργείται κάθε φορά */
| LINK WITH LinkCreateSpecs
/* μόνο ένας σύνδεσμος μπορεί να δημιουργείται κάθε φορά */
| VIEW WITH ViewCreateSpecs

```
        /* μόνο μια όψη μπορεί να δημιουργείται κάθε φορά */
    ;

ObjectModification :-
        AUTHORS AuthorCriteria AuthorModifySpecs
        /* τα χαρακτηριστικά πολλών συγγραφέων μπορούν να τροποποιηθούν με την ίδια λειτουργία */
    | ITEMS ItemCriteria ItemModifySpecs
        /* τα χαρακτηριστικά πολλών items μπορούν να τροποποιηθούν με την ίδια λειτουργία */
    | LINKS LinkCriteria LinkModifySpecs
        /* τα χαρακτηριστικά πολλών συνδέσμων μπορούν να τροποποιηθούν με την ίδια λειτουργία */
    | VIEW OneViewCriterion ViewModifySpecs
        /* τα χαρακτηριστικά μόνο μιας όψης μπορούν να τροποποιηθούν με την ίδια λειτουργία */
    ;

ObjectDeletion :-
        AUTHOR OneAuthorCriterion
        /* μόνο ένας συγγραφέας μπορεί να διαγραφεί με μια λειτουργία */
    | ITEM OneItemCriterion
        /* μόνο ένα item μπορεί να διαγραφεί με μια λειτουργία */
    | LINK LinkCriterion
        /* ένας ή πολλοί σύνδεσμοι μπορούν να διαγραφούν με μια λειτουργία */
    | VIEW OneViewCriterion
        /* μόνο μια όψη μπορεί να διαγραφεί με μια λειτουργία */
    ;

ObjectRetrieval :-
        AUTHORS AuthorCriteria
    | ITEMS ItemCriteria
    | LINKS LinkCriteria
    | VIEWS ViewCriteria
        /* πολλά αντικείμενα της βάσης δεδομένων που ανήκουν στην ίδια κλάση μπορούν να ανακτηθούν
        με την ίδια λειτουργία */
    ;

ObjectPresentation :-
        AUTHOR '[' NumList ']'
    | ITEM '[' NumList ']'
    | LINK '[' NumList ']'
    | VIEW '[' NumList ']'
        /* Η λειτουργία παρουσίασης μπορεί να παρουσιάσει όσα αντικείμενα θέλει ο χρήστης, αρκεί να
        έχουν προηγουμένως ανακτηθεί αυτά και να υπάρχουν οι προσδιοριστές τους στη λίστα
        επικοινωνίας, οπότε μπορούν να επιλεγούν με τους σειριακούς αριθμούς που έχουν μέσα στη λίστα.
        */
    ;

NumList :-
        NumElement
    | NumElement ',' NumList
    ;

NumElement :-
        Number
    | Number '-' Number
        /* τα στοιχεία της λίστας μπορεί να είναι είτε αριθμοί είτε διαστήματα αριθμών */
```

```
    ;


/*********************************************************/
/*  2. Κριτήρια για αναφορά σε ένα ή πολλά αντικείμενα   */
/*  της βάσης δεδομένων                                  */
/*********************************************************/

AuthorCriteria :-
        AndedAuthorCriteria
    | AndedAuthorCriteria OR AuthorCriteria
    ;

AndedAuthorCriteria :-
        AuthorCriterion
    | AuthorCriterion AND AndedAuthorCriteria
        /* τα κριτήρια είναι μια διάζευξη από συζεύξεις συνθηκών */
    ;

AuthorCriterion :-
        OneAuthorCriterion
    | ManyAuthorsCriterion
        /* ένα κριτήριο μπορεί να αφορά έναν ή πολλούς συγγραφείς */
    ;

ItemCriteria :-
        AndedItemCriteria
    | AndedItemCriteria OR ItemCriteria
    ;

AndedItemCriteria :-
        ItemCriterion
    | ItemCriterion AND AndedItemCriteria
        /* τα κριτήρια είναι μια διάζευξη από συζεύξεις συνθηκών */
    ;

ItemCriterion :-
        OneItemCriterion
    | ManyItemsCriterion
        /* ένα κριτήριο μπορεί να αφορά ένα ή πολλά items */
    ;

LinkCriteria :-
        AndedLinkCriteria
    | AndedLinkCriteria OR LinkCriteria
    ;

AndedLinkCriteria :-
        LinkCriterion
    | LinkCriterion AND AndedLinkCriteria
        /* τα κριτήρια είναι μια διάζευξη από συζεύξεις συνθηκών */
    ;

LinkCriterion :-
```

```
        OneLinkCriterion
    | ManyLinksCriterion
        /* ένα κριτήριο μπορεί να αφορά έναν ή πολλούς συνδέσμους */
    ;

ViewCriteria :-
        AndedViewCriteria
    | AndedViewCriteria OR ViewCriteria
    ;

AndedViewCriteria :-
        ViewCriterion
    | ViewCriterion AND AndedViewCriteria
        /* τα κριτήρια είναι μια διάζευξη από συζεύξεις συνθηκών */
    ;

ViewCriterion :-
        OneViewCriterion
    | ManyViewsCriterion
        /* ένα κριτήριο μπορεί να αφορά μία ή πολλές όψεις */
    ;

OneAuthorCriterion :-
        SingleAuthorCriterion
    | '(' SingleAuthorCriterion ')'
        /* ένα κριτήριο για ένα συγγραφέα μπορεί να κλείνεται μέσα σε παρενθέσεις ή όχι */
    ;

SingleAuthorCriterion :-
        HAVING ID '=' AuthorId
    | HAVING NAME '=' AuthorName
    | BEING AUTHOR OF ITEM OneItemCriterion
    | BEING AUTHOR OF LINK OneLinkCriterion
        /* ένας συγγραφέας προσδιορίζεται μοναδικά με τον προσδιοριστή ή με το όνομά του, ή ως
        συγγραφέας ενός μοναδικού item ή συνδέσμου */
    ;

OneItemCriterion :-
        SingleItemCriterion
    | '(' SingleItemCriterion ')'
        /* ένα κριτήριο για ένα item μπορεί να κλείνεται μέσα σε παρενθέσεις ή όχι */
    ;

SingleItemCriterion :-
        SICwithoutVersions
    | SICwithoutVersions LATEST VERSION
    | SICwithVersions
        /* ένα item προσδιορίζεται μοναδικά είτε χωρίς αναφορά σε εκδοχές, οπότε μπορούν να εξεταστούν
        μόνο η τελευταία ή όλες οι εκδοχές του, ή με αναφορά σε εκδοχές, οπότε θα εξεταστεί αναγκαστικά
        η συγκεκριμένη εκδοχή του στην οποία γίνεται αναφορά */
    ;

SICwithoutVersions :-
        HAVING ID '=' ItemId
```

```
    | BEING SOURCE OF LINK OneLinkCriterion
    | BEING TARGET OF LINK OneLinkCriterion
        /* ένα item προσδιορίζεται μοναδικά, χωρίς αναφορά σε εκδοχές, με τον προσδιοριστή του, ή ως
        αρχικό ή τελικό item ενός μοναδικού συνδέσμου */
    ;

SICwithVersions :-
        COMING BEFORE ITEM OneItemCriterion
    | COMING AFTER ITEM OneItemCriterion
    | COMING FIRST BEFORE ITEM OneItemCriterion
    | COMING LAST AFTER ITEM OneItemCriterion
        /* ένα item προσδιορίζεται μοναδικά ως προηγούμενη, επόμενη, πρώτη ή τελευταία εκδοχή ενός
        μοναδικού item */
    ;

OneLinkCriterion :-
        SingleLinkCriterion
    | '(' SingleLinkCriterion ')'
        /* ένα κριτήριο για ένα σύνδεσμο μπορεί να κλείνεται μέσα σε παρενθέσεις ή όχι */
    ;

SingleLinkCriterion :-
        HAVING ID '=' LinkId
        /* ένας σύνδεσμος προσδιορίζεται μοναδικά με τον προσδιοριστή του */
    ;

OneViewCriterion :-
        SingleViewCriterion
    | '(' SingleViewCriterion ')'
        /* ένα κριτήριο για μια όψη μπορεί να κλείνεται μέσα σε παρενθέσεις ή όχι */
    ;

SingleViewCriterion :-
        HAVING ID '=' ViewId
    | HAVING NAME '=' ViewName
        /* μια όψη προσδιορίζεται μοναδικά με τον προσδιοριστή ή με το όνομά της */
    ;

ManyAuthorsCriterion :-
        MultipleAuthorsCriterion
    | '(' MultipleAuthorsCriterion ')'
        /* ένα κριτήριο για πολλούς συγγραφείς μπορεί να κλείνεται μέσα σε παρενθέσεις ή όχι */
    ;

MultipleAuthorsCriterion :-
        BEING AUTHOR OF ITEM ManyItemsCriterion
        /* επιστρέφει τους συγγραφείς οποιωνδήποτε από τα items που βρίσκονται μέσα στην
        ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
    | BEING AUTHOR OF LINK ManyLinksCriterion
        /* επιστρέφει τους συγγραφείς οποιωνδήποτε από τους συνδέσμους που βρίσκονται μέσα στην
        ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
    | NOT AuthorCriterion
        /* επιστρέφει όλους τους συγγραφείς της βάσης δεδομένων που δεν ικανοποιούν το κριτήριο */
    | AVAILABLE
```

/* επιστρέφει όλους τους συγγραφείς που υπάρχουν στη βάση δεδομένων */
| MARKED
/* επιστρέφει όλους τους συγγραφείς που έχουν μαρκαριστεί στη λίστα επικοινωνίας, εφόσον η λίστα επικοινωνίας περιέχει προσδιοριστές συγγραφέων */
;

ManyItemsCriterion :-
MultipleItemsCriterion
| '(' MultipleItemsCriterion ')'
/* ένα κριτήριο για μπορεί να κλείνεται μέσα σε παρενθέσεις ή όχι */
;

MultipleItemsCriterion :-
HAVING TIMESTAMP RelOp TimeStamp
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και έχουν δημιουργηθεί μια αποδεκτή χρονική στιγμή */
| HAVING DESCRIPTION StringMatching
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και των οποίων περιγραφή ταιριάζει στο αποδεκτό πρότυπο */
| HAVING CONTENT TYPE RelOp ContentType
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και έχουν περιεχόμενο αποδεκτού τύπου */
| HAVING ATTRIBUTE Attribute
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και περιλαμβάνουν στις ιδιότητες τους τη συγκεκριμένη χαρακτηριστική με οποιαδήποτε τιμή */
| HAVING ATTRIBUTE Attribute WITH VALUE Value
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και περιλαμβάνουν στις ιδιότητές τους τη συγκεκριμένη χαρακτηριστική με τη συγκεκριμένη τιμή */
| AUTHORED BY AUTHOR AuthorCriterion
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και έχουν συγγραφέα οποιονδήποτε από τους συγγραφείς που ικανοποιούν το κριτήριο */
| BEING SOURCE OF LINK LinkCriterion
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και είναι αρχικά items οποιωνδήποτε από τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
| BEING TARGET OF LINK LinkCriterion
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και είναι τελικά items οποιωνδήποτε από τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
| INCLUDED IN VIEW ViewCriterion
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και εγκλείονται σε όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και ικανοποιούν το κριτήριο */
| EXCLUDED FROM VIEW ViewCriterion
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και αποκλείονται από όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και ικανοποιούν το κριτήριο */
| CONTAINED IN VIEW ViewCriterion
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και περιέχονται σε όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και ικανοποιούν το κριτήριο· εδώ "περιέχονται" σημαίνει "ικανοποιούν ένα κριτήριο επιλογής" ή "εγκλείονται" */
| NOT ItemCriterion
/* επιστρέφει όλα τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και δεν ικανοποιούν το κριτήριο */
| AVAILABLE
/* επιστρέφει όλα τα items της ενεργοποιημένης όψης */

| MARKED
/* επιστρέφει όλα τα items που έχουν μαρκαριστεί στη λίστα επικοινωνίας, εφόσον η λίστα επικοινωνίας περιέχει προσδιοριστές items */
;

StringMatching :-
RelOp String
| StringOp RegularExpression
/* το ταίριασμα συμβολοσειρών μπορεί να έχει είτε την έννοια της απλής ισότητας ή διαφοράς είτε την έννοια της παραγωγής από κανονικές εκφράσεις */
;

ManyLinksCriterion :-
MultipleLinksCriterion
| '(' MultipleLinksCriterion ')'
/* ένα κριτήριο για μπορεί να κλείνεται μέσα σε παρενθέσεις ή όχι */
;

MultipleLinksCriterion :-
HAVING TIMESTAMP RelOp TimeStamp
/* επιστρέφει όλους τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και έχουν δημιουργηθεί μια αποδεκτή χρονική στιγμή */
| HAVING TYPE Op Type
/* επιστρέφει όλους τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και έχουν αποδεκτό τύπο */
| HAVING MIRRORTYPE Op Type
/* επιστρέφει όλους τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και έχουν αποδεκτό συμμετρικό τύπο */
| HAVING WEIGHT RelOp Weight
/* επιστρέφει όλους τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και έχουν αποδεκτό βάρος */
| AUTHORED BY AUTHOR AuthorCriterion
/* επιστρέφει όλους τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και έχουν συγγραφέα οποιονδήποτε από τους συγγραφείς που ικανοποιούν το κριτήριο */
| EMANATING FROM ITEM ItemCriterion
/* επιστρέφει όλους τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και ξεκινάνε από οποιοδήποτε από τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
| POINTING TO ITEM ItemCriterion
/* επιστρέφει όλους τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και καταλήγουν σε οποιοδήποτε από τα items που βρίσκονται μέσα στην ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
| CONTAINED IN VIEW ViewCriterion
/* επιστρέφει όλους τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και περιέχονται μέσα σε όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και ικανοποιούν το κριτήριο˙ εδώ "περιέχονται" σημαίνει "ικανοποιούν ένα κριτήριο επιλογής" */
| NOT LinkCriterion
/* επιστρέφει όλους τους συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και δεν ικανοποιούν το κριτήριο */
| AVAILABLE
/* επιστρέφει όλους τους συνδέσμους της ενεργοποιημένης όψης */
| MARKED
/* επιστρέφει όλους τους συνδέσμους που έχουν μαρκαριστεί στη λίστα επικοινωνίας, εφόσον η λίστα επικοινωνίας περιέχει προσδιοριστές συνδέσμων */

```
    ;

ManyViewsCriterion :-
        MultipleViewsCriterion
    | '(' MultipleViewsCriterion ')'
        /* ένα κριτήριο για μπορεί να κλείνεται μέσα σε παρενθέσεις ή όχι */
    ;

MultipleViewsCriterion :-
        HAVING DESCRIPTION StringMatching
        /* επιστρέφει όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και των οποίων η
        περιγραφή ταιριάζει στο αποδεκτό πρότυπο */
    | INCLUDING ITEMS ItemCriterion
        /* επιστρέφει όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και εγκλείουν όλα τα
        items που βρίσκονται μέσα στην ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
    | EXCLUDING ITEMS ItemCriterion
        /* επιστρέφει όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και αποκλείουν όλα τα
        items που βρίσκονται μέσα στην ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
    | CONTAINING ITEMS ItemCriterion
        /* επιστρέφει όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και περιέχουν όλα τα
        items που βρίσκονται μέσα στην ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
    | CONTAINING LINKS LinkCriterion
        /* επιστρέφει όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και περιέχουν όλους τους
        συνδέσμους που βρίσκονται μέσα στην ενεργοποιημένη όψη και ικανοποιούν το κριτήριο */
    | NOT ViewCriterion
        /* επιστρέφει όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα και δεν ικανοποιούν το
        κριτήριο */
    | AVAILABLE
        /* επιστρέφει όλες τις όψεις που έχουν δηλωθεί ή δημιουργηθεί ενδιάμεσα */
    | MARKED
        /* επιστρέφει όλες τις όψεις που έχουν μαρκαριστεί στη λίστα επικοινωνίας, εφόσον η λίστα
        επικοινωνίας περιέχει προσδιοριστές όψεων */
    ;


/*********************************************************/
/*  3. Δημιουργία αντικειμένων                           */
/*********************************************************/

AuthorCreateSpecs :-
        NAME AuthorName ',' PASSWORD AuthorPassword
        /* κατά τη δημιουργία ενός συγγραφέα είναι απαραίτητο να δοθεί το όνομα και το σύνθημά του */
    ;

ItemCreateSpecs :-
        TYPE DUMMY ItemDescription
    | TYPE NonDummyItemType ItemContent ItemDescription ItemProperties
        /* κατά τη δημιουργία ενός item είναι απαραίτητο να δοθεί ο τύπος του περιεχομένου του και
        προαιρετικό να δοθεί η περιγραφή του, αν ο τύπος είναι dummy, ή το περιεχόμενο, η περιγραφή και
        οι ιδιότητές του, αν ο τύπος του δεν είναι dummy˙ κατά τη δημιουργία ενός item δεν υπάρχουν
        σύνδεσμοι που να ξεκινάνε από ή να καταλήγουν σε αυτό */
    ;

NonDummyItemType :-
```

```
        TEXT
    | GRAPHICS
    | IMAGE
    | AUDIO
    | VIDEO
        /* το περιεχόμενο ενός item μπορεί να είναι τύπου text, gra[hics, image, audio ή video */
    ;

ItemContent :-
        /* */
    | ',' CONTENT IN FILE FileName
        /* για ένα non-dummy item μπορεί να μη δίνεται από τη δημιουργία του περιεχόμενο (για να δοθεί
        αργότερα), ή να δίνεται περιεχόμενο με τη μορφή ενός ονόματος αρχείου που περιέχει δεδομένα του
        αντίστοιχου τύπου */
    ;

ItemDescription :-
        /* */
    | ',' DESCRIPTION Description
    | ',' DESCRIPTION AS IN ITEM OneItemCriterion
        /* η περιγραφή ενός item μπορεί να μην έχει, αρχικά, τιμή, ή να έχει μια τιμή, ή να είναι ίδια με την
        περιγραφή ενός άλλου item */
    ;

ItemProperties :-
        /* */
    | ',' PROPERTIES '[' PropertyList ']'
    | ',' PROPERTIES AS IN ITEM OneItemCriterion
        /* οι ιδιότητες ενός item μπορεί να μην ορίζονται, αρχικά, ή να ορίζονται, ή να είναι ίδιες με τις
        ιδιότητες ενός άλλου item */
    ;

PropertyList :-
        Property
    | Property ',' PropertyList
        /* η λίστα των ιδιοτήτων θα περιέχει τουλάχιστον μία ιδιότητα, εφόσον ορίζονται ιδιότητες */
    ;

Property :-
        '<' Attribute ':' Value '>'
        /* κάθε ιδιότητα ενός item θα είναι ένα ζευγάρι από μια χαρακτηριστική και μια τιμή */
    ;

LinkCreateSpecs :-
        TYPE LinkType ',' FROM ITEM LinkExtreme LinkAnchor TO ITEM LinkExtreme LinkAnchor
        LinkWeight
        /* κατά τη δημιουργία ενός συνδέσμου είναι υποχρεωτικό να οριστούν ο τύπος του, το αρχικό και
        το τελικό του item, ενώ είναι προαιρετικό να οριστούν οι άγκυρες του συνδέσμου μέσα στο αρχικό
        και στο τελικό του item και το βάρος του συνδέσμου */
    ;

LinkType :-
        SystemLinkType
    | UserLinkType
```

| UserLinkType MIRRORTYPE UserLinkType
| AS IN LINK OneLinkCriterion
/* ο τύπος ενός συνδέσμου μπορεί να είναι ένας από τους προκαθορισμένους τύπους που υποστηρίζει το σύστημα, ή ένας μονοκατευθυνόμενος ή αμφικατευθυνόμενος τύπος που ορίζει ο χρήστης, ή ίδιος με τον τύπο ενός άλλου συνδέσμου */
;

SystemLinkType :-
DUMMY_RELATIONSHIP
| COMMON_TOPIC
| OPPOSITE_OPINION
| SUPPORTS | SUPPORTED_BY
| REFERS_TO | REFERRED_FROM
| MORE_DETAILS | GENERAL_FRAME
| EXAMPLE
/* οι τύποι συνδέσμων που υποστηρίζει το σύστημα είναι ο τύπος dummy και non-dummy τύποι για μονοκατευθυνόμενους ή αμφικατευθυνόμενους συνδέσμους */
;

LinkExtreme :-
OneItemCriterion
| AS IN LINK OneLinkCriterion
/* το αρχικό ή τελικό item ενός συνδέσμου μπορεί να ορίζεται με ένα κριτήριο ή να είναι ίδιο με το αρχικό ή τελικό item ενός άλλου συνδέσμου */
;

LinkAnchor :-
/* */
| WITH ANCHOR NumList
/* η άγκυρα ενός συνδέσμου μέσα στο αρχικό ή τελικό του item μπορεί να είναι είτε μια λίστα από αριθμούς που ορίζει ένα συγκεκριμένο τμήμα των multimedia δεδομένων, ή να μην ορίζεται, οπότε ο σύνδεσμος συνδέεται με ολόκληρο το αντίστοιχο item */
;

LinkWeight :-
/* */
| ',' WEIGHT Weight
| ',' WEIGHT AS IN LINK OneLinkCriterion
/* το βάρος ενός συνδέσμου μπορεί να μην ορίζεται από την αρχή, ή να ορίζεται, ή να είναι ίδιο με το βάρος ενός άλλου συνδέσμου */
;

ViewCreateSpecs :-
NAME ViewName ',' PASSWORD ViewPassword IncludedItems ExcludedItems ViewDefinitionCriteria
/* κατά τη δημιουργία μιας όψης είναι υποχρεωτικό να δοθεί το όνομα και το σύνθημά της, και προαιρετικό να δοθούν λίστες με items που εγκλείονται ή αποκλείονται και κριτήρια επιλογής για items και συνδέσμους (οι χρήστες, ωστόσο, θα πρέπει να προσέχουν ώστε να μην ορίζονται κενές όψεις */
;

IncludedItems :-
/* */
| ',' INCLUDED ITEMS '[' ItemList ']'

```
    | ',' INCLUDED ITEMS AS IN VIEW OneViewCriterion
        /* σε μια όψη μπορεί να μην εγκλείονται items, ή να εγκλείονται κάποια items, ή να εγκλείονται τα
        ίδια items που εγκλείονται και σε μια άλλη όψη */
    ;

ExcludedItems :-
        /* */
    | ',' EXCLUDED ITEMS '[' ItemList ']'
    | ',' EXCLUDED ITEMS AS IN VIEW OneViewCriterion
        /* από μια όψη μπορεί να μην αποκλείονται items, ή να αποκλείονται κάποια items, ή να
        αποκλείονται τα ίδια items που αποκλείονται και από μια άλλη όψη */
    ;

ItemList :-
        ItemId
    | ItemId ',' ItemList
        /* μια λίστα από items θα περιέχει τουλάχιστον ένα προσδιοριστή */
    ;

ViewSelectionCriteria :-
        /* */
    | ',' CRITERIA '[' CriteriaList ']'
    | ',' CRITERIA AS IN VIEW OneViewCriterion
        /* για μια όψη μπορεί να μην ορίζονται κριτήρια επιλογής, ή να ορίζονται κάποια κριτήρια, ή να
        ορίζονται τα ίδια κριτήρια που ορίζονται και για μια άλλη όψη */
    ;

CriteriaList :-
        '<' ITEMS ItemCriteria '>'
    | '<' LINKS LinkCriteria '>'
    | '<' ITEMS ItemCriteria '> <' LINKS LinkCriteria '>'
        /* τα κριτήρια επιλογής που ορίζονται για μια όψη μπορεί να επιλέγουν items ή/και συνδέσμους */
    ;


/*********************************************************/
/*  4. Τροποποίηση αντικειμένων                          */
/*********************************************************/

    /* ΣΗΜΕΙΩΣΗ : Στους κανόνες που ακολουθούν η τροποποίηση ενός αντικειμένου ορίζεται, για
απλότητα, ως μια ακολουθία από τροποποιήσεις των μεταβλητών του αντικειμένου αυτού. Ο ορισμός
αυτός είναι υπερβολικά γενικός και επιτρέπει εντολές τροποποίησης που περιλαμβάνουν δύο ή
περισσότερες τροποποιήσεις της ίδιας μεταβλητής. Αυτές, ωστόσο, οι εντολές, που είναι συντακτικά
σωστές αλλά σημασιολογικά λανθασμένες, θα πρέπει να ανιχνεύονται από ένα εργαλείο που βρίσκεται σε
υψηλότερο επίπεδο από το συντακτικό αναλυτή (αυτή, άλλωστε, η τακτική υιοθετείται και σε
καθιερωμένες γλώσσες χειρισμού δεδομένων όπως η SQL). Όταν ανιχνεύεται σημασιολογικό λάθος μπορεί
να δίνεται προειδοποίηση στο χρήστη, ή η εντολή να αποτυγχάνει.*/


AuthorModifySpecs :-
        AuthorModifySpec
    | AuthorModifySpec AuthorModifySpec
        /* τουλάχιστον μία τροποποίηση πρέπει να γίνει, και μόνο δύο τροποποιήσεις είναι δυνατές εδώ (για
        το όνομα και το σύνθημα ενός συγγραφέα) */
```

```
    ;

AuthorModifySpec :-
        NameModify
    | PasswordModify
    ;

NameModify :-
        SETTING NAME NewName
    ;

PasswordModify :-
        SETTING PASSWORD FROM OldPassword TO NewPassword
        /* το παλιό σύνθημα ζητείται από το σύστημα για ασφάλεια */
    ;

ItemModifySpecs :-
        ItemModifySpec
    | ItemModifySpec ItemModifySpecs
        /* τουλάχιστον μία τροποποίηση πρέπει να γίνει */
    ;

ItemModifySpec :-
        ItemContentModify
    | ItemDescriptionModify
    | ItemTypeModify
    | ItemPropertiesModify
    ;

ItemContentModify :-
        EDITING CONTENT
        /* τροποποίηση του περιεχομένου ενός item με κλήση του κατάλληλου editor */
    | SETTING CONTENT AT FILE FileName
        /* αλλαγή του αρχείου στο οποίο βρίσκεται το περιεχόμενο ενός item· τα δεδομένα του νέου
        αρχείου πρέπει να είναι του ίδιου τύπου με το περιεχόμενο του item και το τελευταίο δεν μπορεί να
        είναι τύπου dummy */

ItemDescriptionModify :-
        SETTING DESCRIPTION Description
    | SETTING DESCRIPTION AS IN ITEM OneItemCriterion
        /* η περιγραφή ενός item μπορεί να πάρει μια καινούρια τιμή ή να γίνει ίδια με την περιγραφή ενός
        άλλου item */
    ;

ItemTypeModify :-
        SETTING TYPE NonDummyItemType
        /* ο τύπος του περιεχομένου ενός item μπορεί να αλλάξει μόνο αν ήταν αρχικά dummy και δεν
        μπορεί, φυσικά, να γίνει dummy μετά την αλλαγή */
    ;

ItemPropertiesModify :-
        SETTING PROPERTIES '[' PropertyList ']'
    | SETTING PROPERTIES AS IN ITEM OneItemCriterion
    | PropertiesOnOff
```

/* οι ιδιότητες ενός item μπορούν να αλλάξουν συνολικά, ή να γίνουν ίδιες με τις ιδιότητες ενός άλλου item, ή συγκεκριμένες ιδιότητες μπορούν να προστεθούν ή/και να διαγραφούν */
;

PropertiesOnOff :-
PropertyOnOff
| PropertyOnOff PropertiesOnOff
;

PropertyOnOff :-
ADDING PROPERTY Property
| DELETING PROPERTY Property
;

LinkModifySpecs :-
LinkModifySpec
| LinkModifySpec LinkModifySpec
/* τουλάχιστον μία τροποποίηση πρέπει να γίνει, και δύο μόνο τροποποιήσεις μπορούν να γίνουν εδώ (για τον τύπο και το βάρος του συνδέσμου) */
;

LinkModifySpec :-
LinkTypeModify
| LinkWeightModify
;

LinkTypeModify :-
SETTING TYPE LinkType
| SETTING TYPE LinkType MIRRORTYPE LinkType
/* ο τύπος ενός συνδέσμου μπορεί να αλλάζει σε έναν τύπο καθορισμένο από το σύστημα ή καθορισμένο από το χρήστη αλλά μονοκατευθυνόμενο, ή σε έναν τύπο καθορισμένο από το χρήστη και αμφικατευθυνόμενο */
;

LinkWeightModify :-
SETTING WEIGHT Weight
| SETTING WEIGHT AS IN LINK OneLinkCriterion
/* το βάρος ενός συνδέσμου μπορεί να πάρει μια καινούρια τιμή, ή να πάρει την τιμή που έχει το βάρος ενός άλλου συνδέσμου */
;

ViewModifySpecs :-
ViewModifySpec
| ViewModifySpec ViewModifySpecs
/* τουλάχιστον μία τροποποίηση πρέπει να γίνει */
;

ViewModifySpec :-
NameModify
| PasswordModify
| CriteriaModify
| IncItemsModify
| ExcItemsModify
/* μπορούν να αλλάξουν το όνομα ή το σύνθημα μιας όψης, τα εγκλειόμενα ή αποκλειόμενα items

και τα κριτήρια που επιλέγουν items ή/και συνδέσμους */
;

CriteriaModify :-
SETTING CRITERIA'[' CriteriaList ']'
| SETTING CRITERIA AS IN VIEW OneViewCriterion
| CriteriaOnOff
/* τα κριτήρια επιλογής μπορούν να αλλάξουν συνολικά, ή να γίνουν ίδια με τα κριτήρια επιλογής μιας άλλης όψης, ή συγκεκριμένα κριτήρια για items ή/και συνδέσμους μπορούν να προστεθούν ή να διαγραφούν */
;

CriteriaOnOff :-
CriterionOnOff
| CriterionOnOff CriteriaOnOff
;

CriterionOnOff :-
ADDING CRITERION FOR ITEMS ItemCriterion
| DELETING CRITERION FOR ITEMS ItemCriterion
| ADDING CRITERION FOR LINKS LinkCriterion
| DELETING CRITERION FOR LINKS LinkCriterion
;

IncItemsModify :-
SETTING INCLUDED ITEMS '[' ItemList ']'
| SETTING INCLUDED ITEMS AS IN VIEW OneViewCriterion
| IncItemsOnOff
/* η λίστα των εγκλειόμενων items μπορεί να αλλάξει συνολικά, ή να γίνει ίδια με τη λίστα των εγκλειόμενων items μιας άλλης όψης, ή συγκεκριμένα items μπορούν να προστεθούν ή να διαγραφούν */
;

IncItemsOnOff :-
IncItemOnOff
| IncItemOnOff IncItemsOnOff
;

IncItemOnOff :-
ADDING INCLUDED ITEM ItemId
| DELETING INCLUDED ITEM ItemId
;

ExcItemsModify :-
SETTING EXCLUDED ITEMS '[' ItemList ']'
| SETTING EXCLUDED ITEMS AS IN VIEW OneViewCriterion
| ExcItemsOnOff
/* η λίστα των αποκλειόμενων items μπορεί να αλλάξει συνολικά, ή να γίνει ίδια με τη λίστα των αποκλειόμενων items μιας άλλης όψης, ή συγκεκριμένα items μπορούν να προστεθούν ή να διαγραφούν */
;

ExcItemsOnOff :-
ExcItemOnOff

```
    | ExcItemOnOff ExcItemsOnOff
    ;

ExcItemOnOff :-
        ADDING EXCLUDED ITEM ItemId
    | DELETING EXCLUDED ITEM ItemId
    ;


/*********************************************************/
/*  5. Τελεστές, κανονικές εκφράσεις και τιμές           */
/*  που καθορίζονται από τους χρήστες ή το σύστημα       */
/*********************************************************/

Op :-
        '='
    | '!='
        /* τελεστές ισότητας/διαφοράς */
    ;

RelOp :-
        '<'
    | '>'
    | '<='
    | '>='
    | Op
        /* τελεστές σύγκρισης */
    ;

StringOp :-
        '@'
        /* τελεστής που δηλώνει το ταίριασμα μιας συμβολοσειράς σε μια κανονική έκφραση που παράγει
        συμβολοσειρές */
    ;

RegExpr :-
        Character
    | RegExpr RegExpr
    | RegExpr ',' RegExpr
    | '(' RegExpr ')*'
        /* μια κανονική έκφραση μπορεί να δημιουργείται από χαρακτήρες, από παράθεση κανονικών
        εκφράσεων, από διάζευξη κανονικών εκφράσεων ή από τον αστερίσκο Kleene κάποιας κανονικής
        έκφρασης */
    ;

Character :-
    'A'  |  'B'  |  'C'  |  'D'  |  'E'  |  'F'  |  'G'
         |  'H'  |  'I'  |  'J'  |  'K'  |  'L'  |  'M'
         |  'N'  |  'O'  |  'P'  |  'Q'  |  'R'  |  'S'
         |  'T'  |  'U'  |  'V'  |  'W'  |  'X'  |  'Y'
         |  'Z'  |  'a'  |  'b'  |  'c'  |  'd'  |  'e'
         |  'f'  |  'g'  |  'h'  |  'i'  |  'j'  |  'k'
         |  'l'  |  'm'  |  'n'  |  'o'  |  'p'  |  'q'
         |  'r'  |  's'  |  't'  |  'u'  |  'v'  |  'w'
```

| 'x' | 'y' | 'z' | '0' | '1' | '2'
| '3' | '4' | '5' | '6' | '7' | '8'
| '9' | '0' | '~' | '`' | '!' | '@'
| '#' | '$' | '%' | '^' | '&' | '['
| ']' | '{' | '}' | '<' | '>' | '.'
| '?' | '/' | '\' | '|' | '+' | ' '
| ';' | ':' | '"' | '\"' | '-' | '='
| <space>

/* οι παραπάνω είναι οι χαρακτήρες που μπορούν να συμμετέχουν στις συμβολοσειρές που παράγονται από τις κανονικές εκφράσεις˙ στους χαρακτήρες αυτούς δεν περιλαμβάνονται οι μεταχαρακτήρες ( ) , και * που χρησιμοποιούνται για να ορίσουν τη δομή μιας κανονικής έκφρασης */

;

AuthorId , **ItemId** , **LinkId** , **ViewId** :-

SystemSpecifiedIdentifier

/* τιμές που δίνονται από το σύστημα˙ η ακριβής τους μορφή θα καθοριστεί κατά την υλοποίηση */

;

TimeStamp :-

SystemSpecifiedTimeStamp

/* τιμή που δίνεται από το σύστημα˙ η ακριβής της μορφή θα καθοριστεί κατά την υλοποίηση */

;

AuthorName , **ViewName** , **NewName** , **AuthorPassword** , **ViewPassword** , **OldPassword** , **NewPassword** , **String** , **Description** , **Attribute** , **Value** , **FileName** :-

UserSpecifiedString

/* τιμές που δίνονται από τους χρήστες˙ η ακριβής τους μορφή θα καθοριστεί κατά την υλοποίηση */

;

ContentType , **Type** , **UserLinkType** :-

UserSpecifiedType

/* τιμές που δίνονται από τους χρήστες˙ η ακριβής τους μορφή θα καθοριστεί κατά την υλοποίηση */

;

Weight , **Number** :-

UserSpecifiedNumber

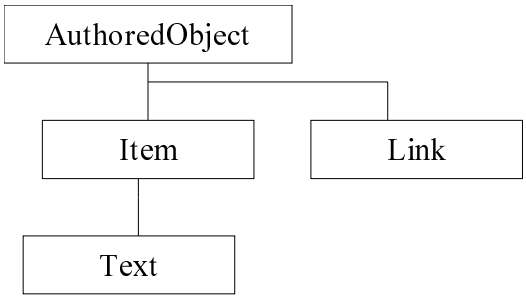
/* τιμές που δίνονται από τους χρήστες˙ η ακριβής τους μορφή θα καθοριστεί κατά την υλοποίηση */

;

Γ. ΥΛΟΠΟΙΗΣΗ ΠΡΩΤΟΤΥΠΟΥ ΓΙΑ ΜΙΑ HYPERMEDIA ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Γ.1. Εισαγωγή

Ο κώδικας που παρατίθεται στο παράρτημα αυτό υλοποιεί ένα περιορισμένο τμήμα της βάσης δεδομένων που έχουμε περιγράψει. Πιο συγκεκριμένα, υλοποιούνται μόνο οι κλάσεις AuthoredObject, Item, Link και Text, των οποίων η ιεραρχία φαίνεται στο πιο κάτω σχήμα:



Η κλάση VersionedObject δεν έχει υλοποιηθεί ως ξεχωριστή κλάση, αλλά οι λειτουργίες της υλοποιούνται μέσα στην κλάση Item, γιατί μόνο τα αντικείμενα της κλάσης αυτής απαιτούν τήρηση εκδοχών. Στην τήρηση εκδοχών δε χρησιμοποιείται καμία τεχνική συμπίεσης των δεδομένων, και δεν υποστηρίζεται προς το παρόν η απομάκρυνση εκδοχών ούτε η συλλογή απορριμμάτων, ενώ χρησιμοποιούνται ειδικές μέθοδοι για αναζήτηση κάποιας εκδοχής βάσει κριτηρίων, δηλαδή η μέθοδος GetVersion(Criteria, ObjectId) υλοποιείται μέσω πιο εξειδικευμένων μεθόδων. Όλα τα δεδομένα αποθηκεύονται σε συμβατικό δίσκο, αφού δεν ήταν διαθέσιμο κατά την ανάπτυξη του πρωτοτύπου, υλικό multimedia αποθηκευμένο σε οπτικό μέσο. Η μεταβλητή *id* έχει μεταφερθεί από την κλάση Object, που βρισκόταν στην κορυφή της ιεραρχίας των κλάσεων, στις κλάσεις Item και Link, με αποτέλεσμα τα αντικείμενα διαφορετικών κλάσεων να αποθηκεύονται σε διαφορετικά αρχεία. Αυτό έγινε γιατί η C++ είναι γλώσσα ισχυρών τύπων και έτσι αίτηση για ανάκτηση σε αντικείμενο τύπου item ενός προσδιοριστή που αντιστοιχεί σε αντικείμενο τύπου link (ή αντίστροφα) (δηλαδή μία αίτηση του τύπου "anItem.Retrieve(anId)", όπου το αντικείμενο anItem έχει δηλωθεί ως μέλος της κλάσης item και το αντικείμενο με προσδιοριστή "anId" ανήκει στην κλάση Link), θα είχε απρόβλεπτα αποτελέσματα. Από τις υποκλάσεις του Item, μόνο η κλάση Text έχει υλοποιηθεί, και αυτό γιατί δεν υπήρχαν πακέτα επεξεργασίας άλλων τύπων δεδομένων στο σύστημα όπου αναπτύχθηκε ο κώδικας. Για την επεξεργασία των κειμένων χρησιμοποιήθηκε ο editor "vi" του Unix. Στο επίπεδο του user interface υποστηρίζεται μόνο η περιπλάνηση και όχι ο *δηλωτικός* (declarative) τρόπος επεξεργασίας των δεδομένων. Ο κώδικας αναπτύχθηκε σε C++ σε σταθμό εργασίας Sun και κάτω από το λειτουργικό σύστημα Unix.

Γ.2. Μοντέλο δεδομένων και μέθοδοι

Για κάθε μια από τις προαναφερθείσες κλάσεις ορίζονται οι μεταβλητές και μέθοδοι που αναφέρονται πιο κάτω. Θα πρέπει να σημειωθεί ότι σε αρκετές περιπτώσεις τα αποτελέσματα που πρέπει να επιστρέφουν οι μέθοδοι εμφανίζονται ως παράμετροι. Αυτό συμβαίνει γιατί στη C++, όπως και στη C, δεν μπορούν να οριστούν συναρτήσεις που να επιστρέφουν δεδομένα τύπου διαφορετικού από αυτούς που εξ αρχής υπάρχουν στη γλώσσα.

Κλάση AuthoredObject

μεταβλητές:

authorId
timeStamp

μέθοδοι:
AuthoredObject()
GetAuthorId()
SetAuthorId(newAuthorId)
GetTimeStamp()
SetTimeStamp(NewTimeStamp)
Size()
PlaceToBuffer(buffer)
RetrieveFromBuffer(buffer)

Η μέθοδος AuthoredObject(), και γενικότερα οι μέθοδοι που το όνομά τους είναι ίδιο με το όνομα της κλάσης στην οποία ορίζονται, ονομάζεται *κατασκευαστής* (constructor), και σκοπός της είναι να κάνει *αρχικοποίηση* (initialization) των στιγμιοτύπων της κλάσης αυτής. Αντίστοιχα μπορεί να υπάρχει και η μέθοδος *καταστροφέας* (destructor) που καλείται όταν ένα αντικείμενο παύει να υπάρχει και συμβολίζεται με τον χαρακτήρα "~" ακολουθούμενο από το όνομα της κλάσης του αντικειμένου αυτού. Οι μέθοδοι Size(), PlaceToBuffer() και RetrieveFromBuffer() χρησιμοποιούνται για την αποθήκευση και ανάκτηση των αντικειμένων από το δίσκο, και επανορίζονται σε κάθε κλάση για την οποία, ή για τις κλάσεις-παιδιά της οποίας, της οποίας, πρέπει να αποθηκευτούν και να ανακτηθούν κάποια στιγμιότυπα. Τέλος σημειώνεται πως η ονοματολογία στον κώδικα διαφέρει από αυτήν που χρησιμοποιείται εδώ στο ότι χρησιμοποιείται ο χαρακτήρας "_" (underscore) για να χωρίσει τις λέξεις που αποτελούν το όνομα κάποιας μεταβλητής ή μεθόδου, ενώ εδώ χρησιμοποιούνται κεφαλαία.

Κλάση Link

μεταβλητές:
id
source
target
type
mirrorType
weight

μέθοδοι:
Create(authorId,source,target,type,mirrorType,weight)
Delete()
Size()
PlaceToBuffer(buffer)
RetrieveFromBuffer(buffer)
Save()
Retrieve(id)
Display()
ReturnLinkId()
ReturnSource()
ReturnTarget()
ReturnType()
ReturnMirrorType()
ReturnWeight()

Κλάση Item

μεταβλητές:
- id
- prevVersionPtr
- prevVersionSize
- nextVersionPtr
- nextVersionSize
- description
- contentType
- properties
- anchoredLinks

μέθοδοι:
- Item()
- ~Item()
- Create(author,description,contentType)
- Retrieve(id)
- RetrieveOfAuthor(id,author)
- RetrieveWithLinkId(itemId,linkId)
- GetPreviousVersion()
- GetPreviousVersionOfAuthor(author)
- Save()
- Delete()
- GetId()
- GetDescription()
- GetProperties()
- GetAnchoredLinks()
- HasLinkId(linkId)
- GetLinkAnchor(linkId,anchorStart,anchorLength)
- SetDescription(newDescription)
- addProperty(attribute,value)
- dropProperty(attribute,value)
- addAnchoredLink(linkId,anchorStart,anchorLength)
- dropAnchoredLink(linkId)
- GetContentType()
- PlaceToBuffer(buffer)
- RetrieveFromBuffer(buffer)
- Size()

Κλάση Text

μεταβλητές:
- data

μέθοδοι:
- Text()
- ~Text()
- GetData()
- SetData(newData)
- Delete()
- PlaceToBuffer(buffer)
- RetrieveFromBuffer(buffer)
- Size()

Γ.3. Το σύστημα αρχείων

Για την αποθήκευση όλων των αντικειμένων χρησιμοποιούνται τρία αρχεία και πιο συγκεκριμένα το αρχείο "linkfile", το αρχείο "itemfile" και το αρχείο "datafile". Στο αρχείο "linkfile" αποθηκεύονται τα στιγμιότυπα της κλάσης link, και είναι οργανωμένα με κάποια μορφή άμεσης προσπέλασης πάνω στο πρωτεύον κλειδί, που είναι το πεδίο id. Τα κλειδιά δίνονται από το σύστημα κατά σειριακό τρόπο, ενώ λαμβάνεται πρόνοια να χρησιμοποιούνται ξανά τα κλειδιά των συνδέσμων που σβήνονται.

Για την αποθήκευση των στιγμιοτύπων της κλάσης Item και των υποκλάσεών της, χρησιμοποιούνται δύο αρχεία. Αυτό γίνεται γιατί, σε αντίθεση με τα στιγμιότυπα της κλάσης Link, τα στιγμιότυπα της κλάσης Item δεν έχουν σταθερό μέγεθος. Έτσι, στο αρχείο "datafile" αποθηκεύονται τα πραγματικά δεδομένα των items χωρίς καμία οργάνωση, και στο αρχείο "itemfile", που είναι οργανωμένο με κάποια μορφή άμεσης προσπέλασης, αποθηκεύονται πληροφορίες για το που βρίσκονται τα δεδομένα του κάθε item μέσα στο αρχείο "datafile" και τι μέγεθος έχουν. Η πρόσθεση μιας καινούριας εκδοχής ενός item επιτυγχάνεται με την πρόσθεση (appending) στο αρχείο "datafile" της περιγραφής της καινούριας εκδοχής του item και την αλλαγή στο αρχείο "itemfile" του δείκτη στα δεδομένα και του μεγέθους του εν λόγω item. Οι πληροφορίες για την προηγούμενη εκδοχή φυλάσσονται στις μεταβλητές PrevVersionPtr και PrevVersionSize της νέας εκδοχής. Όπως και για την κλάση Link, λαμβάνεται πρόνοια να χρησιμοποιούνται ξανά τα κλειδιά των items που σβήνονται.

Γ.4. Κώδικας

Ο κώδικας είναι χωρισμένος σε αρκετά αρχεία, για να είναι πιο εύκολη η κατανόησή του, μια και κάθε αρχείο περιλαμβάνει δηλώσεις ή/και μεθόδους που αφορούν κάποιο συγκεκριμένο τμήμα της εφαρμογής. Πιο αναλυτικά, υπάρχουν τα εξής αρχεία:

"item.c" : user interface και επεξεργασία δοσοληψιών
"text.h" : ορισμός της κλάσης Text
"item.h" : ορισμός της κλάσης Item
"link.h" : ορισμός της κλάσης Link
"authored.h" : ορισμός της κλάσης AuthoredObject
"system.h" : μέθοδοι για την αποθήκευση και ανάκτηση των items και μηχανισμός για τήρηση εκδοχών
"date.h" : μέθοδοι για το διάβασμα και την εκτύπωση της ημερομηνίας του συστήματος
"sequential.h" : μέθοδοι για το σειριακό διάβασμα των αντικειμένων
"mklinkfl.c", "mkitemfl.c", "mkdatafl.c" : κώδικας για την αρχικοποίηση των αρχείων της βάσης
"authcons.h", "filecons.h", "itemcons.h", "linkcons.h" : σταθερές που χρησιμοποιούνται σε διάφορα τμήματα της εφαρμογής.

Ακολουθεί ο κώδικας.

Αρχείο "item.c"

```
#include "text.h"
#include "sequent.h"
#include <string.h>

typedef struct __links_list
{
link1 link;
struct __links_list *next;
} link_list_node;
```

```
void call_vi(void **,int);
void free_links_list(link_list_node *);

char *selections[] =
{
        "1...Create Item",
        "2...Retrieve Item",
        "3...View description",
        "4...View Content",
        "5...View properties",
        "6...Edit Contents",
        "7...Edit description",
        "8...Add property",
        "9...Drop property",
        "0...Delete Current Item",
        "a...View links",
        "b...Add link",
        "c...Delete link",
        "d...View content type",
        "e...Get a previous version of current item",
        "f...Save current item",
        "g...View ids of all items",
        "h...Traverse link",
        "i...Exit",
        (char *)0
};


main()
{
int itempresent = 0,itemindb,itemmodified;
long transaction_status;
class text atext;
char buff[256],buff1[256],buff2,auid[256];
char prefered_author[256];
int selection;
char *dataptr, *descriptionptr;
int i,item_id;
property_node *p1;
link_list_node *links,*l1,*l2;
anchored_link_node *ln;
int content_type;
float weight;
int type,mirror_type;
LinkId_t link_id;
date version_date;

links = new link_list_node;
links->next = 0;

printf("Please enter your name : ");
gets(auid);
auid[MAXAUTHORNAME - 1] = '\0';
printf("Press <RETURN> : ");
```

```
gets(buff);

do
{
	system("clear");

	if (itempresent)
	{
		printf("Current item is: %d\n",atext.get_id());
		printf("Author is : %s\n",atext.get_author_id());
		printf("Time stamp is: ");
		atext.get_time_stamp(&version_date);
		print_date(version_date);
		printf("\n\n");
	}
	else
		printf("No current item\n\n");
	printf("Enter selection:\n");
	for(i=0;selections[i]!=(char *)0;i++)
		printf("\t%s\n",selections[i]);
	do
	{
		gets(buff);
		if (buff[0] >= 'a' && buff[0] <='i')
			selection = buff[0] - 'a' + 10;
		else
			if (buff[0] >= '0' && buff[0] <= '9')
				selection = buff[0] - '0';
			else
				selection = 127;
	}
	while (selection > 18 || selection < 0);
	system("clear");
	if (itempresent || selection == 1 || selection == 2 ||
		selection == 16 || selection == 18)
		switch (selection)
		{
			case 0: /*The "delete item" transaction */
				atext.Delete();
				atext.set_author_id(auid);
				itemmodified = itempresent = 1;
				break;

			case 1: /* The "create item" procedure */
				/* Get rid of any links that might be present */
				free_links_list(links);
				/*
					Save the current item (if such an item exists),
					if it has been modified
				*/
				if (itempresent && itemmodified)
				{
					date cur_date;
```

```
                get_system_date(&cur_date);
                atext.set_time_stamp(cur_date);
                atext.save();
        }
        /* Get the description */
        call_vi((void **)&descriptionptr,0);
        /* And the content type */
        printf("Enter content type : ");
        gets(buff1);
        content_type = atoi(buff1);
        /* Now create the item */
        atext.create(auid,descriptionptr,content_type);
        /* Now ask for some contents */
        call_vi((void **)&dataptr,0);
        atext.set_data(dataptr);
        itemmodified = itempresent = 1;
        itemindb = 0;
        break;

case 2: /* The "Retrieve item" transaction */
        /* Free links memory */
        free_links_list(links);
        /*
                Save the current item, if it has been modified
        */
        if ((itempresent != 0) && itemmodified)
        {
                date cur_date;

                get_system_date(&cur_date);
                atext.set_time_stamp(cur_date);
                atext.save();
        }
        printf("Enter id : ");
        gets(buff);
        item_id = atoi(buff);
        printf("Enter prefered author ");
        printf("(<RETURN> for none) :");
        gets(prefered_author);
        if (prefered_author[0] == 0) /*No preference*/
                transaction_status = atext.retrieve(item_id);
        else
                transaction_status=atext.retrieve_of_author
                (item_id,prefered_author);
        if (transaction_status == ITEMNOTFOUND)
        {
                /* Item not found */
                itempresent = 0;
                printf("Item not found.\n");
        }
        else
                if (transaction_status == NOSUCHVERSION)
                {
                        /*
```

```
                        The item was found, but the version
                        requested was not
                */
                itempresent = 0;
                printf("No version of this author found\n");
        }
        else
        {
                /*
                        Item found. Retrieve the links emerging
                        from it
                */
                itemindb = itempresent = 1;
                itemmodified = 0;
                for (ln=atext.get_anchored_links(),l1=links;
                        ln != 0; ln = ln->next,l1 = l1->next)
                {
                        l2 = new link_list_node;
                        l2->next = 0;
                        l1->next = l2;
                        (l2->link).Retrieve(ln->id);
                }
        }
    break;

case 3: /* Print the description */
    printf("%s\n",atext.get_description());
    break;

case 4: /* Print data */
    printf("%s\n",atext.get_data());
    break;

case 5 : /* Print properties */
    for(p1=atext.get_properties(); p1!=0; p1 = p1->next)
    {
        printf("Attribute : %s, Value : %s\n", p1->attribute,p1->value);
    }
    break;

case 6 : /* Edit data */
    itemmodified = 1;
    dataptr = atext.get_data();
    call_vi((void **)&dataptr,1);
    atext.set_data(dataptr);
    atext.set_author_id(auid);
    break;

case 7: /* Edit description */
    itemmodified = 1;
    atext.set_author_id(auid);
    descriptionptr = atext.get_description();
    call_vi((void **)&descriptionptr,1);
    atext.set_description(descriptionptr);
```

```
        break;

case 8: /* Add property */
        printf("Enter attribute : ");
        gets(buff);
        if (buff[0] == '\0')
        {
                printf("Zero length attributes not allowed\n");
                break;
        }
        printf("Enter value : ");
        gets(buff1);
        if (buff1[0] == '\0')
        {
                printf("Zero length values not allowed.\n");
                break;
        }
        if (atext.add_property(buff,buff1) != 0)
                printf("Property already exists.\n");
        else
        {
                printf("Property added.\n");
                itemmodified = 1;
        }
        break;

case 9: /* Drop property */
        printf("Enter attribute : ");
        gets(buff);
        printf("Enter value : ");
        gets(buff1);
        if (atext.drop_property(buff,buff1))
                printf("No such property.\n");
        else
        {
                printf("Property dropped.\n");
                itemmodified = 1;
        }
        break;

case 10: /* Print links */
        for (l1 = links->next;l1 != 0;l1=l1->next)
                (l1->link).display();
        break;

case 11: /* Add link transaction */
        /*
        Read the target name and check if it is
        identical to the current item's name or if
        that item does not exist. Both cases are
        error cases.
        */
        printf("Enter target items id : ");
        gets(buff);
```

```
                item_id = atoi(buff);
                if (item_id == atext.get_id())
                        printf("Cannot link an item to itself.\n");
                else
                        if (item_exists(item_id) == 0)
                        {
                                date cur_date;

                                get_system_date(&cur_date);
                                /*
                                        Now read the Type, mirror_Type and weight
                                        fields. Anchors are ignored
                                */
                                printf("Enter type : ");
                                gets(buff);
                                type = atoi(buff);
                                printf("Enter inverse type (%d for NULL) :", NULLTYPE);
                                gets(buff);
                                mirror_type = atoi(buff);
                                printf("Enter weight : ");
                                gets(buff);
                                weight = atof(buff);
                                l2 = new link_list_node;
                                /*
                                        Create the new link. Add its id to the
                                        current item's list of ids, and the whole
                                        link to the list of links.
                                */
                                (l2->link).Create(auid,atext.get_id(),
                                                item_id, type,mirror_type, weight);
                                (l2->link).set_time_stamp(cur_date);
                                l2->next = links->next;
                                links->next = l2;
                                atext.add_anchored_link((l2->link).ReturnLinkid(),0,0);
                                (l2->link).Save();
                                {
                                        class text temp_text;

                                        temp_text.retrieve(item_id);
                                        temp_text.add_anchored_link
                                        ((l2->link).ReturnLinkid(),0,0);
                                        temp_text.set_time_stamp(cur_date);
                                        temp_text.save();
                                }
                                itemmodified = 1;
                        }
                        else
                                printf("Target item does not exist.\n");
                break;

        case 12: /* The "delete link" transaction */
                printf("Enter id : ");
                gets(buff);
                link_id = atoi(buff);
```

```
        /*
                Delete the id from the current item's id list,
                and the link from the links list.
        */
        if (atext.drop_anchored_link(link_id) == 0)
        {
                for (l1=links->next,l2=links;l1 != 0; l1 = l1->next)
                {
                        if ((l1->link).ReturnLinkid() == link_id)
                        {
                                l2->next = l1->next;
                                delete l1;
                                break;
                        }
                        l2 = l1;
                }
                itemmodified = 1;
        }
        else
                printf("No such link id.\n");
        break;

case 13: /* Print content type */
        printf("Content type is : %d\n",
        atext.get_content_type());
        break;

case 14: /* Get a previous version */
/*
        If the current item has been modified, it will
        be first saved, and then the previous version
        will be retrieved. As an efect, you will end up
        with the version you last modified, if you
        don't specify an author.
*/
        if ( (itempresent != 0) && itemmodified)
        {
                date cur_date;

                get_system_date(&cur_date);
                atext.set_time_stamp(cur_date);
                atext.save();
                itemmodified = 0;
        }
        printf("Enter prefered author ");
        printf("or just <ENTER> for any author : ");
        gets(prefered_author);
        if (prefered_author[0] == '\0') /* Any author */
                transaction_status = atext.get_previous_version();
        else
        transaction_status = atext.get_previous_version_of_author(auid);
        if (transaction_status == NOSUCHVERSION)
                printf("Sorry, version not found.\n");
        else
```

```
                /*
                        Update the links list to match current version
                */
        {
                itemindb = itempresent = 1;
                itemmodified = 0;
                free_links_list(links);
                for (ln = atext.get_anchored_links(), l1 = links; ln != 0;
                        ln = ln->next,l1 = l1->next)
                {
                        l2 = new link_list_node;
                        l2->next = 0;
                        l1->next = l2;
                        (l2->link).Retrieve(ln->id);
                }
        }
        break;

case 15:
        {
                date cur_date;

                get_system_date(&cur_date);
                atext.set_time_stamp(cur_date);
                atext.save();
                itemmodified = 0;
                break;
        }

case 16:
        /* Print ids of all items in the database */
        initialize_search();
        while(get_next_id(&item_id) == SUCCESS)
                printf("%d\n",item_id);
        if (! itemindb && itempresent)
                printf("%d\n",atext.get_id());
        break;

case 17: /* Traverse link transaction */
        printf("Enter link id : ");
        gets(buff);
        link_id = atoi(buff);
        if (atext.has_link_id(link_id) != 0)
        {
                printf("No such link id in this item.\n");
                break;
        }
        for (l1 = links->next; ; l1 = l1->next)
                if ((l1->link).ReturnLinkid() == link_id)
                        break;
        item_id = (l1->link).ReturnTarget();
        if (item_id == atext.get_id())
        {
                if ((l1->link).Return_mirror_Type()==NULLTYPE)
```

```
                        {
                            printf("Untraversable in that direction.\n");
                            break;
                        }
                        item_id = (l1->link).ReturnSource();
                    }
                    free_links_list(links);
                    if ((itempresent !=0) && itemmodified)
                    {
                        date cur_date;

                        get_system_date(&cur_date);
                        atext.set_time_stamp(cur_date);
                        atext.save();
                    }
                    atext.retrieve_with_linkid(item_id,link_id);
                    itemindb = itempresent = 1;
                    itemmodified = 0;
                    for (ln = atext.get_anchored_links(),l1 = links;
                        ln != 0; ln=ln->next,l1=l1->next)
                    {
                        l2 = new link_list_node;
                        l2->next = 0;
                        l1->next = l2;
                        (l2->link).Retrieve(ln->id);
                    }
                    break;

                case 18: /* Exit */
                    if (itempresent && itemmodified)
                    {
                        date cur_date;

                        get_system_date(&cur_date);
                        atext.set_time_stamp(cur_date);
                        atext.save();
                    }
                    printf("So long %s\n",auid);
                    break;

                default :
                    printf("Something is wrong arround here...\n");
                    break;
            }

            if (selection != 18)
            {
                printf("Press <Return>");
                gets(buff);
            }
        }
        while(selection != 18);
        return 0;
}
```

```
#define EDITFILE ".TMPFILE.EDIT"

void call_vi(void **data, int datapresent)
/*
  Call vi to create or modify the contents or the
  description.
*/
{
  FILE *fp;
  long datalength;
  char command[255];

  fp = fopen(EDITFILE,"w");
  if (datapresent != 0)
        fprintf(fp,"%s",(char *)*data);
  fclose(fp);
  sprintf(command,"vi %s",EDITFILE);
  system(command);
  fp = fopen(EDITFILE,"r");
  fseek(fp,0,SEEK_END);
  datalength = ftell(fp);
  fseek(fp,0,SEEK_SET);
  *data = new char [datalength+1];
  *((char *)*data + datalength) = '\0';
  fread(*data,datalength,1,fp);
  fclose(fp);
  unlink(EDITFILE);
}

#undef EDITFILE


void free_links_list(link_list_node *lp)
/*
  Free memory occupied by the links list
*/
{
  link_list_node *lp1,*lp2;

  for(lp1 = lp->next;lp1 != 0;lp1 = lp2 )
  {
        lp2 = lp1->next;
        delete lp1;
  }
  lp->next = 0;
}
```

Αρχείο "text.h"

```
#ifndef __TEXTFILE__
#define __TEXTFILE__
#include <string.h>
#include "item.h"


class text : public item
/*
  The base class is declared 'public' so that all methods
  defined in it are accessible
*/
{
  char *data;

  public:
        text() {data = 0;}
        ~text() {if (data != 0) delete data;}
        void set_data(char *);
        char *get_data(void) {return data;}
        int Delete(void)
        {
          delete data;
          data = new char[1];
          *data = '\0';
          item::Delete();
        }
        /*
          The following methods redefine the inherited from
          class 'item' methods with the same name...
        */
        int place_to_buffer(void *);
        int retrieve_from_buffer(void *);
        int size();
};


inline int text::size()
{
  return (data == 0) ?
        (item::size()) : (item::size()+strlen(data)+1);
}


void text::set_data(char *dta)
{
  if (data != 0)
        delete data;
  data = dta;
}


int text::retrieve_from_buffer(void *buffer)
{
```

```
  int total_size;
  char *bp;

  if (data != 0)
        delete data;
  total_size = item::retrieve_from_buffer(buffer);
  bp = (char *)buffer + total_size;
  total_size += strlen(bp) + 1;
  data = new char [strlen(bp)+1];
  strcpy(data,bp);
  return total_size;
}


int text::place_to_buffer(void *buffer)
{
  char *bp;
  int total_size;

  total_size = item::place_to_buffer(buffer);
  bp = (char *)buffer + total_size;
  strcpy(bp,data);
  return total_size + strlen(data);
}


#endif
```

Αρχείο "item.h"

```
#ifndef __ITEMFILE__
#define __ITEMFILE__
#include <string.h>
#include "authored.h"
#include "itemcons.h"
#include "link.h"
#include "filecons.h"
#include "system.h"
#include <std.h>
#include "vercons.h"
#include "date.h"
#include <values.h>

#define ITEMNOTFOUND (-1L)
#define NOSUCHVERSION (-2L)

typedef struct property_list
{
  char *attribute;
  char *value;
```

```
  struct property_list *next;
} property_node;


typedef struct anchored_link_list
{
  LinkId_t id;
  int anchor_start, anchor_length;
  struct anchored_link_list *next;
} anchored_link_node;


class item : public authored_object
{
  int id;
  long prev_version_ptr;
  long prev_version_size;
  long next_version_ptr;       /* defined but not supported */
  long next_version_size;    /* defined but not supported */
  char *description;
  int content_type;
  property_node *properties;
  anchored_link_node *anchored_links;

  public:
        item();
        ~item();
        void create(char *,char *,int);
        int retrieve(int);
        int retrieve_of_author(int,const char *);
        int retrieve_with_linkid(int, LinkId_t);
        int get_previous_version(void);
        int get_previous_version_of_author(const char *);
        void save(void);
        void Delete(void);
        int get_id(void) {return id;}
        char *get_description(void) {return description;}
        property_node *get_properties(void)
          {return properties->next;}
        anchored_link_node *get_anchored_links(void)
          {return anchored_links->next;}
        int has_link_id(LinkId_t);
        int get_link_anchor(LinkId_t, int *, int *);
        void set_description(char *);
        int add_property(char *,char *);
        int drop_property(char *, char *);
        int add_anchored_link(LinkId_t,int,int);
        int drop_anchored_link(LinkId_t);
        int get_content_type(void) {return content_type;}
        /*
          The following methods are defined to be 'virtual' so
          that the binding will be late, i.e. it will take
          place during run time, not during compile time. This
          causes the calling of the correct member function,
```

```
        depending on which class the item receiving the
        message belongs to.
       */
       virtual int place_to_buffer(void *);
       virtual int retrieve_from_buffer(void *);
       virtual int size(void);
};


item::item()
/* Initialize pointer variables */
{
 id = 0;
 description = 0;
 properties = new property_node;
 properties->attribute = properties->value = 0;
 properties->next = 0;
 anchored_links = new anchored_link_node;
 anchored_links->next = 0;
}


item::~item()
/* Free memory held by item */
{
 property_node *p1,*p2;
 anchored_link_node *l1,*l2;

 if (description != 0)
      delete description;
 p1 = properties;
 while (p1 != 0)
 {
      p2 = p1->next;
      if (p1->attribute != 0)
        delete p1->attribute;
      if (p1->value != 0)
        delete p1->value;
      delete p1;
      p1 = p2;
 }
 l1 = anchored_links;
 while (l1 != 0)
 {
      l2 = l1->next;
      delete l1;
      l1 = l2;
 }
}


void item::create(char *auth_nam,char *desc,int cont_type)
/*
 The item::create method frees memory currently owned by
```

```
  the item, and sets the data fields to the given values
*/
{
  anchored_link_node *l1,*l2;
  property_node *p1,*p2;

  id = new_key();
  prev_version_ptr = prev_version_size = 0L;
  next_version_ptr = next_version_size = 0L;
  set_author_id(auth_nam);
  if (description != 0)
        delete description;
  description = new char[strlen(desc)+1];
  strcpy(description,desc);
  content_type = cont_type;
  p1 = properties->next;
  while (p1 != 0)
  {
        p2 = p1->next;
        if (p1->attribute != 0)
          delete p1->attribute;
        if (p1->value != 0)
          delete p1->value;
        delete p1;
        p1 = p2;
  }
  properties->next = 0;
  l1 = anchored_links->next;
  while (l1 != 0)
  {
        l2 = l1->next;
        delete l1;
        l1 = l2;
  }
  anchored_links->next = 0;
}


int item::size(void)
/* Return length of data fields of the item */
{
  int length;
  anchored_link_node *ap;
  property_node *pp;

  length = 4 * sizeof(long); /* version pointers */
  length += strlen(description) + 1;
  for (pp = properties->next; pp != 0; pp = pp->next)
        length += strlen(pp->attribute) + strlen(pp->value) +2;
  for (ap = anchored_links->next; ap != 0; ap = ap->next)
        length += sizeof(LinkId_t) + 2 * sizeof(int);
  length += sizeof(int);       /* length for content_type */
  /*
        add sizeof(LinkId_t)+1 for markers used in the buffers
```

```
 */
 return length+authored_object::size()+sizeof(LinkId_t)+1;
}


void item::set_description(char *new_description)
/*
 Free memory currently owned by "description" and set it
 to the new value
*/
{
 if (description != 0)
      delete description;
 description = new char[strlen(new_description)+1];
 strcpy(description,new_description);
}


int item::add_property(char *new_attribute,char *new_value)
/*
 Test if property already exists. If so, return 1, else
 add it and return 0
*/
{
 property_node *p1,*p2,*new_node;

 p1 = properties->next;
 p2 = properties;
 while (p1 != 0)
 {
      if (! strcmp(p1->attribute,new_attribute) &&
            ! strcmp(p1->value,new_value)                )
              return 1;
      p2 = p1;
      p1 = p1->next;
 }
 new_node = new property_node;
 new_node->attribute = new char[strlen(new_attribute) +1];
 strcpy(new_node->attribute,new_attribute);
 new_node->value = new char[strlen(new_value) + 1];
 strcpy(new_node->value,new_value);
 new_node->next = 0;
 p2->next = new_node;
 return 0;
}


int item::drop_property(char *attr,char *val)
/*
 If the property does not exist, return 1, else delete
 it and return 0.
*/
{
 property_node *p1,*old_property;
```

```
 p1 = properties;
 old_property = properties->next;
 while (old_property != 0)
 {
      if (!strcmp(old_property->attribute,attr) &&
            !strcmp(old_property->value,val) )
      {
            delete old_property->attribute;
            delete old_property->value;
            p1->next = old_property->next;
            delete old_property;
            return 0;
      }
      p1 = old_property;
      old_property = old_property->next;
 }
 return 1;
}


int item::add_anchored_link(LinkId_t lid, int st, int le)
/*
 If the link already exists return 1, else add it to the
 list and return 0.
*/
{
 anchored_link_node *l1,*l2,*new_node;

 l1 = anchored_links->next;
 l2 = anchored_links;
 while (l1 != 0)
 {
      if (l1->id == lid) /* Just test the id */
        return 1;
      l2 = l1;
      l1 = l1->next;
 }
 new_node  = new anchored_link_node;
 new_node->id = lid;
 new_node->anchor_start = st;
 new_node->anchor_length = le;
 new_node->next = 0;
 l2->next = new_node;
 return 0;
}


int item::drop_anchored_link(LinkId_t lid)
/*
 If the link does not exist return 1,
 else remove it from the list and return 0
*/
{
```

```
 anchored_link_node *l1,*l2;

 l1 = anchored_links;
 l2 = anchored_links->next;
 while (l2 != 0)
 {
       if (l2->id == lid)
       {
             l1->next = l2->next;
             delete l2;
             return 0;
       }
       l1 = l2;
       l2 = l2->next;
 }
 return 1;
}


int item::has_link_id(LinkId_t lid)
{
 anchored_link_node *ap;

 for (ap = anchored_links->next; ap != 0; ap = ap->next)
       if (ap->id == lid)
         return 0;
 return 1;
}


int item::get_link_anchor(LinkId_t lid, int *st, int *le)
/*
 If the link exists, return the anchror in the two int *
 fields. If it doesn't, return 1.
*/
{
 anchored_link_node *ap;

 for (ap = anchored_links->next; ap != 0; ap = ap->next)
       if (ap->id == lid)
       {
         *st = ap->anchor_start;
         *le = ap->anchor_length;
         return 0;
       }
 return 1;
}


int item::retrieve(int item_id)
/*
 Retrieve all data for item "id", that is the description,
 links and properties.
 If an error occurs, i.e. the item does not exist, -1L is
```

```
 returned, else the returned value is the length of the
 actual data.
*/
{
 void *buffer;
 int data_size;

 data_size = retrieve_item_data(item_id,&buffer);
 if (data_size > 0)
 {
      id = item_id;
      (*this).retrieve_from_buffer(buffer);
      delete buffer;
 }
 return (data_size > 0) ? data_size : ITEMNOTFOUND;
}


int item::retrieve_of_author(int item_id,const char *auid)
/*
 Return the last version of a specified item which has
 been created by a specified author.
*/
{
 if ((*this).retrieve(item_id) == ITEMNOTFOUND)
      return ITEMNOTFOUND;
 do
 {
      if (!strcmp(auid,get_author_id()))
        return (*this).size();
      if ((*this).get_previous_version() == NOSUCHVERSION)
        return NOSUCHVERSION;
 }
 while (1);
}


int item::retrieve_with_linkid(int item_id,LinkId_t lid)
/*
 Retrieve the first version of an item with a specific
 link id. Very useful for traversing links.
*/
{
 if ((*this).retrieve(item_id) == ITEMNOTFOUND)
      return ITEMNOTFOUND;
 do
 {
      if ((*this).has_link_id(lid) == 0)
        return (*this).size();
      if ((*this).get_previous_version() == NOSUCHVERSION)
        return NOSUCHVERSION;
 }
 while (1);
}
```

```
int item::get_previous_version(void)
{
 void *buffer;

 if ( (prev_version_ptr == 0)||(prev_version_size == 0))
      return NOSUCHVERSION;
 buffer = new char [prev_version_size];
 read_data(buffer,prev_version_ptr,prev_version_size);
 (*this).retrieve_from_buffer(buffer);
 delete buffer;
 return (*this).size();
}


int item::get_previous_version_of_author(const char *auid)
{
 do
 {
      if ((*this).get_previous_version() == NOSUCHVERSION)
        return NOSUCHVERSION;
      if (!strcmp(auid,get_author_id()))
        return (*this).size();
 }
 while (1);
}


int item::retrieve_from_buffer(void *buffer)
{
 property_node *p1,*p2;
 anchored_link_node *l1,*l2;
 char *bp;
 int total_size;

/*
 Free memory currently occupied by the item.
 Note that the "sentinel" data of the item, i.e. one
 dummy record at the start of the property list, and
 one dummy record at the start of the link list, remain
 as they are, or they should be reallocated a few lines
 bellow.
*/

 if (description != 0)
      delete description;
 p1 = properties->next;
 while (p1 != 0)
 {
      p2 = p1->next;
      delete p1->attribute;
      delete p1->value;
      delete p1;
```

```
        p1 = p2;
    }
    properties->next = 0;
    l1 = anchored_links->next;
    while (l1 != 0)
    {
        l2 = l1->next;
        delete l1;
        l1 = l2;
    }
    anchored_links->next = 0;
    total_size =
        authored_object::retrieve_from_buffer(buffer);
    bp = (char *)buffer + total_size;
    prev_version_ptr = *(long *)bp;
    bp += sizeof(long);
    prev_version_size = *(long *)bp;
    bp += sizeof(long);
    next_version_ptr = *(long *)bp;
    bp += sizeof(long);
    next_version_size = *(long *)bp;
    bp += sizeof(long);
    total_size += 4 * sizeof(long);
    p1 = properties;
    while (strlen(bp) != 0)
    {
        p2 = new property_node;
        p2->next = 0;
        p2->attribute = new char [strlen(bp)+1];
        strcpy(p2->attribute,bp);
        total_size += strlen(bp) + 1;
        bp += strlen(bp) + 1;
        p2->value = new char [strlen(bp) + 1];
        strcpy(p2->value,bp);
        total_size += strlen(bp) + 1;
        bp += strlen(bp) + 1;
        p1 ->next = p2;
        p1 = p2;
    }
    bp++;
    total_size++;
    content_type = *(int *)bp;
    bp += sizeof(int);
    total_size += sizeof(int);
    description = new char [strlen(bp)+1];
    strcpy(description,bp);
    total_size += strlen(bp) + 1;
    bp += strlen(bp) + 1;
    l1 = anchored_links;
    while (*(LinkId_t *)bp != LINKEND)
    {
        l2 = new anchored_link_node;
        l2->id = *(LinkId_t *) bp;
        l2->anchor_start = *(int *)(bp + sizeof(LinkId_t));
```

```
        l2->anchor_length =
          *(int *)(bp + sizeof(LinkId_t) + sizeof(int));
        l2->next = 0;
        l1->next = l2;
        l1 = l2;
        bp += sizeof(LinkId_t) + 2 * sizeof(int);
        total_size += sizeof(LinkId_t) + 2 * sizeof(int);
  }
  bp += sizeof(LinkId_t);
  total_size += sizeof(LinkId_t);
  return total_size;
}


void item::save(void)
{
  void *data_buffer;
  int data_size;
  struct disk_item di;

  data_size = (*this).size();
  data_buffer =
        new char [u_quotient(data_size,DISKBLOCK)*DISKBLOCK];
  get_info(id,&di);
  if (di.id == id)
  {
        prev_version_ptr = di.dataptr;
        prev_version_size = di.datasize;
  }
  (*this).place_to_buffer(data_buffer);
  save_item(id,data_buffer,data_size);
}


int item::place_to_buffer(void *buffer)
/*
  Place all the data in a buffer. The buffer holds
  the data in the following format :
              <supreclass data>  prev_version_ptr
              prev_version_size next_version_ptr
              next_version_size <property_list> '\0'
              content_type  description '\0'
              <anchored_link_list> LINKEND
        where
              <property_list>::=<property> '\0' |
                                    <property> '\0' <property_list>

              <property> ::= attribute '\0' value '\0'

              <anchored_link_list>::=<anchored_link> |
                                          <anchored_link><linkid_list>
         and
              <anchored_link> ::=
                link_id anchor_start anchor_length
```

```
  'buffer' is assumed to be long enough to hold all fields.
*/
{
  char *bp;
  int total_size;
  property_node *pp;
  anchored_link_node *ap;

  total_size = authored_object::place_to_buffer(buffer);
  bp = (char *)buffer + total_size;
  *(long *)bp = prev_version_ptr;
  *(long *)(bp + sizeof(long)) = prev_version_size;
  *(long *)(bp + 2 * sizeof(long)) = next_version_ptr;
  *(long *)(bp + 3 * sizeof(long)) = next_version_size;
  bp += 4 * sizeof(long);
  total_size += 4 * sizeof(long);
  for (pp = properties->next; pp != 0; pp = pp->next)
  {
       strcpy(bp,pp->attribute);
       total_size += strlen(pp->attribute) + 1;
       bp += strlen(pp->attribute) + 1;
       strcpy(bp,pp->value);
       total_size += strlen(pp->value) + 1;
       bp += strlen(pp->value) + 1;
  }
  *bp++ = '\0';
  total_size++;
  *(int *)bp = content_type;
  bp += sizeof(int);
  total_size += sizeof(int);
  strcpy(bp,description);
  bp += strlen(description) + 1;
  total_size += strlen(description) + 1;
  for (ap = anchored_links->next;ap != 0; ap = ap->next)
  {
       *(LinkId_t *)bp = ap->id;
       *(int *)(bp + sizeof(LinkId_t)) = ap->anchor_start;
       *(int *)(bp+sizeof(LinkId_t)+sizeof(int)) =
         ap->anchor_length;
       bp += sizeof(LinkId_t) + 2 * sizeof(int);
       total_size += sizeof(LinkId_t) + 2 * sizeof(int);
  }
  *(LinkId_t *)bp = LINKEND;
  total_size += sizeof(LinkId_t);
  return total_size;
}


void item::Delete(void)
/*
  Delete an item from the database. This means, drop all
  link identifiers, and keywords, and set the description
  to a null string.
```

```
*/
{
 anchored_link_node *ap1,*ap2;
 property_node *pp1,*pp2;

 if (description != 0)
      delete description;
 description = new char[1];
 *description = '\0';
 for (ap1 = anchored_links->next; ap1 != 0; )
 {
      ap2 = ap1->next;
      delete ap1;
      ap1 = ap2;
 }
 anchored_links->next = 0;
 for (pp1 = properties->next; pp1 != 0; )
 {
      pp2 = pp1->next;
      if (pp1->attribute != 0)
        delete pp1->attribute;
      if (pp1->value != 0)
        delete pp1->value;
      delete pp1;
      pp1 = pp2;
 }
 properties->next = 0;
}

#endif
```

Αρχείο "link.h"

```
#ifndef __LINK__

#define __LINK__
#include "authored.h"
#include <stream.h>
#include <stdio.h>
#include "linkcons.h"
#include <string.h>
#include <std.h>

#ifndef SEEK_SET
#define SEEK_SET (0)
#define SEEK_END (2)
#endif

#define NULLTYPE (-1)
```

```
class link1 : public authored_object
{
 LinkId_t id;
 int source;
 int target;
 int Type;
 int mirror_Type;
 float weight;

 public:
      void Create(char *,int,int,int,int,float);
      int Delete();
      int size()
        {return authored_object::size() + LinkRecSize;}
      int place_to_buffer(void *);
      int retrieve_from_buffer(void *);
      void Save();
      int  Retrieve(LinkId_t);
      void display();
      LinkId_t ReturnLinkid() {return id;}
      int ReturnSource() {return source;}
      int ReturnTarget() {return target;}
      int ReturnType() {return Type;}
      int Return_mirror_Type() {return mirror_Type;}
      float Returnweight() {return weight;}
};


void link1::display()
{
 date aDate;

 printf("LinkId : %d, Source : %d, Target : %d,"
                      " Type : %d, Weight: %f",
                      id,source,target,Type,weight);
 if (mirror_Type == NULLTYPE)
      printf(". Mirror type is NULL\n");
 else
      printf(", Mirror type: %d\n",mirror_Type);
 printf("Author is : %s, at : ",get_author_id());
 get_time_stamp(&aDate);
 print_date(aDate);
 printf("\n");
}


void link1::Create(char *auid,int src, int trg,int type,
                           int mirror_t,float weigh)
/*
 Create the link. This is done by assigning the data to
 the corresponding fields, and assigning a unique key.
*/
{
```

```
FILE *fp;
LinkId_t new_key,limits[2];
char _dummy[size()];

set_author_id(auid);
source = src;
target = trg;
Type= type;
mirror_Type = mirror_t;
weight=weigh;
if ((fp = fopen(LINKFILE,"r+b")) == 0)
{
     fprintf(stderr,"Couldn't open linkfile\n");
     exit (2);
}
/*
     Get the next available key and the first key in
     the deleted key list
*/
if (fread(limits,sizeof(LinkId_t),2,fp) != 2)
{
     fprintf(stderr,"file %s is malformed.\n",LINKFILE);
     exit(2);
}
if (limits[1] == 0) /* No keys in the deleted list */
{
     new_key = limits[0]++;
     /* Write the new values */
     fseek(fp,0,SEEK_SET);
     fwrite(limits,sizeof(LinkId_t),2,fp);
     fclose(fp);
     /*
       And append a record to the link file for the new link
     */
     fp = fopen(LINKFILE,"ab");
     *(LinkId_t *)(_dummy + authored_object::size()) = 0;
     fwrite(_dummy,size(),1,fp);
     fclose(fp);
}
else /* There exists a key in the deleted key list */
{
     /* So use it */
     new_key = -limits[1];
     /* Get the next key in the list and mark it as first */
     fseek(fp,2*sizeof(LinkId_t)+size()*(new_key-1)+
             authored_object::size(),SEEK_SET);
     fread(&limits[1],sizeof(LinkId_t),1,fp);
     fseek(fp,0,SEEK_SET);
     fwrite(limits,sizeof(LinkId_t),2,fp);
     fseek(fp,2*sizeof(LinkId_t)+size()*(new_key-1),
             SEEK_SET);
     *(LinkId_t *)(_dummy + authored_object::size()) = 0;
     fwrite(_dummy,size(),1,fp);
     fclose(fp);
```

```
  }
  id = new_key;
}


int link1::Retrieve(LinkId_t Lid)
{
  FILE *fp;
  char link_buff[size()];

  fp=fopen(LINKFILE,"rb");
  if (fseek(fp,(Lid-1)*size()+2*sizeof(LinkId_t),SEEK_SET)
        ==-1|| feof(fp))
  /* The key is too large */
  {
        fclose(fp);
        return 1;
  }
  if (fread(link_buff,size(),1,fp) != 1)
  {
        fclose(fp);
        return 1;
  }
  fclose(fp);
  retrieve_from_buffer(link_buff);
  if (id != Lid)
        return 1;
  return 0;
}


int link1::retrieve_from_buffer(void *link_buff)
{
  char *link_bp;
  int superclass_size;

  superclass_size =
        authored_object::retrieve_from_buffer(link_buff);
  link_bp = (char *)link_buff + superclass_size;
  id = *(LinkId_t *)link_bp;
  link_bp += sizeof(LinkId_t);
  source = *(int *)link_bp;
  link_bp += sizeof(int);
  target = *(int *)link_bp;
  link_bp += sizeof(int);
  Type = *(int *)link_bp;
  link_bp += sizeof(int);
  mirror_Type = *(int *)link_bp;
  link_bp += sizeof(int);
  weight = *(float *)link_bp;
  return 0;
}
```

```
void link1::Save()
/* Save the link */
{

 FILE *fp;
 char link_buff[size()];

 fp=fopen(LINKFILE,"r+b");
 /* Set the file pointer to the right place */
 fseek(fp,2*sizeof(LinkId_t) +(id - 1)*size(),SEEK_SET);
 /* Place all the data in a buffer */
 place_to_buffer(link_buff);
 /* Write the buffer to the file */
 fwrite(link_buff,size(),1,fp);
 fclose(fp);
}


int link1::place_to_buffer(void *link_buff)
{
 char *link_bp;
 int superclass_size;

 superclass_size =
      authored_object::place_to_buffer(link_buff);
 link_bp = (char *)link_buff + superclass_size;
 *(LinkId_t *)link_bp = id;
 link_bp += sizeof(LinkId_t);
 *(int *)link_bp = source;
 link_bp += sizeof(int);
 *(int *)link_bp = target;
 link_bp += sizeof(int);
 *(int *)link_bp = Type;
 link_bp += sizeof(int);
 *(int *)link_bp = mirror_Type;
 link_bp += sizeof(int);
 *(float *)link_bp = weight;
}


int link1::Delete()
/* Delete a link */
{
 FILE *fp;
 LinkId_t limits[2],last_del;

 fp = fopen(LINKFILE,"r+b");
 fread(limits,sizeof(LinkId_t),2,fp);
 if (limits[0] <= id || id <= 0)
 /* Invalid key ! */
 {
      fclose(fp);
      return 1;
 }
```

```
  /* Add the key to the list of deleted keys as first */
  last_del = limits[1];
  limits[1] = -id;
  fseek(fp,0,SEEK_SET);
  fwrite(limits,sizeof(LinkId_t),2,fp);
  fseek(fp,2*sizeof(LinkId_t) + (id-1)*size() +
              authored_object::size(), SEEK_SET);
  fwrite(&last_del,sizeof(LinkId_t),1,fp);
  fclose(fp);
  return 0;
}

#endif
```

Αρχείο "authored.h"

```
#ifndef __AUTHOREDOBJECTS__

#define __AUTHOREDOBJECTS__

#include "date.h"
#include "authcons.h"
#include <string.h>
#include <stdio.h>

#define TIMENOTINITIALIZED (0)

class authored_object
{
  char author_id[MAXAUTHORNAME];
  packedDate_t timestamp;

  public :
       authored_object(void)
       {
         *author_id = '\0';
         timestamp = TIMENOTINITIALIZED;
       }

       char *get_author_id(void)
       {
         return author_id;
       }

       void set_author_id(char *new_id)
       {
         strncpy(author_id,new_id,MAXAUTHORNAME);
       }

       void get_time_stamp (date *aDate)
```

```
        {
         if (timestamp==TIMENOTINITIALIZED)
              get_system_date(aDate);
         else
              unpack_date(aDate,timestamp);
        }

        void set_time_stamp(date aDate)
        {
         timestamp = pack_date(aDate);
        }

        int size(void)
        {
         return MAXAUTHORNAME + sizeof(packedDate_t);
        }

        int place_to_buffer(void *);
        int retrieve_from_buffer(void *);
};


int authored_object::place_to_buffer(void *buffer)
{
 int length;

 length = MAXAUTHORNAME + sizeof(packedDate_t);
 strcpy((char *)buffer,author_id);
 *(packedDate_t *)((char *)buffer+strlen(author_id)+1) =
      timestamp;
 return length;
}


int authored_object::retrieve_from_buffer(void *buffer)
{
 int length;

 length = MAXAUTHORNAME + sizeof(packedDate_t);
 strcpy(author_id,(char *)buffer);
 timestamp = *(packedDate_t *)((char *)buffer +
                        strlen(author_id) + 1);
 return length;
}

#endif
```

Αρχείο "itemcons.h"

```
#ifndef __ITEMCONSTS__
```

```
#define __ITEMCONSTS__
#define DATAFILE "datafile"
struct disk_item
{
  int id;
  long dataptr,datasize;
};
const int ItemRecSize = sizeof(int) + 2*sizeof(long);
#endif __ITEMCONSTS__
```

Αρχείο "filecons.h"

```
#ifndef __FILINGCONSTS__
#define __FILINGCONSTS__
#include "itemcons.h"
#define DISKBLOCK (512)
#define ORGANIZEFILE "itemfile"
#endif
```

Αρχείο "linkcons.h"

```
#ifndef __LINKCONSTS__
#define __LINKCONSTS__
#include "itemcons.h"
typedef int LinkId_t;
const int LinkRecSize =
  sizeof(LinkId_t) + 4*sizeof(int) + sizeof(float);
const LinkId_t LINKEND = (LinkId_t) -1;
#define LINKFILE "linkfile"
#endif
```

Αρχείο "authcons.h"

```
#ifndef __AUTHORCONST__
#define __AUTHORCONST__
#define MAXAUTHORNAME (20)
#endif
```

Αρχείο "system.h"

```
#ifndef __SYSTEMCORE__
```

```
#define __SYSTEMCORE__
#include "filecons.h"
#include <stdio.h>

#ifndef SEEK_SET
#define SEEK_SET (0)
#define SEEK_END (2)
#endif


void read_data(void *buffer, long dataptr, long datasize);


inline long u_quotient(long dividend, long divisor)
/*
  This function returns ceil(dividend,divisor)
*/
{
  long quotient,remainder;

  remainder = dividend % divisor;
  quotient = dividend / divisor;
  return (remainder == 0) ? quotient : quotient + 1;
}


void get_info(int item_id,struct disk_item *di)
/*
  Return information about the location and the size of
  the last version of item 'item_id. If the key is somehow
  invalid, return 0 in all the fields of *di
*/
{
  FILE *fp;
  char buff[ItemRecSize];

  if ((fp = fopen(ORGANIZEFILE,"rb")) == 0)
  {
       fprintf(stderr,"Couldn't open %s\n",ORGANIZEFILE);
       exit(2);
  }
  if (fseek(fp,(item_id-1)*ItemRecSize+2 * sizeof(int),
         SEEK_SET) == -1 || feof(fp))
       /* Bad key ! */
  {
       fclose(fp);
       di->id = (int)(di->dataptr = di->datasize = 0L);
       return;
  }
  if (fread(buff,ItemRecSize,1,fp) != 1)
  {
       fclose(fp);
       di->id = (int)(di->dataptr = di->datasize = 0L);
       return;
```

```
 }
 fclose(fp);
 di->id = *(int *)buff;
 di->dataptr = *(long *)(buff + sizeof(int));
 di->datasize = *(long *)(buff + sizeof(int) +
                          sizeof(long));
}


int new_key(void)
/*
 Return a new key for a new item.
*/
{
 FILE *fp;
 int limits[2],new_key;
 char tmp_item[ItemRecSize];

 if ((fp = fopen(ORGANIZEFILE,"r+b")) == 0)
 {
      fprintf(stderr,"Couldn't open %s\n",ORGANIZEFILE);
      exit(2);
 }
 if (fread(limits,sizeof(int),2,fp) != 2)
 {
      fprintf(stderr,"File %s is malformed\n",ORGANIZEFILE);
      fclose(fp);
      exit(2);
 }
 if (limits[1] == 0)
 {
      new_key = limits[0]++;
      fseek(fp,0L,SEEK_SET);
      fwrite(limits,sizeof(int),2,fp);
      fclose(fp);
      fp = fopen(ORGANIZEFILE,"ab");
      *(int *)tmp_item = 0;
      fwrite(tmp_item,ItemRecSize,1,fp);
      fclose(fp);
 }
 else
 {
      new_key = -limits[1];
      fseek(fp,2 * sizeof(int) + ItemRecSize * (new_key - 1),
            SEEK_SET);
      fread(&limits[1],sizeof(int),1,fp);
      fseek(fp,0L,SEEK_SET);
      fwrite(fp,sizeof(int),2,fp);
      *(int *)tmp_item = 0;
      fseek(fp,2 * sizeof(int) + ItemRecSize * (new_key - 1),
            SEEK_SET);
      fwrite(tmp_item,ItemRecSize,1,fp);
      fclose(fp);
 }
```

```
 return new_key;
}


void save_item(int id, void *data_buffer,long size)
{
 FILE *fp1, *fp2;
 long dataptr;
 char item_buff[ItemRecSize];

 if ((fp1 = fopen(ORGANIZEFILE,"r+b")) == 0)
 {
      fprintf(stderr,"Couldn't open file %s\n",ORGANIZEFILE);
      exit(2);
 }
 if ((fp2 = fopen(DATAFILE,"ab")) == 0)
 {
      fprintf(stderr,"Couldn't open file %s\n",DATAFILE);
      exit(2);
 }
 fseek(fp1,2 * sizeof(int) + ItemRecSize * (id - 1),
            SEEK_SET);
 fseek(fp2,0,SEEK_END);
 dataptr = ftell(fp2);
 *(int *)item_buff = id;
 *(long *)(item_buff + sizeof(int)) = dataptr;
 *(long *)(item_buff + sizeof(int) + sizeof(long)) =size;
 fwrite(item_buff,ItemRecSize,1,fp1);
 fclose(fp1);
 fwrite(data_buffer,DISKBLOCK,u_quotient(size,DISKBLOCK),
            fp2);
 fclose(fp2);
}


int retrieve_item_data(int id, void **buffer)
{
 FILE *fp;
 char item_buff[ItemRecSize];
 long size;

 if ((fp = fopen(ORGANIZEFILE,"rb")) == 0)
 {
      fprintf(stderr,"Couldn't open file %s\n",ORGANIZEFILE);
      exit(2);
 }
 if (fseek(fp,2*sizeof(int) + (id - 1) * ItemRecSize,
       SEEK_SET) == -1 || feof(fp))
 {
  fclose(fp);
  return -1;
 }
 if (fread(item_buff,ItemRecSize,1,fp) != 1)
 {
```

```
        fclose(fp);
        return -1;
  }
  fclose(fp);
  if (*(int *)item_buff != id)
        return -1;
  *buffer= new char
        [(size=*(long *)(item_buff+sizeof(int)+sizeof(long)))];
  read_data(*buffer,
                    *(long *)(item_buff + sizeof(int)),size);
  return size;
}


void read_data(void *buffer, long dataptr, long datasize)
{
  FILE *fp;

  if ((fp = fopen(DATAFILE,"rb")) == 0)
  {
        fprintf(stderr,"Couldn't open file %s\n",DATAFILE);
        exit(2);
  }
  fseek(fp,dataptr,SEEK_SET);
  fread(buffer,datasize,1,fp);
  fclose(fp);
}


inline int item_exists(int key)
{
  struct disk_item di;

  get_info(key,&di);
  return ( (di.id == 0) || (di.id != key)) ? 1 : 0;
}

#endif
```

Αρχείο "sequential.h"

```
#ifndef __SEQUENTIAL__
#define __SEQUENTIAL__
#include "filecons.h"
#include <stdio.h>
#include <std.h>
#include <string.h>

#define NOT_INITIALIZED (1)
#define SUCCESS (0)
```

```
#ifndef SEEK_SET
#define SEEK_SET (0)
#define SEEK_END (2)
#endif

/*
 The following variables are declared outside of any
 function in this file, so that they are common to all
 functions in this file.
*/

static int current_id;
static int initialized = 0;
static int end_of_file = 0;


void initialize_search(void)
{
 initialized = 1;
 end_of_file = 0;
 current_id = 0;
}


int get_next_id(int *next_id)
{
 char seq_buffer[ItemRecSize];
 FILE *fp;

 if (! initialized)
      return NOT_INITIALIZED;
 if (end_of_file)
      return EOF;
 if ((fp = fopen(ORGANIZEFILE,"rb")) == 0)
 {
      fprintf(stderr,"Couldn't open file %s\n",ORGANIZEFILE);
      exit(2);
 }
 for( ; 1 ;)
 {
      current_id ++;
      fseek(fp,2*sizeof(int) + (current_id-1)* ItemRecSize,
              SEEK_SET);
      if (fread(seq_buffer,ItemRecSize,1,fp) != 1)
      {
       end_of_file = 1;
       fclose(fp);
       return EOF;
      }
      if (*(int *)seq_buffer == current_id)
      {
       fclose(fp);
       *next_id = current_id;
```

```
        return SUCCESS;
      }
  }
}

#endif
```

Αρχείο "date.h"

```
#ifndef __SYSTEMDATE__
#define __SYSTEMDATE__

#include <std.h>
#include <stdio.h>
#define DATEFILE "TMPDATE"

typedef struct
{
 unsigned short day,month, year,hour,minute;
} date;

typedef long packedDate_t;


void get_system_date(date *cur_date)
{
 FILE *fp;
 char command[255];

 sprintf(command,
              "date '+%%d%%t%%m%%t%%y%%t%%H%%t%%M%%n' >%s\n",
              DATEFILE);
 system(command);
 fp = fopen(DATEFILE,"r");
 fscanf(fp,"%hu%hu%hu%hu%hu%hu",&(cur_date->day),
            &(cur_date->month),&(cur_date->year),
            &(cur_date->hour),&(cur_date->minute));
 fclose(fp);
 unlink(DATEFILE);
}


inline packedDate_t pack_date(date aDate)
{
 return (long)aDate.minute | ((long)aDate.hour << 6) |
            ((long)aDate.day << 11) |
            ((long)aDate.month << 16) |
            ((long)aDate.year << 20);
}
```

```
void unpack_date(date *aDate,packedDate_t packed_date)
{
 aDate->minute = (unsigned short)(packed_date&0x3f);
 aDate->hour = (unsigned short)((packed_date >> 6 )&0x1f);
 aDate->day = (unsigned short)((packed_date >> 11)&0x1f);
 aDate->month = (unsigned short)((packed_date>>16)&0x0f);
 aDate->year = (unsigned short)((packed_date>>20)&0xfff);
}


void print_date(date aDate)
{
 printf("(Date: %hu/%hu/%hu\tTime: %hu:%02hu)\n",
            aDate.day,aDate.month,aDate.year,aDate.hour,
            aDate.minute);
}

#endif
```

Αρχείο "mkitemfl.c"

```
#include <stdio.h>
#include "filecons.h"

main()
/*
 This program creates the item file. It initializes the
 first two (int) entries to contain 1 and 0 respectively.
 This means that the next available key is the key 1,
 and that there are no keys in the 'deleted key' list.
*/
{
 FILE *fp;
 int limits[2] ;

 limits[0] = 1;
 limits[1] = 0;
 fp = fopen(ORGANIZEFILE,"w");
 fwrite(limits,sizeof(int),2,fp);
 fclose(fp);
 return 0;
}
```

Αρχείο "mklinkfl.c"

```
#include <stdio.h>
```

```
#include "linkcons.h"

main()
/*
  This program creates the link file. It initializes the
  first two (LinkId_t) entries to contain 1 and 0,
  respectively. This means that the next available key is
  the key 1, and that there are no keys in the deleted
  key list.
*/
{
  FILE *fp;
  LinkId_t limits[2] ;

  limits[0] = 1;
  limits[1] = 0;
  fp = fopen(LINKFILE,"w");
  fwrite(limits,sizeof(LinkId_t),2,fp);
  fclose(fp);
  return 0;
}
```

Αρχείο "mkdatafl.c"

```
#include <stdio.h>
#include "filecons.h"

main()
/*
  This program initializes the data file. Its sole purpose
  is to create the file, so that it exists, and to write
  DISKBLOCK characters to it, so that no item data will be
  stored at address 0 in the file, because address 0 is
  interpreted as a NULL pointer.
*/
{
  char buff[DISKBLOCK];
  FILE *fp;

  fp = fopen(DATAFILE,"wb");
  fwrite(buff,DISKBLOCK,1,fp);
  fclose(fp);
  return 0;
}
```

Δ. ΑΓΓΛΟ-ΕΛΛΗΝΙΚΟ ΓΛΩΣΣΑΡΙ ΟΡΩΝ

Δ.1. Μεταφρασμένοι Όροι

abstract class	αφηρημένη κλάση
abstract data type	αφηρημένος τύπος δεδομένων
access method	μέθοδος προσπέλασης
accessor function	λειτουργία προσπέλασης
active object	ενεργητικό αντικείμενο
active view	ενεργή όψη
administration	διαχείριση
administrator	διαχειριστής
analog	αναλογικός
anchor	άγκυρα
AND-arc	τόξο-ΚΑΙ
AND-group	ομάδα-ΚΑΙ
annotation	σχολιασμός
arrival region	περιοχή άφιξης
associative searching	συσχετιστική αναζήτηση
atomic transaction	ατομική δοσοληψία
attribute	χαρακτηριστική
attribute/value pair	ζευγάρι χαρακτηριστικής/τιμής
author	συγγραφέας
authored object	αντικείμενο με συγγραφέα
authoring rights	συγγραφικά δικαιώματα
background	παρασκήνιο
backtrack command	εντολή οπισθοχώρησης
backup	τήρηση αντιγράφων
backward delta	δέλτα προς τα πίσω
backward traversal	διάσχιση προς τα πίσω
base table	βασικός πίνακας
bidirectional	αμφικατευθυνόμενος
bit	δυαδικό ψηφίο
bitmap array	πίνακας δυαδικών ψηφίων
buffer	ενδιάμεση μνήμη
bug	λάθος
button	κουμπί
call by reference	κλήση με αναφορά
candidate key	υποψήφιο κλειδί
CD	οπτικός δίσκος
CD-ROM	οπτικός δίσκος ανάγνωσης μόνο
CD-ROM Extended Architecture	Εκτεταμένη Αρχιτεκτονική CD-ROM
characteristic	χαρακτηριστικό
child class	κλάση-παιδί
class	κλάση
class hierarchy	ιεραρχία κλάσεων
class instance	στιγμιότυπο κλάσης

code modifiability	δυνατότητα τροποποίησης του κώδικα
code reusability	δυνατότητα επαναχρησιμοποίησης του κώδικα
collaboration support	υποστήριξη συνεργασίας
command language	γλώσσα εντολών
communication list	λίστα επικοινωνίας
compatible	συμβατός
competing process	ανταγωνιστική διεργασία
compiler	μεταγλωττιστής
compression	συμπίεση
conceptual document	ιδεατό έγγραφο
conceptual level	ιδεατό επίπεδο
concurrency control	έλεγχος σύγχρονης προσπέλασης
conjunction	σύζευξη
consistency	συνέπεια
consistency check	έλεγχος συνέπειας
consistent state	συνεπής κατάσταση
constructive modification	δημιουργική τροποποίηση
constructor	κατασκευαστής
content search	αναζήτηση ως προς το περιεχόμενο
corporate database	εταιρική βάση δεδομένων
corporate environment	περιβάλλον εταιρείας
correctness	ορθότητα
criterion	κριτήριο
current item	τρέχον item
daemon process	διεργασία-δαίμονας
data abstraction	αφαίρεση δεδομένων
data availability	διαθεσιμότητα δεδομένων
data base	βάση δεδομένων
data base administration	διαχείριση βάσης δεδομένων
data base administrator	διαχειριστής βάσης δεδομένων
data base schema	σχήμα βάσης δεδομένων
data encapsulation	περιχαράκωση δεδομένων
data flow diagram	διάγραμμα ροής δεδομένων
data manipulation language	γλώσσα χειρισμού δεδομένων
data sharing	διαμοιρασμός δεδομένων
data variable	μεταβλητή
deadlock	αδιέξοδο
decision support system	σύστημα υποστήριξης αποφάσεων
declarative	δηλωτικός
decompression	αποσυμπίεση
default value	προκαθορισμένη τιμή
delta	μεταβολή
delta technique	τεχνική δέλτα
departure region	περιοχή αναχώρησης
destructive modification	καταστροφική τροποποίηση
destructor	καταστροφέας
dictionary	λεξικό
digital	ψηφιακός
disjunction	διάζευξη
disorientation	αποπροσανατολισμός
early binding	πρόωρη δέσμευση

editing session	σύνοδος τροποποιήσεων
electronic document	ηλεκτρονικό έγγραφο
electronic mail	ηλεκτρονικό ταχυδρομείο
evaluation	αποτίμηση
exclusive lock	αποκλειστικό κλείδωμα
expert system	έμπειρο σύστημα
explicit	άμεσος
extensibility	δυνατότητα επεκτάσεων
extraction rule	κανόνας εξαγωγής
fisheye view	όψη προοπτικής
form	φόρμα
forward delta	δέλτα προς τα εμπρός
forward traversal	διάσχιση προς τα εμπρός
frame	πλαίσιο
friend function	φίλη συνάρτηση
full version	πλήρης εκδοχή
function prototype	πρωτότυπο συνάρτησης
garbage collection	συλλογή απορριμμάτων
garbage collector	συλλέκτης απορριμμάτων
global map	γενικός χάρτης
hard locking	σκληρό κλείδωμα
hardware	υλικό
hardware failure	βλάβη υλικού
high-level programming language	γλώσσα προγραμματισμού υψηλού επιπέδου
highlighting	έντονος φωτισμός
hypergraph	υπεργράφος
icon	εικονίδιο
identifier	προσδιοριστής
image processor	επεξεργαστής εικόνας
implicit	έμμεσος
inconsistent state	ασυνεπής κατάσταση
index	ευρετήριο
indivisible locking	αδιαίρετο κλείδωμα
information retrieval	ανάκτηση πληροφορίας
initialization	αρχικοποίηση
integrity	ακεραιότητα
integrity constraint	περιορισμός ακεραιότητας
integrity control	έλεγχος ακεραιότητας
intelligent data base	ευφυής βάση δεδομένων
interconnection administration	διαχείριση διασυνδέσεων
interconnection construction	κατασκευή διασυνδέσεων
interconnection updating	ενημέρωση διασυνδέσεων
interpreter	διερμηνευτής
journal	ημερολόγιο
journaling	τήρηση ημερολογίου
keyword	λέξη-κλειδί
landmark view	όψη προσανατολισμού
late binding	καθυστερημένη δέσμευση
light pen	οπτικό μολύβι
link	σύνδεσμος
local map	τοπικός χάρτης

mailing function	λειτουργία ταχυδρομείου
maintainability	δυνατότητα συντήρησης
mass storage	μαζική αποθήκευση
message	μήνυμα
message passing	πέρασμα μηνυμάτων
metasymbol	μετασύμβολο
method	μέθοδος
modelling construct	δομή μοντελοποίησης
modifiability	δυνατότητα αλλαγών
module	ενότητα
mouse	ποντίκι
multi-user environment	περιβάλλον πολλών χρηστών
navigation	περιπλάνηση
nested query	φωλιασμένη ερώτηση
node	κόμβος
node centrality	κεντρικότητα κόμβων
object	αντικείμενο
object-based	βασισμένος στα αντικείμενα
object-orientation	προσανατολισμός στα αντικείμενα
object-oriented	προσανατολισμένος στα αντικείμενα
office environment	περιβάλλον γραφείου
one phase locking	κλείδωμα σε μία φάση
on-line help	βοήθεια κατά τη χρήση
operator overloading	επιφόρτωση τελεστών
optimistic concurrency control	αισιόδοξος έλεγχος σύγχρονης προσπέλασης
OR-arc	τόξο-Η
organisation rule	κανόνας οργάνωσης
overview diagram	γενικό διάγραμμα
package	πακέτο
parent class	κλάση-γονέας
parser	συντακτικός αναλυτής
passive object	παθητικό αντικείμενο
password	σύνθημα
physical level	φυσικό επίπεδο
pointer	δείκτης
polymorphic redefinition	πολυμορφικός επανορισμός
polymorphism	πολυμορφισμός
presentation rule	κανόνας παρουσίασης
private method	ιδιωτική μέθοδος
private part	ιδιωτικό μέρος
private variable	ιδιωτική μεταβλητή
process	διεργασία
programming environment	περιβάλλον προγραμματισμού
property	ιδιότητα
property inheritance	κληρονομικότητα ιδιοτήτων
protected method	προστατευμένη μέθοδος
protected variable	προστατευμένη μεταβλητή
public method	δημόσια μέθοδος
public part	δημόσιο τμήμα
public variable	δημόσια μεταβλητή
query	ερώτηση

querying	υποβολή ερωτήσεων
raw data	ακατέργαστα δεδομένα
read key	κλειδί ανάγνωσης
read-only medium	μέσο ανάγνωσης μόνο
read/write medium	μέσο ανάγνωσης και εγγραφής
recompilation	επαναμεταγλώττιση
record	εγγραφή
recovery	επανόρθωση
referential integrity	ακεραιότητα αναφορών
relation	σχέση
relationship	συσχέτιση
report generator	γεννήτρια αναφορών
reserved word	δεσμευμένη λέξη
reverse video	αντίστροφος φωτισμός
rollback	επαναφορά
rule-based	βασισμένος σε κανόνες
sampler	δειγματολήπτης
scanner	σαρωτής εικόνας
schema	σχήμα
sector	τομέας
selection criterion	κριτήριο επιλογής
semantic	σημασιολογικός
session	σύνοδος
shadow paging	κρυφή σελιδοποίηση
shareable locking	διαμοιράσιμο κλείδωμα
shared lock	διαμοιραζόμενο κλείδωμα
sibling class	αδελφική κλάση
single-user environment	περιβάλλον ενός χρήστη
soft locking	μαλακό κλείδωμα
software	λογισμικό
software engineering	τεχνολογία λογισμικού
software failure	βλάβη λογισμικού
spreadsheet	λογιστικό φύλλο
standard	πρότυπο
static graphics	στατικά γραφικά
storage structure	δομή αποθήκευσης
string	συμβολοσειρά
source code	πηγαίος κώδικας
source item	αρχικό item
strongly-typed programming language	γλώσσα προγραμματισμού ισχυρών τύπων
structure	δομή
structured design	δομημένος σχεδιασμός
structured programming	δομημένος προγραμματισμός
structure search	αναζήτηση ως προς τη δομή
subclass	υποκλάση
superclass	υπερκλάση
syntax variable	συντακτική μεταβλητή
system-specified value	τιμή που καθορίζεται από το σύστημα
teleconferencing	τηλεδιάσκεψη
tailorability	δυνατότητα προσαρμογής
target item	τελικό item

text processor	επεξεργαστής κειμένου
text searching	αναζήτηση κειμένου
token	διακριτικό
transaction	δοσοληψία
transaction abort	εγκατάλειψη δοσοληψίας
transaction commitment	μονιμοποίηση δοσοληψίας
transaction processing	επεξεργασία δοσοληψίας
transfer rate	ρυθμός μεταβίβασης
transitive closure	μεταβατικό κλείσιμο
translator	μεταφραστής
traversal	διάσχιση
two phase locking	κλείδωμα σε δύο φάσεις
undo	ακυρώνω
unidirectional	μονοκατευθυνόμενος
uniformity	ομοιομορφία
update conflict	σύγκρουση ενημερώσεων
user group	ομάδα χρηστών
user-specified value	τιμή που καθορίζεται από το χρήστη
value	τιμή
version	εκδοχή
versioning	τήρηση εκδοχών
videodisk	βιντεοδίσκος
view	όψη
view consistency	συνέπεια μιας όψης
view maintainability	συντηρησιμότητα μιας όψης
view practicality	πρακτικότητα μιας όψης
view preciseness	ακρίβεια μιας όψης
virtual table	εικονικός πίνακας
weakly-typed programming language	γλώσσα προγραμματισμού ασθενών τύπων
workstation	σταθμός εργασίας
WORM disk	οπτικός δίσκος μίας εγγραφής-πολλών αναγνώσεων
write key	κλειδί εγγραφής
zoom-in	εστίαση από κοντά
zoom-out	εστίαση από μακριά

Δ.2. Αμετάφραστοι Όροι

bitmapped
bottom-up
browser
debugger
dummy
editor
formatter
home item
hypermedia
hypertext
interface
item
multimedia
script
top-down

video

Δ.3. Βοηθήματα για την απόδοση των όρων

Τα παρακάτω λεξικά χρησιμοποιήθηκαν για την απόδοση στα ελληνικά των τεχνικών όρων.

1. "Ελληνικό Γλωσσάρι Πληροφορικής", ΕΠΥ, Αθήνα, Δεκέμβριος 1985.
2. "Σύγχρονο Λεξικό Πληροφορικής", Γ. Γαρίδης και Μ. Δεληγιαννάκης, Δίαυλος, 1989.
3. "Oxford English-Greek Learner's Dictionary", D.N. Stavropoulos και A.S. Hornby, Oxford University Press, 1977.

Ε. ΓΕΝΙΚΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

Η βιβλιογραφία που αναφέρεται στο παράρτημα αυτό είναι ομαδοποιημένη σε τρεις κατηγορίες, ανάλογα με την προέλευσή της. Στο τέλος κάθε βιβλιογραφικής αναφοράς γράφονται μέσα σε παρενθέσεις οι αριθμοί των κεφαλαίων του κυρίως κειμένου στα οποία η αναφορά αυτή χρησιμοποιήθηκε.

Ε.1. Αναφορές του project SAFE

A1. **[HAR89α]** Lynda Hardman, "Navigation in a Hypermedia Environment", SAFE/HYP/OWL-pap/elts_of_nav, Σεπτέμβριος 1989 (9).
A2. **[HAR89β]** Lynda Hardman, "Requirements of Navigation Metaphors in a Hypermedia Learning Environment", SAFE/HYP/OWL-pap/rqts_of_nav_met, Σεπτέμβριος 1989 (9).
A3. **[KIB89]** Mike Kibby, "The Learner's View of Hypermedia", SAFE/HYP/wp/mk080989, Σεπτέμβριος 1989 (9).
A4. **[KOM89]** P. A. M. Kommers, I.A. Ferreira και I. E. Jonker, "HYPERTEXT and Conceptual Mapping", SAFE/HYP/6/HCI-del/mk/fw/lr/pk, Σεπτέμβριος 1989 (3).
A5. **[TAN89α]** Gary Tanner, "Navigating Through Hyperchaos", SAFE/HYP/HCI-pap/GT240789, Ιούλιος 1989 (3/5/6/9).
A6. **[TAN89β]** Gary Tanner, "On the Use of Metaphor", SAFE/HYP/HCI-pap/GT010989, Σεπτέμβριος 1989 (9).
A7. **[WEI89α]** F. Weimar, "Storage Techniques for Hypermedia Material", HYP/6 First Intermediate Report on Hypermedia Structures, SAFE/HYP/6/HCI-del/mk/fw/lr/pk, 1989 (3).
A8. **[WEI89β]** F. Weimar, SAFE/HYP/Phil-Rep/FW06, September 1989 (3).

Ε.2. Πρακτικά συνεδρίων και δημοσιεύσεις σε επιστημονικά περιοδικά

B1. **[AKS88]** Robert M. Akscyn, Donald L. McCracken και Elise A. Yoder, "KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations", CACM, τεύχος 31/7, Ιούλιος 1988 (3/5/7/8/9).
B2. **[CON87]** Jeff. Conklin, "A Survey of Hypertext", MCC Technical Report, τεύχος STP-356-86/2 (9).
B3. **[CUN85]** Brian P. Mc Cune, Richard M. Tong, Jeffrey S. Dean και Daniel G. Shapiro, "RUBRIC: A System for Rule-Based Information Retrieval", IEEE Transactions on Software Engineering, τεύχος SE-11/9, Σεπτέμβριος 1985 (3/5).
B4. **[DYK89]** R.P. Ten Dyke και J.C. Kunz, "Object-Oriented Programming", IBM Systems Journal, τεύχος 28/3, 1989 (2).
B5. **[FRE89]** K. A. Frenkel, "The Next Generation of Interactive Technologies", CACM, τεύχος 32/7, Ιούνιος 1989 (3).
B6. **[FUR86]** G.W. Furnas, "Generalised Fisheye Views", πρακτικά ACM Conference on Human Factors in Computing Systems, 1986 (9).
B7. **[GARG88]** Pankaj K. Garg, "Abstraction Mechanisms in Hypertext", CACM, τεύχος 31/7, Ιούλιος 1988 (7).
B8. **[GARR86]** L. Garrett, K. Smith και N. Meyrowitz, "Intermedia: Issues, Strategies and Tactics in the Design of a Hypermedia Document System", πρακτικά CSCW'86, Δεκέμβριος 1986 (3/5/8/9).
B9. **[HAL88]** Frank G. Halasz, "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems", CACM, τεύχος 31/7, Ιούλιος 1988 (3/6/7/8/9).
B10. **[HAN87]** Robin Hanson, "Toward Hypertext Publishing. Issues and Choices in Database Design", Μάρτιος 1987 (7).
B11. [MAD89] C.M.Madsen, "Using Persistent Objects to Implement an Environment for Cooperative Work", ACM SIGOIS Bulletin, τεύχος 10/4, Δεκέμβριος 1989 (5).
B12. **[NAN88]** J.Nanard, M.Nanard και H.Richy, "Conceptual Documents: A Mechanism for Specifying Active Views in Hypertext", ACM Conf. on Document Processing Systems, 1988 (5/6).

Β13. **[NEU88]** Erich J. Neuhold, Junzhong Gu και Thomas C. Rakow (Institute for Integrated Publication and Information Systems), "An Object-Oriented Transaction Model", 1988 (8).
Β14. **[NIE90]** J. Nielsen, "The Art of Navigating through Hypertext", CACM, τεύχος 33/3, Μάρτιος 1990 (3).
Β15. **[POG85]** A. Poggio, J. Garcia Luna Aceves, E. Craighill, D. Morgan, L. Aguilar, D. Worthington και J. Hight, "CCWS: A Computer-Based, Multimedia Information System", IEEE Computer, Οκτώβριος 1985 (3/9).
Β16. **[REY85]** Joyce K. Reynolds, Jonathan B. Postel, Alan R. Katz, Greg G. Finn και Annette L. DeSchon, "The DARPA Experimental Multimedia Mail System", IEEE Computer, Οκτώβριος 1985 (3).
Β17. **[RIP89]** G. D. Ripley, "DVI - A Digital Multimedia Technology", CACM, τεύχος 32/7, Ιούνιος 1989 (3).
Β18. **[SAK85]** Shiro Sakata και Tetsuo Ueda "A Distributed Interoffice Mail System", IEEE Computer, Οκτώβριος 1985 (3).
Β19. [SHI89] J.J.Shilling και P.F.Sweeney, "Three Steps to Views: Extending the Object-Oriented Paradigm", πρακτικά OOPSLA'89, Οκτώβριος 1989 (6).
Β20. **[SHU89]** Simon Shum, "Enriching the Spatial Environment for Hypertext Browsers: Developing an Approach to Browser Design", PACIS Workshop, Ανοιχτό Πανεπιστήμιο, Μάιος 1989 (9).
Β21. **[SMI87]** Karen E. Smith και Stanley B. Zdonic, "Intermedia: A Case Study of the Differences Between Relational and Object-Oriented Database Systems", πρακτικά OOPSLA'87, 1987 (3/4).
Β22. **[WEG87]** P. Wegner, "Dimensions of Object-Based Language Design", πρακτικά OOPSLA'87, Οκτώβριος 1987 (2).
Β23. **[WOE86]** D. Woelk, W. Kim και W. Luther "An Object-Oriented Approach to Multimedia Databases", ACM 0-89791-191-1/86/0500/0311, 1986 (4).
Β24. **[YAN85]** Nicole Yankelowich, Norman Meyrowitz και Andries van Dam, "Reading and Writing the Electronic Book", IEEE Computer, Οκτώβριος 1985 (3).

Ε.3. Βιβλία

Γ1. **[APP87]** Apple Computers, "HyperCard: User's Guide", Apple Computers, 1987 (3/9).
Γ2. **[BER88]** John Thomas Berry, "C++ Programming", The Waite Group, 1988 (2).
Γ3. **[BOO83]** Booch G., "Software Engineering with Ada", Benjamin-Cummings, 1983 (2).
Γ4. **[COX86]** Brad J. Cox, "Object-Oriented Programming: An Evolutionary Approach", Addison-Wesley, 1986 (2/7).
Γ5. **[DAT84]** C. J. Date, "An Introduction to Database Systems", Addison-Wesley, 1984 (6/8).
Γ6. [KAE86] Ted Kaehler και Dave Patterson, "A Taste of Smalltalk", W.W.Norton, 1986 (2).
Γ7. **[KER84]** Brian W. Kernighan και Rob Pike, "The UNIX Programming Environment", Prentice Hall, 1984 (7).
Γ8. **[KOR86]** H. Korth, A. Silberschatz, "Database System Concepts", 1986 (6/8).
Γ9. **[LOR86]** Lorensen W., "Object-Oriented Design", CRD Software Engineering Guidelines, General Electrics Co., 1986 (2).
Γ10. **[PAR89]** Kamran Parsaye, M. Chingel, S. Khoshafian και H. Wong, "Intelligent Databases", Wiley, 1989 (2/6/9).
Γ11. **[PIN88]** Lewis J. Pinson και Richard S. Wiener, "An Introduction to Object-Oriented Programming and Smalltalk", Addison-Wesley, 1988 (2/5).
Γ12. **[PRE82]** R.S. Pressman, "Software Engineering: A Practitioner's Approach", Mc Graw Hill, 1982 (2).