ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ο ΜΗΧΑΝΙΣΜΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ ΟΝΤΟΤΗΤΩΝ ΓΙΑ ΤΗ ΓΛΩΣΣΑ ΠΑΡΑΣΤΑΣΗΣ ΓΝΩΣΗΣ TELOS

Γιώργος Γεωργιαννάκης

Μεταπτυχιακή Εργασία

Ηράκλειο, Φεβρουάριος 1994

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ο ΜΗΧΑΝΙΣΜΟΣ ΑΠΟΘΗΚΕΥΣΗΣ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ ΟΝΤΟΤΗΤΩΝ ΓΙΑ ΤΗ ΓΛΩΣΣΑ ΠΑΡΑΣΤΑΣΗΣ ΓΝΩΣΗΣ TELOS

Εργασία που υποβλήθηκε από τον
Γιώργο Ι. Γεωργιαννάκη
ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Γιώργος Γεωργιαννάκης
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Γιάννης Βασιλείου, Καθηγητής, Επόπτης

Χρήστος Νικολάου, Αναπληρωτής Καθηγητής, Μέλος

Πάνος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Φεβρουάριος 1994

Ο Μηχανισμός Αποθήκευσης και Διαχείρισης Οντοτήτων για τη Γλώσσα Παράστασης Γνώσης TELOS

Γιώργος Ι. Γεωργιαννάκης

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

ΠΕΡΙΛΗΨΗ

Η TELOS είναι μια γλώσσα αναπαράστασης γνώσης, η οποία παρέχει μηχανισμούς για περιγραφή σημασιολογικών δικτύων και χρησιμοποιεί το οντοκεντρικό μοντέλο αναπαράστασης δεδομένων. Στο ερευνητικό πρόγραμμα Ithaca η TELOS χρησιμοποιείται για την περιγραφή οντοτήτων λογισμικού, οι οποίες αποθηκεύονται σε μια βάση περιγραφής λογισμικού. Τα περιεχόμενα αυτής της βάσης εξετάζονται από μηχανικούς λογισμικού και επιλέγονται για αναχρησιμοποίηση, ή μετατρέπονται έτσι ώστε να αντανακλούν τις μεταβολές που συμβαίνουν στις οντότητες λογισμικού που περιγράφουν.

Αντικείμενο της παρούσας εργασίας είναι η σχεδίαση και υλοποίηση των διαφόρων δομών που χρησιμοποιούνται για την αναπαράσταση και αποθήκευση των δεδομένων της βάσης. Οι δομές αυτές περιλαμβάνουν κατάλληλους μηχανισμούς που επιτρέπουν στις διάφορες εφαρμογές της TELOS την πρόσβαση, εισαγωγή, μεταβολή και διαγραφή των δεδομένων της βάσης, αποτελώντας έτσι τον πυρήνα μέσω του οποίου πραγματοποιούνται οι πράξεις στη βάση δεδομένων.

Κάθε οντότητα διακρίνεται μέσω ενός μοναδικού αναγνωριστικού οντότητας που της παραχωρείται από το μηχανισμό του καταλόγου συστήματος. Ο μηχανισμός αυτός χρησιμοποιείται για την πραγματοποίηση κάθε προσπέλασης στη βάση για οποιαδήποτε οντότητα και διατηρεί πληροφορία για τη θέση και την κατάσταση των οντοτήτων στη μνήμη ή το δίσκο. Ο κατάλογος συστήματος αποτελεί μια δομή αναζήτησης στην οποία η αναζήτηση των οντοτήτων πραγματοποιείται μέσω των αναγνωριστικών οντοτήτων που τις διακρίνουν. Οι μηχανισμοί αναζήτησης είναι αρκετά αποδοτικοί, επιτρέποντας στο σύστημα να παρουσιάζει αξιόλογες επιδόσεις σε τυπικές εφαρμογές.

Οι οντότητες αναπαρίστανται μέσω δομών δεδομένων που είναι κοινές για τα αντίγραφα των οντοτήτων στη μνήμη και το δίσκο αποφεύγοντας έτσι το υπολογιστικό κόστος για μετατροπή οντοτήτων από τη μια αναπαράσταση στην άλλη κατά τη μεταφορά των οντοτήτων μεταξύ μνήμης και δίσκου. Οι οντότητες οργανώνονται στο δίσκο σε δομές που ονομάζονται τμήματα οντοτήτων και έχουν μέγεθος ίσο με το μέγεθος των σελίδων δίσκου (disk pages). Οι οντότητες που βρίσκονται σε ένα τμήμα οντοτήτων περιγράφονται από ένα κοινό τύπο οντοτήτων. Η μονάδα μεταφοράς μεταξύ δίσκου και μνήμης είναι το τμήμα οντοτήτων. Κατά τη μεταφορά μιας οντότητας από το δίσκο στη μνήμη, μεταφέρεται ολόκληρο το τμήμα οντοτήτων που περιέχει τη συγκεκριμένη οντότητα, προκαλώντας έτσι προανάκληση όλων των υπόλοιπων οντοτήτων που βρίσκονται στο τμήμα αυτό.

Οι χρήστες του συστήματος προσδιορίζουν τις οντότητες χρησιμοποιώντας λογικά ονόματα, ενώ το σύστημα χρησιμοποιεί αναγνωριστικά οντοτήτων για κάθε πράξη σε οντότητες. Η μετάφραση του λογικού ονόματος στο αντίστοιχο αναγνωριστικό οντότητας για κάθε οντότητα, αλλά και η αντίστροφη διαδικασία πραγματοποιούνται από τον κατάλογο λογικών ονομάτων. Ο κατάλογος αυτός παρέχει μηχανισμούς αναζήτησης προκειμένου να γίνεται αποδοτικότερη η μετάφραση αυτή.

Οι δομές αποθήκευσης για τον κατάλογο συστήματος, τα τμήματα οντοτήτων και τον κατάλογο λογικών ονομάτων, τοποθετούνται σε κατάλληλους μηχανισμούς κρυφής μνήμης στη μνήμη του υπολογιστή. Οι μηχανισμοί αυτοί επιτρέπουν άμεση εκκίνηση του συστήματος σε κάθε εφαρμογή, ανεξάρτητα από το μέγεθος της βάσης δεδομένων που χρησιμοποιείται και βελτιώνουν σημαντικά τις επιδόσεις του συστήματος, οι οποίες είναι ανάλογες των επιδόσεων αρκετών εμπορικών οντοκεντρικών συστημάτων διαχείρισης βάσεων δεδομένων.

Επόπτης : Ιωάννης Βασιλείου , Καθηγητής

A Storage and Memory Management Mechanism for Objects in TELOS

George Y. Yeorgiannakis

Master of Science Thesis

Department of Computer Science
University of Crete

ABSTRACT

TELOS is an object oriented language for knowledge representation which provides efficient mechanisms to describe semantic networks. It is used within the ITHACA project for the description of software components that are stored in a Software Information Base (SIB). Software engineers browse through the contents of the SIB and select software components for reuse or update already existing components.

The purpose of this thesis is the design and implementation of the various structures that are used for storing the contents of the SIB. These structures are integrated with appropriate mechanisms that provide TELOS applications with an interface to access, insert or update the contents of the SIB, forming a kernel through which all operations in the SIB are performed.

Each object is identified by a unique object identifier which is assigned to the object by the system catalogue mechanism. This mechanism implements every access to the SIB for each object and maintains information about the state of the objects and their placement in memory and on disk. The system catalogue is essentially an access mechanism where objects are accessed through use of their unique object identifiers. The access methods are efficient, allowing the system to exhibit competitive performance in typical applications.

Objects on disk and in memory are represented by identical data structures, avoiding thus the computational cost for transforming them from one format to another when they are exchanged between memory and disk. Objects are organized on disk in object blocks, whose size is equal to the disk block size. Objects are read/stored from/on files in a disk block basis. Objects residing in the same block belong to the same object type. When an object is read from disk, the whole object block where the object resides, is fetched in memory, therefore all other objects residing in the same object block are prefetched.

The SIB users identify objects by means of logical names while the runtime system uses the object identifiers for every operation performed on objects. Logical name translation to the corresponding object identifier and vice versa is performed by the symbol table. This table exploits access mechanisms that facilitate efficient translation between logical names and object identifiers.

The storage structures for the object blocks, the system catalogue and the symbol table in memory, reside in corresponding caches. The cache mechanisms enable immediate system start-up regardless of the SIB size and enhance system performance which compares to that of commercial object oriented data base systems.

Supervisor : Yannis Vassiliou, Professor

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Γιάννη Βασιλείου που μου έδωσε την ευκαιρία να ασχοληθώ με τη συγκεκριμένη εργασία, αλλά και για την καθοδήγησή του κατά τη διάρκειά της.

Θα ήθελα ακόμη να ευχαριστήσω τα μέλη της επιτροπής εξέτασης της μεταπτυχιακής μου εργασίας καθηγητές κ. κ. Πάνο Κωνσταντόπουλο και Χρήστο Νικολάου για τις χρήσιμες συμβουλές και παρατηρήσεις που συνέβαλαν στην τελική διαμόρφωση του παρόντος κειμένου.

Επίσης, θα ήθελα να ευχαριστήσω όλα τα μέλη της ερευνητικής ομάδας του ITHACA για την αξιοθαύμαστη συνεργασία τους, αλλά και τη συντροφικότητα τους. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον δρ. Martin Dörr για τις πολύτιμες συμβουλές του και τις διάφορες συζητήσεις που βοήθησαν σημαντικά την ανάπτυξη της εργασίας αυτής και τη Σταυρούλα Κιζλαρίδου που ήταν ακούραστη και στενή συνεργάτης σε διάφορα από τα τμήματα της εργασίας.

Θα ήθελα να ευχαριστήσω τον Νίκο Τσατσάκη για τη συμβολή του στη διόρθωση αυτού του κειμένου και τη φιλία του, όπως επίσης και την Μάγδα Χατζάκη.

Ακόμη, ευχαριστώ τη Μαρία Καραβασίλη για την συμπαράσταση της κατά τη διάρκεια των σπουδών μου και φυσικά τους γονείς μου για την υπομονή και υποστήριξη που μου παρείχαν.

Τέλος, ευχαριστώ το Πανεπιστήμιο Κρήτης και το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την οικονομική και υλικοτεχνική υποστήριξη που μου παρείχαν κατά τη διάρκεια των σπουδών μου.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Το περιεχόμενο της εργασίας

Η παρούσα εργασία ασχολείται με τη σχεδίαση και υλοποίηση ενός μηχανισμού διαχείρισης και αποθήκευσης οντοτήτων σε μια Βάση Περιγραφής Λογισμικού (ΒΠΛ - SIB Software Information Base). Η ΒΠΛ περιέχει περιγραφές λογισμικού που αντιστοιχούν στα τρία στάδια του κύκλου ζωής του λογισμικού: στην ανάλυση απαιτήσεων, στο σχεδιασμό και την υλοποίηση. Επίσης, περιέχει περιγραφές για τις διάφορες σχέσεις μεταξύ των περιγραφών. Η εργασία αυτή πραγματοποιήθηκε στο Ινστιτούτο Πληροφορικής του ΙΤΕ στα πλαίσια του ερευνητικού προγράμματος Ithaca[1], αντικείμενο του οποίου είναι η αναχρησιμοποίηση λογισμικού.

Η περιγραφή της δομής της ΒΠΛ γίνεται στην γλώσσα TELOS [**?**], η οποία αποτελεί μια οντοκεντρική γλώσσα παράστασης γνώσης. Στη γλώσσα TELOS οι περιγραφές οντοτήτων λογισμικού και οι πιθανές σχέσεις μεταξύ τους αποτελούν ανεξάρτητες οντότητες. Στη Β.Π.Λ. αποθηκεύονται οι οντότητες TELOS, τα λογικά ονόματά τους - στην TELOS ο χρήστης μπορεί να δώσει ένα λογικό όνομα σε κάθε οντότητα που ορίζει - και τον κατάλογο συστήματος, μια δομή που παρέχει πληροφορία για το είδος της κάθε οντότητας, την θέση που βρίσκεται στο δίσκο ή/και στη μνήμη και την κατάστασή της.

1.2 Η γλώσσα παράστασης γνώσης TELOS

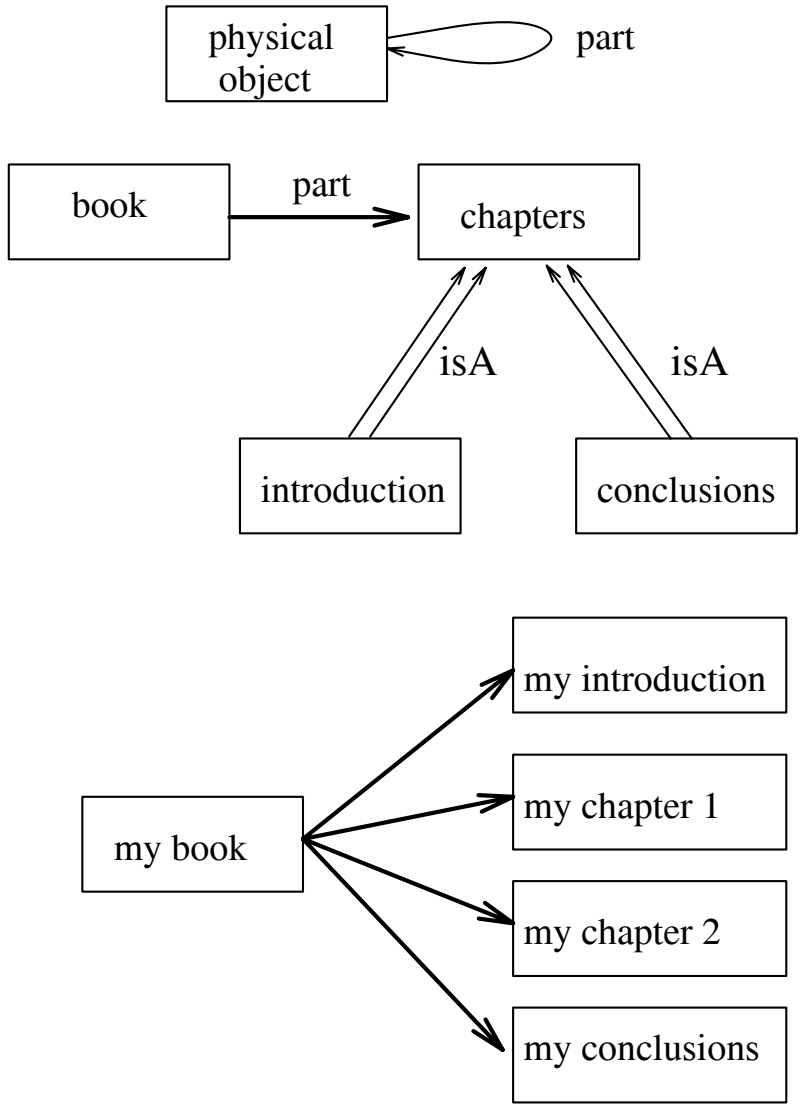
Η TELOS είναι μια γλώσσα που αναπτύχθηκε για να υποστηρίξει τους μηχανικούς λογισμικού στην ανάπτυξη πληροφοριακών συστημάτων καθ' όλη τη διάρκεια του ''κύκλου ζωής'' του λογισμικού. Η γλώσσα αυτή χρησιμοποιεί αρχές από το χώρο αναπαράστασης

[1] Integrated tool for highly advanced computer applications

γνώσης, καθώς η ανάπτυξη πληροφοριακών συστημάτων είναι μια διεργασία που απαιτεί ειδική γνώση, η οποία πρέπει να αναπαρίσταται με ευχέρεια και πληρότητα.

Όπως έχουμε ήδη αναφέρει, η TELOS ακολουθεί το οντοκεντρικό μοντέλο αναπαράστασης γνώσης. Η βάση γνώσης αποτελείται από δομημένα οντότητες που κατηγοριοποιούνται σε δυο ειδών βασικές κλάσεις οντοτήτων: τις *ανεξάρτητες οντότητες* (Individuals) και τα *γνωρίσματα* (Attributes). Οι οντότητες και τα γνωρίσματα τυγχάνουν ομοιόμορφης διαχείρισης από τους μηχανισμούς δόμησης της βάσης γνώσης και αποτελούν τις βασικές δομές της γλώσσας TELOS. Οι βασικές αυτές δομές στην TELOS ονομάζονται *προτάσεις* (propositions). Μια βάση γνώσης στην TELOS αποτελείται από τέτοιες *προτάσεις* οι οποίες ομαδοποιούνται σε κλάσεις και συνδέονται με άλλες προτάσεις μέσω γνωρισμάτων. Με τον τρόπο αυτό δημιουργούνται νέες προτάσεις. Σύμφωνα με το οντοκεντρικό μοντέλο, κάθε τέτοια νέα πρόταση αποτελεί ξεχωριστή οντότητα. Κάθε οντότητα έχει ένα μοναδικό, παραγόμενο από το σύστημα αναγνωριστικό όνομα, ενώ μπορεί να έχει και κάποιο λογικό όνομα που παρέχει ο χρήστης.

Επιπλέον, η TELOS χρησιμοποιεί κάποιους βασικούς μηχανισμούς δόμησης που χρησιμοποιούνται σε βάσεις παράστασης γνώσης και σημασιολογικά μοντέλα δεδομένων, παρέχοντας τη δυνατότητα της βαθμιαίας και ομαλής περιγραφής της πληροφορίας στο επίπεδο ακρίβειας που απαιτείται. Οι μηχανισμοί αυτοί είναι οι οποίοι αντιστοιχούν στις σχέσης *ταξινόμησης-κατηγοριοποίησης* (classification-instantiation), *σύνθεσης-ανάλυσης* (aggregation-decomposition) και *γενίκευσης-εξειδίκευσης* (generalization-specialization), δημιουργούν ιεραρχίες και είναι ανεξάρτητοι μεταξύ τους. Οι ιεραρχίες ταξινόμησης και γενίκευσης μπορεί να εκτείνονται σε πολλά επίπεδα. Με αυτό τον τρόπο παρέχεται στο χρήστη η δυνατότητα να περιγράφει σταδιακά και στο βαθμό ακρίβειας που αυτός επιθυμεί τα μοντέλα περιγραφής πληροφοριακών συστημάτων που δημιουργεί. Τα μοντέλα αυτά μπορούν να αποκτήσουν ιεραρχίες οντοτήτων σε διαφορετικά επίπεδα αφαίρεσης καθώς και ομαδοποιήσεις οντοτήτων που έχουν κοινά χαρακτηριστικά. Οι μηχανισμοί αναπαράστασης της TELOS δημιουργούν ένα σημασιολογικό δίκτυο. Οι κόμβοι και οι σύνδεσμοι του δικτύου αντιστοιχούν στις οντότητες και τα γνωρίσματα της TELOS αντίστοιχα. Στο σχήμα **??** παρουσιάζεται ένα μοντέλο από το φυσικό κόσμο που μπορεί να περιγραφεί σε TELOS όπου διακρίνονται τρία επίπεδα της ιεραρχίας ταξινόμησης. Όλες οι οντότητες ανήκουν στη κλάση **physical object** η οποία βρίσκεται στο ανώτερο επίπεδο ταξινόμησης. Οι κλάσεις οντοτήτων **book, chapter, introduction** και **conclusions** αποτελούν περιπτώσεις για τη κλάση **physical object**. Στο κατώτερο επίπεδο ταξινόμησης βρίσκονται οι περιπτώσεις των κλάσεων του μεσαίου επιπέδου. Επίσης, διακρίνεται η χρήση των μηχανισμών γενίκευσης-εξειδίκευσης, για τις κλάσεις **introduction** και **conclusions** που αποτελούν εξειδικεύσεις της



Σχήμα 1.1: Ένα μοντέλο TELOS

Τα επίπεδα της ιεραρχίας ταξινόμησης χωρίζονται με διακεκομμένες γραμμές. Στο σχήμα αυτό επίσης μπορούμε να διακρίνουμε την εφαρμογή του μηχανισμού γενίκευσης-εξειδίκευσης μεταξύ της κλάσης **chapter** και των υποκλάσεών της **introduction** και **conclusions** και την εφαρμογή του μηχανισμού σύνθεσης μέσω του γνωρίσματος **part** ανάμεσα στις κλάσεις **book** και **chapter**.

κλάσης **chapter**. Τέλος με το μηχανισμό απόδοσης γνωρίσματος συνδέεται η κλάση **book** με τη κλάση **chapter** μέσω του γνωρίσματος **part**.

1.3 Δομές Αποθήκευσης για την TELOS

Μπορούμε να διακρίνουμε τρεις διαφορετικές δομές πληροφορίας που βρίσκονται σε μια βάση γνώσης που έχει περιγραφεί σε TELOS: τις ίδιες τις οντότητες οι οποίες αναπαρίστανται από επιμέρους κλάσεις υλοποιημένες σε C++ [**?**], τα λογικά ονόματα αυτών των οντοτήτων καθώς και τον κατάλογο συστήματος (system catalogue). Ο κατάλογος συστήματος περιέχει πληροφορία για τις διάφορες οντότητες η οποία δεν χρειάζεται να παρουσιάζεται (transparent information), όπως την τοποθέτηση των οντοτήτων στη μνήμη ή το δίσκο και την κατάσταση που βρίσκεται μια οντότητα: νέα, αλλαγμένη, διαγραμμένη κ.λ.π.

Κατά τον σχεδιασμό και την υλοποίηση του μηχανισμού αποθήκευσης οντοτήτων, του μηχανισμού για τον κατάλογο συστήματος και του μηχανισμού του καταλόγου λογικών ονομάτων, στα πλαίσια της παρούσας εργασίας, χρειάστηκε να μελετήσουμε τη χρήση αντίστοιχων μηχανισμών για συστήματα διαχείρισης βάσεων δεδομένων και να επιλέξουμε και να υλοποιήσουμε αποδοτικές μεθόδους αναζήτησης για τους παραπάνω μηχανισμούς.

Σκοπός της εργασίας ήταν να δημιουργηθεί ένας αποτελεσματικός μηχανισμός αναπαράστασης και αποθήκευσης της πληροφορίας που κρατείται στη βάση περιγραφής λογισμικού και η επίτευξη αξιόλογων επιδόσεων από το σύστημα διαχείρισης της πληροφορίας αυτής. Έτσι, επιλέξαμε τη δημιουργία μηχανισμών αποθήκευσης, οι οποίοι αξιοποιούν τη σημασιολογία των δομών της γλώσσας TELOS και μπορούν να προσφέρουν σημαντικές επιδόσεις στο σύστημα που προκύπτει, όντας ειδικά σχεδιασμένοι για αυτό.

Επίσης δημιουργήσαμε μηχανισμούς κρυφής μνήμης για τις διάφορες δομές αποθήκευσης, οι οποίοι βελτιώνουν τις επιδόσεις του συστήματος ακόμα και για πολύ μεγάλες βάσεις. Η υποστήριξη πολλών ταυτόχρονων δοσοληψιών στη βάση, πραγματοποιείται μέσω του πρωτοκόλλου κλειδώματος δύο φάσεων. Στη συνέχεια της παραγράφου, επιχειρείται μια απλή παρουσίαση των μηχανισμών αυτών.

Στην υλοποίηση της TELOS για το ερευνητικό πρόγραμμα Ithaca δημιουργούνται πέντε βασικοί τύποι οντοτήτων για την αναπαράσταση οντοτήτων/γνωρισμάτων. Οι οντότητες και τα γνωρίσματα παρουσιάζονται με δυο διαφορετικούς τύπους, ανάλογα με το αν πρόκειται για ατομικές οντότητες/γνωρίσματα ή κλάσεις οντοτήτων/γνωρισμάτων. Επίσης, έχει υλοποιηθεί ένας τύπος trigger, για να παρουσιάζει περιορισμούς που αφορούν τη συμπεριφορά των διαφόρων οντοτήτων. Οι οντότητες κάθε βάσης αποτελούν κατηγοριοποιήσεις οντοτήτων που είναι εγγενείς στο σύστημα και χρησιμοποιούνται για την αναπαράσταση όλων των

δομών που δημιουργούνται μέσω της γλώσσας αναπαράστασης TELOS. Οι οντότητες έχουν την ίδια μορφή στη μνήμη και το δίσκο, διευκολύνοντας και απλοποιώντας τη μεταφορά τους από το ένα μέσο στο άλλο. Οι οντότητες ίδιου τύπου ομαδοποιούνται σε blocks που έχουν μέγεθος ίδιο με αυτό του ελάχιστου χώρου που μπορεί να διαβαστεί/γραφτεί από/στο δίσκο (disk page). Επιτυγχάνεται έτσι το διάβασμα/γράψιμο μιας οντότητας με μια μόνο προσπέλαση στο δίσκο, ενώ οι ενταμιευτές (buffers) του συστήματος διαχείρισης αρχείων χρησιμοποιούνται χωρίς προβλήματα κατακερματισμού (fragmentation).

Κάθε οντότητα στην TELOS έχει ένα μοναδικό αναγνωριστικό όνομα (το οποίο σε οντοκεντρικά συστήματα συνήθως ονομάζεται OID (object identifier) ή ID) που παράγεται από το σύστημα και χαρακτηρίζει την οντότητα. Επιπλέον, κάθε οντότητα στην TELOS έχει ένα λογικό όνομα (στην περίπτωση που ο χρήστης δεν ορίσει κάποιο λογικό όνομα για κάποια οντότητα, το σύστημα δημιουργεί και αποθηκεύει ένα). Όταν κάποιος χρήστης θέλει μέσω μιας εφαρμογής να αναφερθεί σε κάποια οντότητα, χρησιμοποιεί το λογικό της όνομα, ενώ το σύστημα της TELOS χρησιμοποιεί το αναγνωριστικό της οντότητας αυτής. Χρειάζεται λοιπόν ένας πίνακας μετάφρασης (translation table) από λογικά ονόματα σε αναγνωριστικά ονόματα συστήματος και αντίστροφα. Για κάθε οντότητα δημιουργείται μια πλειάδα (tuple) που περιέχει το αναγνωριστικό και το λογικό όνομα της οντότητας. Ταξινομημένες κατά αύξον αναγνωριστικό όνομα, οι πλειάδες αυτές αποτελούν τη πρώτη δομή του καταλόγου ονομάτων (symbol table). Η δομή αυτή εξυπηρετεί τη μετάφραση ενός αναγνωριστικού για κάποια οντότητα, στο αντίστοιχο λογικό όνομα για την οντότητα αυτή. Εφαρμόζοντας linear hashing στα λογικά ονόματα των οντοτήτων, δημιουργείται η δεύτερη δομή του καταλόγου ονομάτων, η οποία χρησιμοποιείται για τη μετάφραση του λογικού ονόματος κάποιας οντότητας στο μοναδικό αναγνωριστικό για την οντότητα αυτή.

Κάθε σύστημα διαχείρισης βάσεων δεδομένων διατηρεί έναν εσωτερικό κατάλογο, ο οποίος περιέχει περιγραφές για τα δεδομένα που αποθηκεύονται, και τη θέση στην οποία βρίσκονται στην κύρια μνήμη και το δίσκο. Η δομή αυτή συνήθως ονομάζεται κατάλογος συστήματος (system catalogue) και το σύστημα τη χρησιμοποιεί όποτε χρειάζεται να αναφερθεί σε κάποια οντότητα ή να πάρει πληροφορία για την κατάσταση μιας οντότητας. Στην περίπτωση του συστήματος για την TELOS, ο κατάλογος συστήματος είναι ένας έμμεσος (indirect) πίνακας από τμήματα και προσπελαύνεται-δεικτοδοτείται με βάση τα αναγνωριστικά συστήματος για τις οντότητες. Η δομή του καταλόγου συστήματος περιέχει διάφορες ρουτίνες που αναλαμβάνουν τη δημιουργία μοναδικών αναγνωριστικών για τις οντότητες, φροντίζοντας ταυτόχρονα να παραμένει πυκνός αυτός ο πίνακας αναγνωριστικών, χωρίς δηλαδή να δημιουργούνται κενά λόγω προηγούμενων διαγραφών οντοτήτων. Ο κατάλογος συστήματος χωρίζεται σε δύο υποκαταλόγους, έναν για την περιγραφή της σταθερής

κατάστασης των οντοτήτων στο δίσκο (persistent objects) και έναν για την περιγραφή της κατάστασης των οντοτήτων που βρίσκονται στη μνήμη.

Η συνολική αρχιτεκτονική του συστήματος διαχείρισης οντοτήτων στη μνήμη και το δίσκο παρουσιάζεται στο σχήμα **??** . Οι διάφορες εφαρμογές έχουν άμεση προσπέλαση στον κατάλογο συμβόλων και τον κατάλογο συστήματος, αλλά όχι και στη δομή διαχείρισης οντοτήτων. Ουσιαστικά ο μηχανισμός του καταλόγου συστήματος αποτελεί τη μοναδική δομή μέσω της οποίας γίνεται η διαχείριση των οντοτήτων για τις διάφορες εφαρμογές οι οποίες μπορούν να προσπελαύνουν τις οντότητες μιας βάσης μονάχα μέσω κλήσεων στη δομή του καταλόγου συστήματος. Επίσης, προκειμένου να βελτιωθεί η απόδοση του μηχανισμού διαχείρισης οντοτήτων χρησιμοποιούνται μηχανισμοί κρυφής μνήμης για τον κατάλογο συστήματος, τον κατάλογο συμβόλων καθώς και τις ίδιες τις οντότητες.

1.4 Οργάνωση της εργασίας

Στο δεύτερο κεφάλαιο αυτής της εργασίας αναφέρουμε τις γενικές αρχές των οντοκεντρικών συστημάτων διαχείρισης βάσεων δεδομένων και παρουσιάζουμε μια ανασκόπηση διαφόρων σύγχρονων τέτοιων συστημάτων.

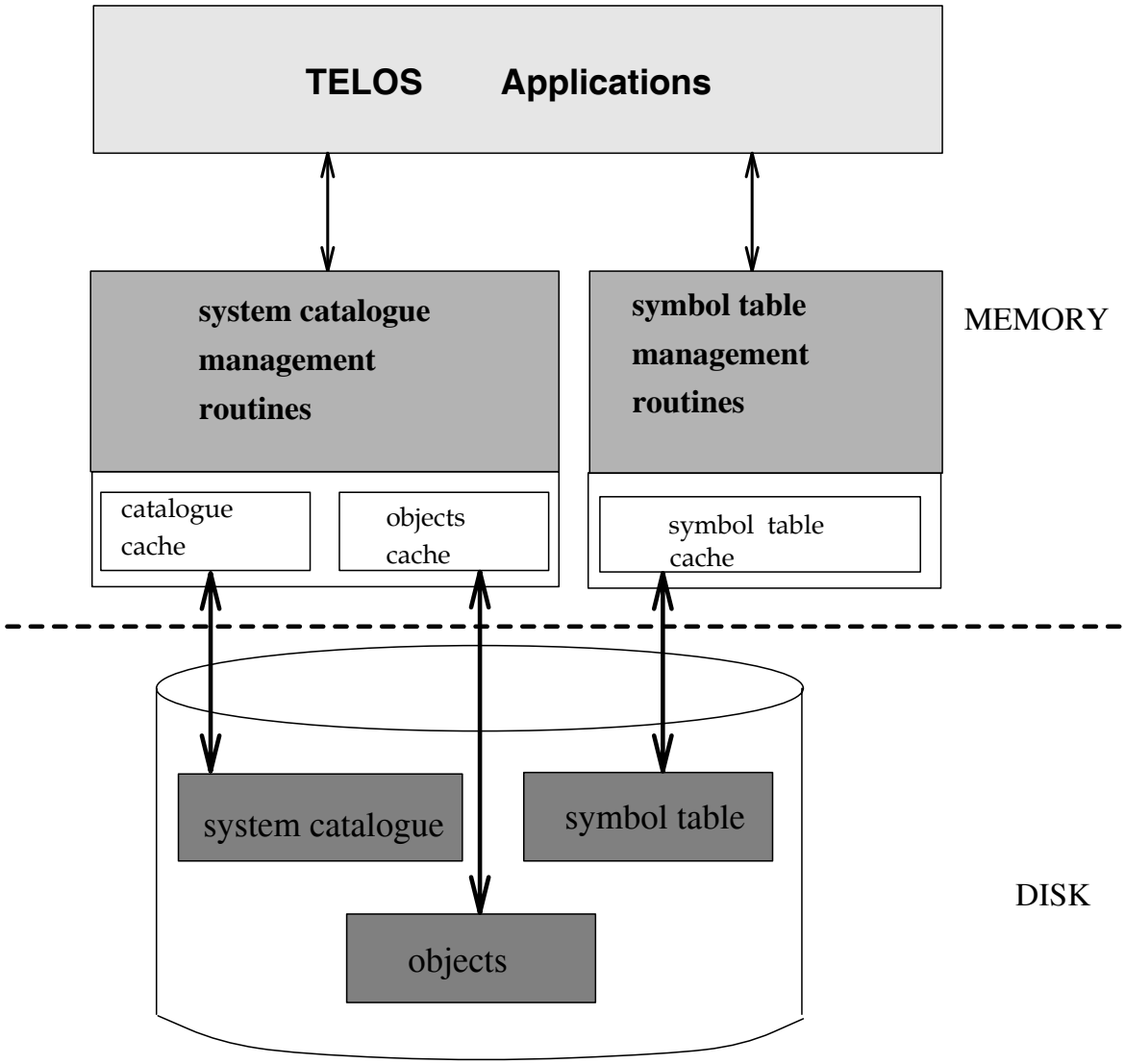
Στο τρίτο κεφάλαιο παρουσιάζουμε τους διάφορους τύπους οντοτήτων που χρησιμοποιήθηκαν για τη τρέχουσα υλοποίηση της γλώσσας αναπαράστασης γνώσης TELOS, τις σχέσεις μεταξύ αυτών των οντοτήτων καθώς και την αναπαράστασή τους.

Στο τέταρτο κεφάλαιο παρουσιάζουμε την αρχιτεκτονική και τη λειτουργία του καταλόγου συστήματος για τις διάφορες οντότητες που μπορεί να βρίσκονται σε μια ΒΠΛ. Ακόμα παρουσιάζουμε την οργάνωση των οντοτήτων σε δομές που βρίσκονται στη μνήμη και στο δίσκο και έναν απλό μηχανισμό που χρησιμοποιείται για να παρέχει στο σύστημα ταυτόχρονη προσπέλαση για ερωτήσεις ή αλλαγές στη ΒΠΛ σε πολλούς χρήστες (concurrency control).

Στο πέμπτο κεφάλαιο παρουσιάζουμε την αρχιτεκτονική και την οργάνωση του καταλόγου συμβόλων για τις διάφορες οντότητες μιας ΒΠΛ.

Στα έκτο κεφάλαιο παρουσιάζουμε το γενικό μηχανισμό κρυφής μνήμης που χρησιμοποιείται για τις δομές αναπαράστασης των οντοτήτων, του καταλόγου συστήματος και του καταλόγου συμβόλων. Επίσης, παρουσιάζουμε τους επιμέρους ειδικότερους μηχανισμούς κρυφής μνήμης για κάθε μια από τις προηγούμενες δομές καθώς και τις διάφορες παραμέτρους και αλγόριθμους αντικατάστασης για τους μηχανισμούς αυτούς.

Στο έβδομο κεφάλαιο παρουσιάζουμε μια αξιολόγηση και μετρήσεις επιδόσεων του συστήματος αναφορικά με ευρέως χρησιμοποιούμενα προγράμματα αξιολόγησης επιδόσεων



Σχήμα 1.2: Η αρχιτεκτονική του συστήματος διαχείρισης οντοτήτων

Οι εφαρμογές στο περιβάλλον της TELOS προσπελαύνουν την αποθηκευμένη πληροφορία στο δίσκο μέσω των υποσυστημάτων διαχείρισης του καταλόγου συστήματος και του καταλόγου συμβόλων. Τα λογικά ονόματα για τις οντότητες προσπελαύνονται μέσω του υποσυστήματος διαχείρισης του καταλόγου συμβόλων, ενώ το υποσύστημα του καταλόγου συστήματος αποτελεί το υποσύστημα διαχείρισης οντοτήτων. Τα υποσυστήματα αυτά χρησιμοποιούν μηχανισμούς κρυφής μνήμης όπου κρατούνται οι διάφορες δομές που μεταφέρονται από το δίσκο στη μνήμη.

(benchmarks). Επίσης, αναφέρονται οι επιδόσεις άλλων αντίστοιχων συστημάτων για τα ίδια προγράμματα.

Τέλος, στο όγδοο κεφάλαιο παρουσιάζουμε τα συμπεράσματα που αποκομίσαμε κατά τη διάρκεια της παρούσας εργασίας και προτείνουμε επεκτάσεις και τροποποιήσεις που μπορούν να βελτιώσουν την απόδοση του συστήματος.

1.5 Η υλοποίηση

Η παρούσα εργασία πραγματοποιήθηκε στα πλαίσια του ερευνητικού προγράμματος ITHACA στο Ινστιτούτο Πληροφορικής του ΙΤΕ.

Η υλοποίηση των διαφόρων δομών και συναρτήσεων έγινε σε γλώσσα C++ για το λειτουργικό σύστημα UNIX.

Οι διάφορες δομές αποθηκεύονται στο δίσκο χρησιμοποιώντας δυαδικά αρχεία (binary files) του UNIX οδηγώντας στη παραγωγή ενός συστήματος το οποίο μπορεί να λειτουργήσει σε πολλά διαφορετικά συστήματα UNIX χωρίς να χρειάζεται ειδικές μετατροπές ή ειδικό hardware. Για τη διαχείριση των αρχείων της βάσης χρησιμοποιούνται οι τυποποιημένες συναρτήσεις κλήσης του λειτουργικού συστήματος (system calls) που παρέχει το λειτουργικό σύστημα UNIX όπως περιγράφονται στο [**?**].

Η υλοποίηση της εργασίας χρειάστηκε συνολικά 2 χρόνια. Στο διάστημα αυτό δημιουργήθηκαν διάφορες ενδιάμεσες εκδόσεις του κώδικα μια και η πορεία κατασκευής του περιλάμβανε ταυτόχρονα και τη σχεδίαση για τα διάφορα υποτμήματα. Συνολικά η τρέχουσα έκδοση του κώδικα αποτελείται από 16.000 γραμμές.

Το σύστημα αποθήκευσης που υλοποιήθηκε σε αυτή την εργασία έχει χρησιμοποιηθεί για την δημιουργία μεγάλου πλήθους βάσεων σε εφαρμογές που δεν αφορούν αναχρησιμοποίηση λογισμικού (όπως θέματα εργασίας σε μαθήματα και ένα Μουσειακό Σύστημα Τεκμηρίωσης αντικειμένων) εφόσον η γλώσσα TELOS παρέχει ένα ισχυρό και πλήρη μηχανισμό αναπαράστασης, πράγμα που αποτελεί ικανό έλεγχο ορθότητας του μηχανισμού αποθήκευσης.

Αν και το μέγεθος της βάσης που μπορεί να υποστηρίξει το τρέχον σύστημα είναι της τάξεως 2^{30} οντοτήτων, το σύστημα έχει ελεγθεί πλήρως μέχρι στιγμής για βάσεις που περιέχουν μέχρι 850.000 οντότητες.

Οι έλεγχοι για τις επιδόσεις του συστήματος που πραγματοποιήθηκαν, έδειξαν ότι οι επιδόσεις του συστήματος είναι τουλάχιστον εφάμιλλες των επιδόσεων διαφόρων εμπορικών οντοκεντρικών συστημάτων διαχείρισης βάσεων δεδομένων.

Κεφάλαιο 2

Οντοκεντρικά Συστήματα Διαχείρισης Βάσεων Δεδομένων

2.1 Οντοκεντρικές Βάσεις Δεδομένων

Η συνεχής ανάπτυξη όλο και πιο σύνθετων και μεγαλύτερων σε όγκο εφαρμογών με μεγάλο κόστος συντήρησης και ανάπτυξης αυξάνει τις απαιτήσεις που καλείται να καλύψει η τεχνολογία βάσεων δεδομένων. Έτσι, την τελευταία τριακονταετία είχαμε τέσσερεις γενιές εξέλιξης στην τεχνολογία αυτή: τα συστήματα αρχείων, τα ιεραρχικά και CODASYL συστήματα και τέλος τα σχεσιακά συστήματα βάσεων δεδομένων.

Οι περιορισμοί της συμβατικής τεχνολογίας βάσεων δεδομένων οδηγούν στην ανάπτυξη της πέμπτης γενιάς τεχνολογίας βάσεων δεδομένων. Η συμβατική τεχνολογία δεν μπορεί να παρέχει ικανοποιητική αναπαράσταση σύνθετων οντοτήτων ή πολλαπλών διαφορετικών τύπων δεδομένων όπως εικόνες, ήχο ή κείμενα. Αρχές σημασιολογικής αναπαράστασης όπως η γενίκευση-εξειδίκευση, σύνθεση-απόδοση γνωρίσματος δυσχεραίνουν την επίδοση τέτοιων συστημάτων. Οι αρχές αυτές συνήθως υποστηρίζονται μέσω μεθοδολογιών που κατασκευάζονται ειδικά για συγκεκριμένες εφαρμογές. Οι νέες εφαρμογές όπως συστήματα σχεδίασης μέσω υπολογιστή, συστήματα αυτοματισμού γραφείου, βάσεις γνώσης και επιστημονικές εφαρμογές συνεχώς απαιτούν αλλαγή του εννοιολογικού σχήματος της βάσης δεδομένων, πράγμα που δεν μπορούν να υποστηρίξουν πλήρως οι σχεσιακές βάσεις δεδομένων [**?**] [**?**] [**?**].

Τα οντοκεντρικά συστήματα βάσεων δεδομένων αποτελούν μία νέα γενιά συστημάτων βάσεων δεδομένων τα οποία καλύπτουν τις αδυναμίες των συμβατικών συστημάτων βάσεων δεδομένων που μόλις αναφέραμε. Τα συστήματα αυτά στηρίζονται στο *οντοκεντρικό μοντέλο αναπαράστασης δεδομένων.* Το μοντέλο αυτό βασίζεται σε θεμελιώδεις αρχές που συναντάμε σε οντοκεντρικές γλώσσες προγραμματισμού και γλώσσες αναπαράστασης

γνώσης. Όπως παρατηρείται στα [**?**] [**?**] αν και έχουν αναπτυχθεί πολλά οντοκεντρικά συστήματα βάσεων δεδομένων, μέχρι στιγμής δεν έχει οριστεί αυστηρά ένα καθολικό οντοκεντρικό μοντέλο αναπαράστασης δεδομένων του οποίου περιπτώσεις να είναι τα μοντέλα αναπαράστασης των συστημάτων αυτών. Μπορούμε όμως να διακρίνουμε κάποια χαρακτηριστικά που διαμορφώνουν ένα βασικό οντοκεντρικό μοντέλο αναπαράστασης δεδομένων που υποστηρίζεται από όλα τα οντοκεντρικά συστήματα βάσεων δεδομένων :

- **Οντότητα - Αναγνωριστικό οντότητας:** κάθε αντικείμενο του πραγματικού κόσμου αποτελεί μια ξεχωριστή οντότητα. Κάθε τέτοια οντότητα διακρίνεται μέσω ενός μοναδικού αναγνωριστικού.

- **Γνωρίσματα - Μέθοδοι:** κάθε οντότητα μπορεί να έχει μία κατάσταση και μία συμπεριφορά. Η κατάσταση της οντότητας περιγράφεται από τις τιμές των γνωρισμάτων για την οντότητα αυτή. Η τιμή ενός γνωρίσματος για μία οντότητα αποτελεί μία νέα οντότητα (η οποία με τη σειρά της μπορεί να έχει διάφορα γνωρίσματα). Η συμπεριφορά μιας οντότητας καθορίζεται από το σύνολο των μεθόδων (συναρτήσεων) οι οποίες χρησιμοποιούν ή και μεταβάλλουν τα γνωρίσματα της οντότητας.

- **Κλάσεις οντοτήτων:** Οι κλάσεις οντοτήτων είναι γενικές οντότητες οι οποίες χρησιμοποιούνται για να ομαδοποιούν ειδικότερες οντότητες οι οποίες έχουν κοινά σύνολα γνωρισμάτων και μεθόδων. Κάθε οντότητα κατηγοριοποιείται κάτω από τουλάχιστον μία κλάση οντοτήτων αποτελώντας *περίπτωση* (instance) αυτής της κλάσης οντοτήτων. Η σχέση της κατηγοριοποίησης ανάμεσα στις οντότητες δημιουργεί μία ιεραρχία κατηγοριοποίησης.

- **Ιεραρχία κλάσεων οντοτήτων - Κληρονόμηση ιδιοτήτων:** Στα οντοκεντρικά συστήματα ο χρήστης μπορεί να δημιουργήσει κλάσεις οντοτήτων οι οποίες παράγονται από άλλες κλάσεις οντοτήτων και κληρονομούν όλα τα γνωρίσματα των τελευταίων, δημιουργώντας έτσι μια ιεραρχία κλάσεων οντοτήτων. Στην περίπτωση αυτή, η κλάση οντοτήτων που παράγεται ονομάζεται *υποκλάση* (ή εξειδίκευση) της κλάσης οντοτήτων που προϋπήρχε ενώ η τελευταία ονομάζεται *υπερκλάση* (ή γενίκευση) της κλάσης οντοτήτων που δημιουργήθηκε. Οι υποκλάσεις μπορούν να έχουν και επιπλέον γνωρίσματα από αυτά των υπερκλάσεών τους. Μερικά συστήματα επιτρέπουν σε μία υποκλάση να έχει μονάχα μία υπερκλάση ενώ άλλα επιτρέπουν σε μία υποκλάση να έχει ταυτόχρονα πολλές υπερκλάσεις. Στην τελευταία περίπτωση η ιεραρχία κλάσεων και υποκλάσεων δημιουργεί ένα κατευθυνόμενο γράφο χωρίς κύκλους.

Η υλοποίηση ενός οντοκεντρικού συστήματος διαχείρισης βάσεων δεδομένων απαιτεί τη λήψη αποφάσεων σε διάφορους τομείς όπως την αρχιτεκτονική του συστήματος και τις

δομές αποθήκευσης για τις οντότητες, τον ορισμό και τις μεταβολές στο εννοιολογικό σχήμα της βάσης, τη διαμόρφωση ερωτήσεων στη βάση, την παροχή δυνατότητας ταυτόχρονης προσπέλασης στη βάση σε πολλούς χρήστες (concurrency control). Πολλές από τις τεχνικές που χρησιμοποιούνται σε συμβατικά συστήματα βάσεων δεδομένων στους προηγούμενους τομείς μπορούν να εφαρμοστούν απευθείας σε οντοκεντρικές βάσεις δεδομένων. Όμως τα χαρακτηριστικά του οντοκεντρικού μοντέλου αναπαράστασης δεδομένων μπορούν να οδηγήσουν σε σημαντικές αλλαγές στις συμβατικές τεχνικές ή και να απαιτήσουν τη δημιουργία εντελώς νέων τεχνικών. Μπορούμε να διακρίνουμε στα παρακάτω τον τρόπο με τον οποίο το οντοκεντρικό μοντέλο επηρεάζει την αρχιτεκτονική ενός συστήματος βάσεων δεδομένων που το υποστηρίζει:

- **Αναγνωριστικό οντότητας:** Στα οντοκεντρικά συστήματα κάθε οντότητα έχει ένα μοναδικό αναγνωριστικό που δημιουργείται από το ίδιο το σύστημα. Υπάρχουν διάφοροι τρόποι για τη δημιουργία αναγνωριστικών οντοτήτων, αλλά συνήθως χρησιμοποιούνται μονάχα δύο προσεγγίσεις. Σύμφωνα με την πρώτη, κάθε οντότητα έχει ως αναγνωριστικό ένα διάνυσμα με μορφή <*αναγνωριστικό κλάσης*, *αναγνωριστικό οντότητας*>. Το πεδίο *αναγνωριστικό κλάσης* είναι το αναγνωριστικό για την κλάση οντοτήτων όπου ανήκει η οντότητα, ενώ το πεδίο *αναγνωριστικό οντότητας* αποτελεί το αναγνωριστικό για την οντότητα μέσα στην κλάση οντοτήτων ή σε ολόκληρη τη βάση δεδομένων. Σύμφωνα με τη δεύτερη προσέγγιση χρησιμοποιείται μονάχα το *αναγνωριστικό οντότητας* για τη διάκριση μιας οντότητας. Ακόμα και στην περίπτωση αυτή το *αναγνωριστικό κλάσης* για την οντότητα πρέπει να είναι διαθέσιμο στην ίδια την οντότητα έτσι ώστε να είναι δυνατή η διάκριση της περιγραφής των γνωρισμάτων της οντότητας μέσω της κλάσης οντοτήτων στην οποία ανήκει. Ουσιαστικά το αναγνωριστικό οντότητας αποτελεί για τις οντοκεντρικές βάσεις δεδομένων το αντίστοιχο του κυρίως κλειδιού (primary key) των σχεσιακών βάσεων δεδομένων. Σημαντική διαφορά είναι ότι το αναγνωριστικό αυτό ορίζεται από το σύστημα και όχι από το σχεδιαστή της βάσης όπως γίνεται στις σχεσιακές βάσεις δεδομένων.

- **Ιεραρχία κλάσεων οντοτήτων - Ιεραρχία κατηγοριοποίησης:** Καθώς το οντοκεντρικό μοντέλο αναπαράστασης δεδομένων χρησιμοποιεί αρχές από σημασιολογικά μοντέλα αναπαράστασης δεδομένων όπως οι σχέσεις γενίκευσης και κατηγοριοποίησης, έχει μεγαλύτερες δυνατότητες αναπαράστασης από ότι το σχεσιακό μοντέλο. Το εννοιολογικό σχήμα της βάσης για το οντοκεντρικό μοντέλο αναπαράστασης δεδομένων έχει δύο διαστάσεις που δημιουργούνται από τις σχέσεις αυτές. Έτσι, κάθε οντότητα έχει μια θέση στην ιεραρχία κλάσεων και μια στην ιεραρχία κατηγοριοποίησης αντίστοιχα. Η δυναμική αυτή του οντοκεντρικού μοντέλου κάνει πιο πολύπλοκους τους

μηχανισμούς:

- μεταβολής του εννοιολογικού σχήματος της βάσης
- απάντησης ερωτήσεων και δομών αποθήκευσης.

Για παράδειγμα, όταν μια κλάση οντοτήτων διαγράφεται από τη βάση, οι υποκλάσεις της παύουν να έχουν όλα τα γνωρίσματα που κληρονόμησαν από την κλάση αυτή, ενώ οι περιπτώσεις (instances) αυτών των υποκλάσεων παύουν να έχουν τιμές γι'αυτά τα γνωρίσματα. Αντίστοιχα οι δομές αποθήκευσης των οντοτήτων μπορούν να μεταβάλλονται συνεχώς, καθώς η εισαγωγή ή διαγραφή υπερκλάσεων ή κλάσεων οντοτήτων γι'αυτές μπορεί να αυξήσει ή να μειώσει τα περιεχόμενα των δομών αποθήκευσης που πρέπει να έχουν τη δυνατότητα να μεταβάλλονται δυναμικά.

Τα οντοκεντρικά συστήματα διαχείρισης βάσεων δεδομένων που έχουν υλοποιηθεί μέχρι στιγμής αποτελούν επεκτάσεις σχεσιακών συστημάτων τα οποία υποστηρίζουν διαχείριση οντοτήτων, ή επεκτάσεις οντοκεντρικών γλωσσών προγραμματισμού όπου χρησιμοποιείται σταθερή μνήμη (stable memory) στην αποθήκευση των οντοτήτων. Σύμφωνα με το [**?**], τα οντοκεντρικά συστήματα βάσεων δεδομένων χωρίζονται σε τρείς κατηγορίες:

- **Οντοκεντρικά ως προς τη δομή:** το μοντέλο αναπαράστασης δεδομένων για τέτοια συστήματα επιτρέπει τον ορισμό πολύπλοκων οντοτήτων.
- **Οντοκεντρικά ως προς τη λειτουργία:** το μοντέλο αναπαράστασης δεδομένων στην περίπτωση αυτή επιτρέπει τη δημιουργία τελεστών οι οποίοι χειρίζονται τις οντότητες. Αυτά τα συστήματα είναι ταυτόχρονα οντοκεντρικά ως προς τη δομή καθώς οι τελεστές προϋποθέτουν την ύπαρξη οντοτήτων στις οποίες και εφαρμόζονται.
- **Οντοκεντρικά ως προς τη συμπεριφορά:** σε τέτοια συστήματα το μοντέλο αναπαράστασης δεδομένων παρέχει τη δυνατότητα στο χρήστη να ορίσει τύπους οντοτήτων και τελεστές που λειτουργούν σ' αυτούς τους τύπους. Στην περίπτωση αυτή η διαχείριση για τις περιπτώσεις (instances) των τύπων οντοτήτων γίνεται μέσω των τελεστών που έχουν ορισθεί για τις αντίστοιχες κλάσεις.

Στη συνέχεια του κεφαλαίου επιχειρούμε μια μικρή παρουσίαση σε διάφορα οντοκεντρικά συστήματα διαχείρισης βάσεων δεδομένων. Στην παρουσίαση αυτή εστιάζουμε την προσοχή μας στο μοντέλο αναπαράστασης δεδομένων και στους μηχανισμούς αποθήκευσης και διαχείρισης οντοτήτων. Έχουμε παραλείψει εντελώς τους μηχανισμούς ερωτήσεων στη βάση που δημιουργείται παραπέμποντας τον αναγνώστη στο [**?**].

2.2 Το σύστημα Iris

Το σύστημα διαχείρισης βάσεων δεδομένων Iris [**?**] [**?**] αποτελεί ένα ερευνητικό πρωτότυπο που υλοποιήθηκε στα εργαστήρια της Hewlett-Packard. Σκοπός του συστήματος είναι να βελτιώσει την παραγωγικότητα του προγραμματιστή βάσεων δεδομένων προσφέροντας υποστήριξη σε εφαρμογές όπως: πληροφοριακά συστήματα γραφείου, συστήματα βάσεων γνώσης και σχεδίαση λογισμικού και υλικού (hardware) μέσω υπολογιστή. Η αρχιτεκτονική του συστήματος αποτελείται από τρία τμήματα: την *επαφή χρήσης* (user interface), το *διαχειριστή οντοτήτων* (object manager) και το *διαχειριστή αποθήκευσης* (storage manager).

Ο διαχειριστής οντοτήτων υποστηρίζει τον ορισμό του εννοιολογικού σχήματος της βάσης, τη διαχείριση δεδομένων και την επεξεργασία ερωτήσεων. Το μοντέλο δεδομένων που χρησιμοποιείται βασίζεται σε τρεις δομές: *οντότητες*, *τύπους οντοτήτων* και *συναρτήσεις*. Οι οντότητες διακρίνονται σε δύο τύπους: βασικές οντότητες (όπως ορμαθοί χαρακτήρων ή ακέραιοι αριθμοί) και μη βασικές οντότητες (όλες οι άλλες οντότητες που δεν περιγράφονται ή δεν μπορούν να αποθηκευθούν άμεσα). Κάθε μη βασική οντότητα έχει ένα μοναδικό αναγνωριστικό μέσω του οποίου γίνεται κάθε πρόσβαση σε αυτή. Οι σχέσεις γενίκευσης και κατηγοριοποίησης υποστηρίζονται για τους τύπους οντοτήτων, ενώ οι συναρτήσεις που ενεργούν στις οντότητες ορίζονται στους τύπους οντοτήτων και κληρονομούνται από τις υποκλάσεις τους.

Ο διαχειριστής αποθήκευσης αναπτύχθηκε χρησιμοποιώντας ένα συμβατικό σχεσιακό μηχανισμό αποθήκευσης, το HP-SQL. Ο μηχανισμός HP-SQL παρέχει δυνατότητες ορισμού και διαχείρισης πινάκων (tables) για σχέσεις, αλλά δεν υποστηρίζει πράξεις μεταξύ πινάκων (όπως joins). Έτσι, στο σύστημα Iris δημιουργήθηκαν ειδικοί μηχανισμοί για την ανάθεση αναγνωριστικών στις οντότητες και τη συσχέτιση διανυσμάτων (tuples) μέσω γνωρισμάτων. Η απόδοση του συστήματος περιορίζεται από το διαχειριστή αποθήκευσης, καθώς η μορφή των δεδομένων στο διαχειριστή αποθήκευσης και το σύστημα αποθήκευσης που χρησιμοποιείται είναι διαφορετική. Έτσι, χρησιμοποιούνται διάφοροι μηχανισμοί κρυφής μνήμης για τα δεδομένα, ενώ απαιτείται μετάφραση των δεδομένων από τη μορφή που έχουν στο δίσκο σε αυτή που χρησιμοποιεί ο διαχειριστής οντοτήτων στο σύστημα Iris και αντίστροφα. Σε επόμενες εκδόσεις του συστήματος θα υπάρξει ένα μοντέλο για τις λειτουργίες του διαχειριστή αποθήκευσης έτσι ώστε το σύστημα Iris να μπορεί να λειτουργήσει με διάφορα συστήματα αποθήκευσης.

2.3 Το σύστημα POSTGRES

Το σύστημα βάσεων δεδομένων POSTGRES [**?**] [**?**] [**?**] αποτελεί ένα ερευνητικό πρωτότυπο που σχεδιάστηκε από την ομάδα που σχεδίασε τη σχεσιακή βάση δεδομένων INGRES και υλοποιήθηκε στο Πανεπιστήμιο της Καλιφόρνιας στο Berkeley. Καθοριστική για τη σχεδίαση του συστήματος ήταν η άποψη των σχεδιαστών του, ότι τα σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων πρέπει να υποστηρίζουν διαχείριση δεδομένων, οντοτήτων και γνώσης. Στις απαιτήσεις του συστήματος ακόμα περιέχονται η ανάγκη άμεσης επανάκτησης (instant recovery) ώστε όταν μια δοσοληψία αποτυγχάνει ή συμβαίνει κάποια βλάβη στο σύστημα να υπάρχει δυνατότητα ανάκλησης δεδομένων της βάσης στη μορφή που αυτά είχαν σε προηγούμενες χρονικές στιγμές.

Το μοντέλο αναπαράστασης δεδομένων στο σύστημα POSTGRES εμπεριέχει όλα τα χαρακτηριστικά του βασικού οντοκεντρικού μοντέλου αναπαράστασης δεδομένων, όμως το σύστημα θα μπορούσε εξίσου να χαρακτηρισθεί ως ‘‘σχεσιακό με επεκτάσεις’’ (extended relational) ή ‘‘σχεσιακό με φωλιασμένες σχέσεις’’ (nested relational). Το σύστημα αναπαράστασης δεδομένων παρέχει τη δυνατότητα ορισμού κλάσεων οντοτήτων, περιπτώσεων και μεθόδων για τις κλάσεις οντοτήτων, ενώ υποστηρίζει και κληρονόμηση ιδιοτήτων μέσω της σχέσης εξειδίκευσης. Οι κλάσεις οντοτήτων χωρίζονται σε τρεις κατηγορίες: *βασικές*, *εικονικές* και *εκδόσεις* [**?**]. Βασικές είναι οι κλάσεις οντοτήτων των οποίων οι περιπτώσεις αποθηκεύονται στη βάση δεδομένων. Οι εικονικές κλάσεις οντοτήτων χρησιμοποιούνται για τη δημιουργία όψεων (views) της βάσης δεδομένων και οι περιπτώσεις τους δημιουργούνται όποτε χρειάζονται, χωρίς να αποθηκεύονται στη βάση δεδομένων. Τέλος οι εκδόσεις (versions) κλάσεων οντοτήτων χρησιμοποιούνται για να αναπαραστήσουν τις διαφορετικές μορφές που μπορεί να πάρει μια κλάση οντοτήτων. Για την αποθήκευση των δεδομένων για μια έκδοση κάποιας οντότητας χρησιμοποιούνται εγγραφές οι οποίες περιέχουν μονάχα τη διαφορά της νέας έκδοσης με την περίπτωση από το οποίο προκύπτει η νέα έκδοση. Το σύστημα υποστηρίζει και ορισμό αφαιρετικών τύπων δεδομένων μέσω των οποίων μπορούν να κατασκευαστούν σύνθετες κλάσεις οντοτήτων και μέθοδοι.

Το σύστημα POSTGRES παρέχει στο σχεδιαστή εφαρμογής τη δυνατότητα να ορίζει δικούς του κανόνες που αφορούν τις κλάσεις οντοτήτων ή συγκεκριμένες περιπτώσεις κάποιας κλάσης οντοτήτων. Οι κανόνες αυτοί μπορεί να αφορούν στην εισαγωγή, διαγραφή ή αλλαγή δεδομένων στη βάση. Η εφαρμογή των κανόνων επιτυγχάνεται είτε μέσω ρουτινών που καλούνται από το σύστημα αποθήκευσης είτε μέσω του διερμηνευτή ερωτήσεων ο οποίος μεταβάλλει κάποιες ερωτήσεις ώστε να ικανοποιούνται οι κανόνες πριν οι ερωτήσεις εφαρμοστούν στη βάση.

Το σύστημα αποθήκευσης της POSTGRES [**?**] αποτελείται από τμήματα τα οποία ανα-

λαμβάνουν τη διαχείριση δοσοληψιών και διευκολύνουν την προσπέλαση στις οντότητες που βρίσκονται στη βάση δεδομένων. Η βάση δεδομένων αποθηκεύεται σε μαγνητικούς και οπτικούς δίσκους. Οι μαγνητικοί δίσκοι χρησιμοποιούνται για την αποθήκευση της τρέχουσας κατάστασης της βάσης, ενώ οι οπτικοί δίσκοι περιέχουν προηγούμενες καταστάσεις της βάσης.

Οι διάφορες κλάσεις οντοτήτων αποτελούν τις σχέσεις για ένα σχεσιακό σύστημα αποθήκευσης, ενώ οι περιπτώσεις τους αποτελούν εγγραφές στο δίσκο. Για κάθε σχέση χρησιμοποιείται ένα ξεχωριστό αρχείο στο σύστημα αρχείων του UNIX (ωστόσο σε επόμενες εκδόσεις του συστήματος ενδέχεται να αλλάξει ο τρόπος αποθήκευσης των σχέσεων). Κάθε εγγραφή-οντότητα έχει ένα μοναδικό, παραγόμενο από το σύστημα, αναγνωριστικό εγγραφής-οντότητας. Τα αναγνωριστικά για τις περιπτώσεις κάποιας κλάσης οντοτήτων χρησιμοποιούνται μονάχα αν ο σχεδιαστής της εφαρμογής το απαιτήσει για τη συγκεκριμένη κλάση οντοτήτων. Επίσης, κάθε εγγραφή για μια οντότητα έχει μερικά επιπλέον πεδία τα οποία περιγράφουν πότε δημιουργήθηκε ή διαγράφτηκε η οντότητα, ποια δοσοληψία τη δημιούργησε ή τη διέγραψε, καθώς επίσης και το αναγνωριστικό εγγραφής και ένα δείκτη σε πιθανή επόμενη εγγραφή που περιγράφει νέα έκδοση της συγκεκριμένης οντότητας. Κάθε αλλαγή σε μια οντότητα έχει ως αποτέλεσμα τη δημιουργία μιας επόμενης εγγραφής για την οντότητα. Η νέα εγγραφή αυτή τοποθετείται στο τέλος μιας συνδεδεμένης λίστας εγγραφών και περιέχει μονάχα την διαφορά της νέας κατάστασης (έκδοσης) της οντότητας από την κατάσταση στην οποία βρισκόταν η οντότητα πριν την αλλαγή. Η αρχική εγγραφή στη λίστα εγγραφών μιας οντότητας περιέχει τα πραγματικά δεδομένα για την οντότητα αυτή.

Ο μηχανισμός της λίστας εγγραφών για κάθε οντότητα διευκολύνει την προσπέλαση στην κατάσταση μιας οντότητας σε προηγούμενη χρονική στιγμή. Μια τέτοια προσπέλαση πραγματοποιείται μέσω σειριακής πρόσβασης στις εγγραφές της λίστας εγγραφών της οντότητας, μια και κάθε εγγραφή περιέχει, όπως αναφέραμε, πεδία που περιγράφουν το χρονικό διάστημα κατά το οποίο η οντότητα βρισκόταν στην κατάσταση που ‘‘περιγράφει’’ η εγγραφή. Όμως όλες οι εγγραφές, εκτός της αρχικής, αποτελούν ‘‘εγγραφές διαφορών’’ (delta records) και έτσι σε κάθε προσπέλαση σε τέτοια εγγραφή, πρέπει η πληροφορία αυτής της εγγραφής να ενσωματωθεί σε αυτήν της προηγούμενης, προκειμένου να ανακατασκευασθεί η κατάσταση της οντότητας για το χρονικό διάστημα που αυτή η εγγραφή περιγράφει. Η λειτουργία αυτή απασχολεί αρκετά τον επεξεργαστή, αλλά δεν πρόκειται να επηρεάζει αργότερα την απόδοση του συστήματος, καθώς δημιουργούνται υπολογιστές με μεγάλη υπολογιστική ισχύ. Η απόδοση του συστήματος χειροτερεύει αισθητά καθώς το μέγεθος της βάσης δεδομένων στο δίσκο αυξάνεται δραστικά λόγω των ‘‘εγγραφών διαφοράς’’ όταν συμβαίνουν πολλές αλλαγές στα δεδομένα της βάσης. Το σύστημα POSTGRES όμως δεν έχει

καθόλου καθυστέρηση λόγω δοσοληψιών που αποτυγχάνουν ή πιθανής βλάβης, μια και η βάση δεδομένων στο δίσκο βρίσκεται πάντα σε κατάσταση συνέπειας (consistency). Η μόνη περίπτωση μη σωστής λειτουργίας του συστήματος που χρειάζεται κάποια επεξεργασία, είναι όταν τα αποτελέσματα μιας επιτυχούς δοσοληψίας δεν έχουν γραφτεί πλήρως στη βάση. Η περίπτωση αυτή όμως μπορεί να αποφευχθεί με την ύπαρξη σταθερής μνήμης (stable memory) στον υπολογιστή.

Για κάθε χρήστη του συστήματος POSTGRES δημιουργείται μια ξεχωριστή διεργασία. Ταυτόχρονα υπάρχει μια μοναδική διεργασία, με το όνομα vacuum cleaner, η οποία τρέχει συνεχώς. Η διεργασία αυτή αναλαμβάνει την μεταφορά των λιστών ‘‘εγγραφών διαφοράς’’ από τα διάφορα αρχεία σε ένα σύστημα οπτικών δίσκων έτσι ώστε σχεδόν πάντα στα αρχεία εγγραφών να υπάρχουν μονάχα οι αρχικές εγγραφές για κάθε οντότητα, αντί για τις λίστες εγγραφών. Η διεργασία vacuum cleaner δημιουργεί μια νέα αρχική εγγραφή με την τρέχουσα κατάσταση για κάθε οντότητα για την οποία υπάρχει λίστα εγγραφών που δεν είναι κενή, και την τοποθετεί στη θέση της αρχικής εγγραφής. Ουσιαστικά, είναι η αστοχία αυτής της διεργασίας να δουλέψει αποδοτικά που προκαλεί την πτώση της απόδοσης του συστήματος όταν υπάρχει μεγάλη χρήση. Αξίζει πάντως να αναφέρουμε ότι η απόδοση του συστήματος POSTGRES σύμφωνα με συνήθη προγράμματα μετρήσεως επίδοσης (benchmarks) είναι η μισή της απόδοσης της εμπορικής έκδοσης του συστήματος INGRES [**?**].

2.4 Το σύστημα O_2

Το σύστημα O_2 [**?**] [**?**] αποτελεί ένα νέο εμπορικό σύστημα διαχείρισης βάσεων δεδομένων που προορίζεται για τη δημιουργία εφαρμογών για συστήματα σχεδίασης μέσω υπολογιστή (CAD), πληροφοριακά συστήματα, συστήματα αυτοματισμού γραφείου, αλλά και παραδοσιακά συστήματα βάσεων δεδομένων. Οι αντικειμενικοί στόχοι του συστήματος είναι οι παρακάτω:

(i) αύξηση της παραγωγικότητας κατά την ανάπτυξη εφαρμογών,

(ii) παροχή εργαλείων που εξυπηρετούν στην ανάπτυξη νέων εφαρμογών και

(iii) βελτίωση της ποιότητας των εφαρμογών που αναπτύσσονται, όσον αφορά την απόδοση, εμφάνιση αλλά και δυνατότητα εύκολης συντήρησης.

Η αρχιτεκτονική του συστήματος μπορεί να περιγραφεί ως αρχιτεκτονική πελάτη-σταθμού εξυπηρέτησης. Ο σταθμός εξυπηρέτησης είναι μια διεργασία, μέσω της οποίας οι διεργασίες πελάτες, για τους διάφορους χρήστες του συστήματος μπορούν να προσπελαύνουν και να

μεταβάλλουν τα περιεχόμενα της βάσης δεδομένων. Ο πυρήνας του συστήματος βάσεων δεδομένων ονομάζεται O_2 Engine [**?**] και αποτελείται από τρία επίπεδα:

(i) το επίπεδο διαχείρισης εννοιολογικού σχήματος (schema manager)

(ii) το επίπεδο διαχείρισης οντοτήτων (object manager)

(iii) το επίπεδο διαχείρισης αποθήκευσης που αποτελείται από μια επέκταση του WiSS (Wisconsin Storage System) [**?**].

Το επίπεδο διαχείρισης εννοιολογικού σχήματος αποτελεί μια ξεχωριστή εφαρμογή στο σύστημα O_2, η οποία χρησιμοποιεί την επαφή ζεύξης (interface) που παρέχει το επίπεδο διαχείρισης οντοτήτων. Το επίπεδο αυτό χρησιμοποιείται για τη δημιουργία, αλλαγή και διαγραφή κλάσεων οντοτήτων. Επίσης, αναλαμβάνει την εφαρμογή και έλεγχο των κανόνων συνέπειας (consistency) και κληρονόμησης ιδιοτήτων και δίνει τη δυνατότητα στο σχεδιαστή εννοιολογικού σχήματος να χρησιμοποιεί γνωρίσματα και κλάσεις οντοτήτων που δεν έχουν καθοριστεί πλήρως όταν δημιουργεί νέες κλάσεις οντοτήτων. Η συνέπεια του εννοιολογικού σχήματος της βάσης δεδομένων ελέγχεται μόνον όταν όλες οι κλάσεις οντοτήτων και τα γνωρίσματά τους έχουν καθοριστεί πλήρως. Για τον έλεγχο του εννοιολογικού σχήματος το επίπεδο αυτό χρησιμοποιεί άλλες εφαρμογές του συστήματος O_2 και έτσι το εννοιολογικό σχήμα μπορεί να καθορισθεί μέσω των κλάσεων και μεθόδων που παρέχουν οι τύποι δεδομένων που υποστηρίζει το O_2. Κατόπιν μεταφράζονται σε κλάσεις και μεθόδους σε κάποια από τις γλώσσες προγραμματισμού που υποστηρίζει το σύστημα (όπως C++). Όμως, μπορεί να ακολουθηθεί και η αντίστροφη πορεία, δηλαδή ο σχεδιαστής εννοιολογικού σχήματος μπορεί να χρησιμοποιήσει αμέσως δομές κάποιας γλώσσας προγραμματισμού, οι οποίες μεταφράζονται σε δομές του μοντέλου αναπαράστασης δεδομένων για το σύστημα O_2.

Το επίπεδο διαχείρισης οντοτήτων αποτελείται από διάφορα επιμέρους τμήματα, τα οποία πραγματοποιούν τις διάφορες λειτουργίες του επιπέδου αυτού. Μπορούμε να διακρίνουμε τα επιμέρους τμήματα *διαχείρισης σύνθετων οντοτήτων*, *διαχείρισης δοσοληψιών* και *διαχείρισης δεικτών*. Οι διεργασίες πελατών και η διεργασία εξυπηρέτησης χρησιμοποιούν διαφορετικές μορφές του επιπέδου διαχείρισης οντοτήτων, εφόσον η διεργασία εξυπηρέτησης είναι η μόνη διεργασία που μπορεί να προσπελάσει άμεσα τα δεδομένα του δίσκου, ενώ οι διεργασίες πελάτη λαμβάνουν τα δεδομένα από τη διεργασία αυτή μέσω μηχανισμού αποστολής μηνυμάτων (message passing).

Το υποσύστημα διαχείρισης σύνθετων οντοτήτων υποστηρίζει δημιουργία, διαγραφή, μεταβολή και ανάκληση οντοτήτων, αλλά και κλήση των διαφόρων μεθόδων των οντοτήτων. Μια οντότητα προσπελαύνεται μέσω δύο τύπων αναγνωριστικών οντοτήτων, ανάλογα με

το αν η συγκεκριμένη οντότητα βρίσκεται ήδη στη μνήμη ή στο δίσκο. Τα αναγνωριστικά οντοτήτων είναι είτε *μόνιμα*, είτε *προσωρινά*. Το μόνιμο αναγνωριστικό για μια οντότητα είναι η διεύθυνση της θέσης του δίσκου στην οποία είναι αποθηκευμένη η οντότητα. Στην περίπτωση που μια οντότητα βρίσκεται στη μνήμη, το αναγνωριστικό της οντότητας συντίθεται από τη θέση στην εικονική μνήμη που βρίσκεται η οντότητα και ένα επιπλέον πεδίο που περιγράφει αν η οντότητα βρίσκεται στη μνήμη της διεργασίας εξυπηρέτησης ή στη μνήμη της διεργασίας πελάτη. Φυσικά όλες οι διεργασίες διατηρούν ένα πίνακα μετάφρασης από μόνιμα σε προσωρινά αναγνωριστικά οντοτήτων για τις οντότητες που χρησιμοποιούν.

Το μοντέλο αναπαράστασης δεδομένων επιτρέπει δύο τύπους δεδομένων: *τιμές* και *οντότητες*. Οι *τιμές* έχουν αναγνωριστικά οντοτήτων και μπορούν να εμπεριέχουν *τιμές*. Οι *τιμές* μπορεί να οργανώνονται σε δομημένους τύπους (σύνολα, λίστες και πλειάδες) ή σε ατομικούς τύπους. Κάθε τιμή έχει ένα τύπο ο οποίος περιγράφει τη δομή της. Επιπλέον, οι τιμές μπορούν να έχουν ονόματα οριζόμενα από το χρήστη. Το επίπεδο διαχείρισης οντοτήτων δε διακρίνει ανάμεσα σε οντότητες και δομημένες τιμές και αναφέρεται σε αυτές μέσω αναγνωριστικών οντοτήτων, καθώς τα αναγνωριστικά οντοτήτων ουσιαστικά είναι δείκτες στη θέση στο δίσκο όπου αποθηκεύονται οι οντότητες και οι τιμές. Οι οντότητες και οι δομημένες τιμές αποθηκεύονται σε εγγραφές που δημιουργούνται από το σύστημα αποθήκευσης WiSS.

Η διεργασία εξυπηρέτησης χρησιμοποιεί δυο μηχανισμούς κρυφής μνήμης. Ο ένας μηχανισμός χρησιμοποιείται για τη διαχείριση των σελίδων δεδομένων που περιέχουν εγγραφές του συστήματος αποθήκευσης, ενώ ο άλλος για τη διαχείριση των οντοτήτων που βρίσκονται στη μνήμη. Οι διεργασίες πελάτη έχουν μονάχα το δεύτερο μηχανισμό κρυφής μνήμης. Η μονάδα μεταφοράς ανάμεσα στο δίσκο και τη μνήμη είναι η σελίδα δίσκου (disk page), οπότε πρόσβαση σε μία οντότητα έχει ως αποτέλεσμα και την προανάκληση (prefetching) όλων των οντοτήτων που βρίσκονται στην ίδια σελίδα δίσκου με αυτή.

Όπως έχει ήδη αναφερθεί, το επίπεδο διαχείρισης αποθήκευσης στο O_2 έχει υλοποιηθεί χρησιμοποιώντας το σύστημα WiSS. Η διεργασία εξυπηρέτησης αναλαμβάνει όλες τις κλήσεις του συστήματος αποθήκευσης. Έτσι, η διεργασία αυτή αναλαμβάνει τη διάθεση χώρου στο δίσκο, την αποθήκευση και ανάκληση σελίδων του δίσκου, τη διαχείριση δοσοληψιών και τις περιπτώσεις βλάβης (recovery).

Το σύστημα WiSS σχεδιάστηκε ως ένα ευέλικτο σύστημα αποθήκευσης για τη δημιουργία πειραματικών συστημάτων βάσεων δεδομένων. Καθώς το λειτουργικό σύστημα UNIX δεν έχει σχεδιαστεί για υψηλή απόδοση συστημάτων βάσεων δεδομένων, το WiSS χρησιμοποιεί τμήματα δίσκων (raw disk partitions) για τα αρχεία του, αντί για το σύστημα αρχείων που

παρέχει το UNIX.

Το σύστημα αποθήκευσης WiSS αποτελείται από τέσσερα επίπεδα [**?**]. Το επίπεδο διαχείρισης εγγραφής/ανάγνωσης (physical I/O layer), εκτός από την ανάθεση και διαχείριση χώρου στις συσκευές αποθήκευσης (δίσκους, ταινίες) αναλαμβάνει και τη μεταφορά δεδομένων από/προς τις συσκευές αυτές. Το επόμενο επίπεδο διαχείρισης ενταμιευτών (buffer management layer) χρησιμοποιεί το προηγούμενο επίπεδο για ανάγνωση και αποθήκευση σελίδων από ή στο δίσκο, διατηρεί ένα ενταμιευτή σελίδων και χρησιμοποιεί την πολιτική αντικατάστασης της παλαιότερα χρησιμοποιημένης σελίδας (LRU policy). Ουσιαστικά ο ενταμιευτής σελίδων του συστήματος WiSS αποτελεί την κρυφή μνήμη σελίδων για το επίπεδο διαχείρισης οντοτήτων στο σύστημα O_2. Σε κάθε χρήστη παραχωρείται ένα τμήμα του ενταμιευτή σελίδων και έτσι ο αλγόριθμος αντικατάστασης εφαρμόζεται στο κάθε τμήμα του ενταμιευτή σελίδων που χρησιμοποιεί ο κάθε χρήστης χωριστά αντί για το σύνολο των σελίδων στον ενταμιευτή σελίδων. Το επόμενο επίπεδο αποτελεί το επίπεδο διαχείρισης δομών αποθήκευσης. Το επίπεδο αυτό παραχωρεί μια εγγραφή για κάθε δομή που ορίζεται από το σύστημα διαχείρισης οντοτήτων στο O_2. Επίσης, το επίπεδο αυτό παρέχει τη δυνατότητα δημιουργίας μεγάλων δομών και δεικτών οργανωμένων σε δομές B-δέντρων. Οι δομές B-δέντρων δεικτών χρησιμοποιούνται από το υποσύστημα διαχείρισης δεικτών του O_2 για τη δημιουργία δεικτών στην ιεραρχία κλάσεων οντοτήτων και στην ιεραρχία κατηγοριοποίησης. Το τελευταίο επίπεδο στο WiSS αποτελεί τη διασύνδεση με τα υπόλοιπα τμήματα του συστήματος βάσης δεδομένων και παρέχει διάφορες μεθόδους πρόσβασης στα δεδομένα (όπως σειριακή ή δεικτοδοτημένη αναζήτηση και αναζήτηση μεγάλων δομών δεδομένων). Το σύστημα αποθήκευσης χρησιμοποιεί το πρωτόκολλο κλειδώματος δύο φάσεων (2 phase locking protocol) όπου βασική μονάδα κλειδώματος αποτελεί κάθε αρχείο.

2.5 Το σύστημα EXODUS

Όπως και το σύστημα O_2, το σύστημα EXODUS που δημιουργήθηκε στο Πανεπιστήμιο του Wisconsin προορίζεται για τη δημιουργία πολλών διαφορετικών εφαρμογών που χρησιμοποιούν βάσεις δεδομένων. Το σύστημα παρέχει αρκετά εργαλεία, τα οποία δημιουργούν τα επιμέρους τμήματα μιας εφαρμογής σύμφωνα με τις προδιαγραφές της εφαρμογής, αλλά οι λειτουργίες της βάσης δεδομένων και οι μέθοδοι προσπέλασης σ'αυτήν πρέπει να υλοποιηθούν από τον προγραμματιστή βάσης δεδομένων.

Στην παρούσα εργασία θα παρουσιάσουμε το σύστημα αποθήκευσης για τις οντότητες σε μια βάση που δημιουργείται από το σύστημα EXODUS όπως παρουσιάζεται στο [**?**]. Η ιδιαιτερότητα αυτού του συστήματος αποθήκευσης είναι ότι λειτουργεί με ελάχιστη

πληροφορία για το εννοιολογικό σχήμα της βάσης δεδομένων (minimal semantics).

Η βασική μονάδα αποθηκευμένης πληροφορίας στο σύστημα EXODUS είναι η *οντότητα αποθήκευσης*, ουσιαστικά ένα πλήθος από bytes χωρίς καθόλου δόμηση. Τη σημασιολογία για τις *οντότητες αποθήκευσης* καθορίζουν και αναγνωρίζουν μονάχα οι εφαρμογές που επικοινωνούν με το σύστημα αποθήκευσης. Το σύστημα αποθήκευσης παρέχει λειτουργίες αποθήκευσης και διαχείρισης οντοτήτων αποθήκευσης που είναι ανεξάρτητες από το μέγεθος των οντοτήτων αποθήκευσης. Έτσι, το μέγεθος των οντοτήτων αποθήκευσης μπορεί να μεταβάλλεται δυναμικά.

Όταν μια οντότητα αποθήκευσης αποκτήσει μέγεθος μεγαλύτερο από μια σελίδα το σύστημα αποθήκευσης τη μετατρέπει σε *μεγάλη οντότητα αποθήκευσης* (large storage object). Οι μεγάλες οντότητες αποθήκευσης αναπαριστάνονται στο δίσκο με τη μορφή B^+ δέντρων, όπου τα κλειδιά είναι η θέση (σε bytes) όπου βρίσκεται κάποια πληροφορία στη μεγάλη οντότητα και τα φύλλα περιέχουν τα δεδομένα. Η ρίζα του δέντρου και οι διάφοροι κόμβοι περιέχουν διανύσματα της μορφής <*μετρητής, αριθμός σελίδας*> για κάθε κάθε παιδί της ρίζας. Το πεδίο *μετρητής* περιέχει το μήκος σε bytes από την αρχή της μεγάλης οντότητας αποθήκευσης μέχρι το τέλος του υποδέντρου που έχει ρίζα το συγκεκριμένο διάνυσμα. Οι αλγόριθμοι που χρησιμοποιούνται για εισαγωγή, διαγραφή και επαναϊσοζύγιση (rebalancing) στα B^+ δέντρα έχουν ικανοποιητική απόδοση ακόμη και για πολύ μεγάλα δέντρα.

Το σύστημα EXODUS υποστηρίζει άλλη μια δομή αποθήκευσης που ομαδοποιεί οντότητες αποθήκευσης και καταλαμβάνει ολόκληρα αρχεία. Η δομή αυτή, που ονομάζεται *οντότητα αρχείου* (file object) οργανώνεται και λειτουργεί όπως οι *μεγάλες οντότητες αποθήκευσης* και παρέχει τη δυνατότητα σειριακής αναζήτησης για τις οντότητες αποθήκευσης που περιέχει.

Τα αναγνωριστικά οντοτήτων έχουν τη μορφή <*αριθμός σελίδας, θέση*> για όλους τους τύπους δομών αποθήκευσης. Στην περίπτωση απλών οντοτήτων αποθήκευσης το αναγνωριστικό οντότητας περιέχει τη διεύθυνση στο δίσκο στην οποία είναι αποθηκευμένα τα δεδομένα, ενώ για τις μεγάλες οντότητες αποθήκευσης και τις οντότητες αρχείων δείχνει σε μια δομή που ονομάζεται επικεφαλίδα μεγάλης οντότητας (large object header) η οποία αποτελεί τη ρίζα του B^+ δέντρου για τις μεγάλες δομές.

Το σύστημα διαχείρισης ενταμιευτών παραχωρεί (όπως στο WiSS) τμήματα ενταμιευτών σε κάθε χρήστη. Επιπλέον, παρέχει ειδικές λειτουργίες οι οποίες βελτιστοποιούν τη διαχείριση μεγάλων οντοτήτων (με μέγεθος πολλών σελίδων μνήμης). Η ταυτόχρονη προσπέλαση στη βάση για πολλούς χρήστες διασφαλίζεται μέσω του πρωτόκολλου κλειδώματος δύο φάσεων. Η αντιμετώπιση βλαβών του συστήματος, που μπορούν να οδηγήσουν σε απώλεια δεδομένων, γίνεται μέσω τεχνικών logging για τις απλές οντότητες, ενώ για τις μεγάλες οντότητες χρησιμοποιούνται τεχνικές shadowing.

2.6 Άλλα συστήματα

Στη συνέχεια του κεφαλαίου παρουσιάζουμε συνοπτικά τα χαρακτηριστικά μερικών άλλων ερευνητικών και εμπορικών οντοκεντρικών συστημάτων διαχείρισης βάσεων δεδομένων.

Το σύστημα GemStone [**?**] αποτελεί ένα εμπορικό σύστημα διαχείρισης βάσεων δεδομένων που αναπτύχθηκε στην εταιρεία Servio Logic Corporation. Το σύστημα αυτό προορίζεται για εφαρμογές σχεδίασης μέσω υπολογιστή (CAD), βάσεις γνώσεων και πληροφοριακά συστήματα γραφείου. Το σύστημα αυτό χρησιμοποιεί τις οντοκεντρικές αρχές της γλώσσας προγραμματισμού Smalltalk μέσω της γλώσσας OPAL που χρησιμοποιείται για τον ορισμό του εννοιολογικού μοντέλου της βάσης δεδομένων.

Το σύστημα αυτό αποτελείται από δυο διαφορετικές διεργασίες. Η διεργασία Gem χρησιμοποιείται για τη διαχείριση του εννοιολογικού μοντέλου, ενώ η διεργασία Stone αποτελεί το μηχανισμό διαχείρισης αποθηκευστικού χώρου στο σύστημα VMS. Τα αναγνωριστικά οντοτήτων αντιστοιχούν σε λογικές διευθύνσεις για τις διάφορες οντότητες. Έτσι είναι αποδοτική η μεταφορά των οντοτήτων σε διαφορετικές διευθύνσεις στο δίσκο ενώ οι επιμέρους οντότητες που αποτελούν μια σύνθετη οντότητα αποθηκεύονται σε διαφορετικές θέσεις από την σύνθετη οντότητα. Οι μεγάλες οντότητες αναπαριστώνται μέσω μιας δενδρικής δομής που επιτρέπει μεταφορά τμήματος μονάχα της οντότητας στη μνήμη κατά τη χρησιμοποίησή της από κάποια εφαρμογή.

Το σύστημα ObjectStore [**?**], [**?**] αποτελεί ένα εμπορικό σύστημα που χρησιμοποιείται κυρίως για εφαρμογές σχεδίασης μέσω υπολογιστή, και πληροφοριακά συστήματα γραφείου. Το σύστημα αποτελεί επέκταση της οντοκεντρικής γλώσσας προγραμματισμού C++ και επιτρέπει άμεση πρόσβαση στα δεδομένα στη μνήμη μετατρέποντάς τα σε κατάλληλες δομές της γλώσσας C++.

Το σύστημα Versant [**?**] είναι ένα εμπορικό σύστημα που χρησιμοποιείται σε εφαρμογές σχεδίασης μέσω υπολογιστή και συστήματα αυτοματισμού γραφείου. Το σύστημα αποτελείται από 3 διαφορετικά υποσυστήματα: το σύστημα διαχείρισης οντοτήτων (object manager), το σύστημα αποθήκευσης οντοτήτων (object server) και την επαφή ζεύξης (user interface). Το σύστημα αυτό, όπως και τα περισσότερα οντοκεντρικά συστήματα διαχείρισης βάσεων δεδομένων, ακολουθεί την αρχιτεκτονική πελάτη - σταθμού εξυπηρέτησης. Κάθε διεργασία κάποιου χρήστη προσπελαύνει τη βάση μέσω κλήσεων σε συναρτήσεις της διεργασίας εξυπηρέτησης. Οι οντότητες που χρησιμοποιούνται από το σύστημα αυτό αποτελούν οντότητες της γλώσσας προγραμματισμού C++.

Το σύστημα Ontos [**?**] είναι ένα εμπορικό σύστημα που αποτελεί την εξέλιξη του οντοκεντρικού συστήματος VBase. Το σύστημα αυτό αποτελεί επίσης επέκταση της γλώσσας

προγραμματισμού C++ και χρησιμοποιείται σε αντίστοιχες εφαρμογές με τα προηγούμενα συστήματα. Το σύστημα αυτό παρέχει πολλές διεργασίες εξυπηρέτησης που χρησιμοποιούνται από μια ή περισσότερες διεργασίες πελάτη.

Το σύστημα Orion [**?**] αποτελεί ένα ερευνητικό πρωτότυπο που χρησιμοποιείται σε εφαρμογές τεχνητής νοημοσύνης, σχεδίασης μέσω υπολογιστή και συστήματα αυτοματισμού γραφείου. Το σύστημα αυτό υποστηρίζει σύνθετες οντότητες, μεταβολή του εννοιολογικού σχήματος της βάσης δεδομένων και πολλαπλές εκδόσεις για τις οντότητες. Οι οντότητες αποθηκεύονται μέσω του σχεσιακού συστήματος WiSS και αποτελούνται από γνωρίσματα. Οι σύνθετες οντότητες ορίζονται ως οντότητες που αποτελούνται από σύνθετα γνωρίσματα, τα οποία με τη σειρά τους αποτελούνται από σύνθετα και απλά γνωρίσματα. Όταν ένα σύνθετο γνώρισμα δεν περιέχει άλλα σύνθετα γνωρίσματα αναπαριστάται με τη μορφή πλειάδας στο σχεσιακό μοντέλο. Οι πλειάδες περιέχουν ένα επιπλέον γνώρισμα που αποτελεί το αναγνωριστικό για την κάθε οντότητα που αναπαριστούν.

Το σύστημα Mneme [**?**] είναι ένα σύστημα που χρησιμοποιείται για την μόνιμη αποθήκευση οντοτήτων που ορίζονται σε κάποια οντοκεντρική γλώσσα προγραμματισμού (όπως Smalltalk, Trellis/Owl, C++ ή Ada). Το σύστημα αποτελείται από μια διεργασία εξυπηρέτησης που χρησιμοποιεί κάποιο μηχανισμό αποθήκευσης και πολλές διεργασίες πελάτες, οι οποίες προσπελαύνουν τις αποθηκευμένες οντότητες μέσω της διεργασίας εξυπηρέτησης. Οι οντότητες διακρίνονται μέσω δυο διαφορετικών τύπων αναγνωριστικών οντοτήτων. Έτσι, οι οντότητες που χρησιμοποιούνται από τις διεργασίες πελάτη έχουν προσωρινά αναγνωριστικά, ενώ η διεργασία εξυπηρέτησης χρησιμοποιεί μόνιμα αναγνωριστικά για να τις διακρίνει.

2.7 Συμπεράσματα

Τα συστήματα που παρουσιάστηκαν διακρίνονται σε δύο κατηγορίες: τα συστήματα που αποτελούν επέκταση σχεσιακών συστημάτων υποστηρίζοντας το οντοκεντρικό μοντέλο αναπαράστασης δεδομένων και τα συστήματα που επεκτείνουν οντοκεντρικές γλώσσες προγραμματισμού παρέχοντας μηχανισμούς μόνιμης αποθήκευσης για τις δομές των γλωσσών αυτών. Στην πρώτη κατηγορία ανήκουν τα συστήματα Iris, POSTGRES, EXODUS και Orion. Στη δεύτερη κατηγορία ανήκουν τα συστήματα O_2, ObjectStore, Versant, Ontos και Mneme.

Τα περισσότερα από τα συστήματα αυτά χρησιμοποιούν τυπικούς σχεσιακούς μηχανισμούς αποθήκευσης. Έτσι, η μεταφορά μιας οντότητας από τη μνήμη στο δίσκο και αντίστροφα απαιτεί τη μετατροπή της οντότητας στην κατάλληλη μορφή αναπαράστασης

για το δίσκο ή τη μνήμη αντίστοιχα. Τα περισσότερα από τα συστήματα όμως που αποτελούν επεκτάσεις σε γλώσσες προγραμματισμού παρέχουν άμεση πρόσβαση στις οντότητες που βρίσκονται ήδη στη μνήμη επειδή η αναπαράσταση των οντοτήτων συμπίπτει με δομές της γλώσσας προγραμματισμού. Έτσι, οι οντότητες προσπελαύνονται μέσω δεικτών (pointers) στη μνήμη.

Τα συστήματα αυτά χρησιμοποιούν διαφορετικούς τρόπους για την δημιουργία και χρήση των αναγνωριστικών οντοτήτων. Για παράδειγμα το σύστημα POSTGRES επιτρέπει στο χρήστη να επιλέξει αν χρησιμοποιούνται αναγνωριστικά για τη διάκριση των οντοτήτων. Τα υπόλοιπα συστήματα χρησιμοποιούν πάντα αναγνωριστικά, αλλά στα συστήματα που αποτελούν επεκτάσεις σχεσιακών συστημάτων επιτρέπουν τη διάκριση οντοτήτων μέσω των γνωρισμάτων που περιέχει κάθε πλειάδα (tuple).

Στα συστήματα που αποτελούν επεκτάσεις οντοκεντρικών γλωσσών προγραμματισμού τα αναγνωριστικά οντοτήτων διακρίνονται σε μόνιμα και προσωρινά. Τα προσωρινά αναγνωριστικά αφορούν τις οντότητες που βρίσκονται στη μνήμη. Έτσι, οποτεδήποτε μια οντότητα μεταφέρεται στη μνήμη πρέπει να αλλαχθούν όλες οι αναφορές της σε άλλες οντότητες που βρίσκονται ήδη στη μνήμη, ώστε η οντότητα αυτή να αναφέρεται στις άλλες οντότητες μέσω των προσωρινών τους αναγνωριστικών. Έτσι, τα συστήματα O_2, ObjectStore, Versant, Ontos και Mneme έχουν υψηλές επιδόσεις όταν οι οντότητες που χρησιμοποιούν βρίσκονται στη μνήμη. Όμως η μεταφορά στη μνήμη μιας οντότητας γίνεται αργότερα από ότι στα άλλα συστήματα γιατί περιλαμβάνει ενημέρωση των αναφορών σε άλλες οντότητες για την οντότητα που μεταφέρεται στη μνήμη.

Στα συστήματα O_2 και EXODUS τα μόνιμα αναγνωριστικά οντοτήτων είναι η φυσική διεύθυνση στο δίσκο όπου αποθηκεύεται η οντότητα. Έτσι, κατά την αλλαγή της φυσικής θέσης μιας οντότητας (όταν αλλάζει το μέγεθος της οντότητας) απαιτείται ενημέρωση για όλες τις οντότητες που την αναφέρουν. Αντίθετα στα συστήματα Mneme και GemStone όπου τα αναγνωριστικά είναι ανεξάρτητα από τη θέση των οντοτήτων η αλλαγή της θέσης όπου αποθηκεύεται μια οντότητα μπορεί να πραγματοποιηθεί άμεσα χωρίς να απαιτείται μεταβολή άλλων οντοτήτων. Στο σύστημα POSTGRES όταν μια οντότητα αλλάζει θέση στο δίσκο δεν αλλάζει το αναγνωριστικό της, αλλά χρησιμοποιείται ένας δείκτης στη νέα διεύθυνση για την οντότητα. Έτσι, είναι δυνατόν να απαιτούνται δυο προσπελάσεις στο δίσκο για τη μεταφορά μιας οντότητας στη μνήμη.

Τα συστήματα O_2, ObjectStore, Versant, Ontos, GemStone και Mneme παρέχουν περιορισμένες δυνατότητες αναπαράστασης σε σχέση με τα συστήματα POSTGRES, Iris και Orion επειδή το μοντέλο αναπαράστασης δεδομένων παρέχει ένα μονάχα επίπεδο ταξινόμησης.

Τα συστήματα POSTGRES και EXODUS επιτρέπουν στο χρήστη να επιλέξει τον τρόπο με

τον οποίο ομαδοποιούνται οι οντότητες στο δίσκο (clustering). Τα συστήματα O2, Mneme, Orion και Ontos οργανώνουν τις οντότητες στο δίσκο σε ομάδες που περιέχουν τις επιμέρους οντότητες για μια σύνθετη οντότητα.

Κεφάλαιο 3

Το Σύστημα Παράστασης Γνώσης TELOS

3.1 Εισαγωγή

Η TELOS είναι μια γλώσσα αναπαράστασης γνώσης που δημιουργήθηκε για να βοηθήσει τους μηχανικούς στην ανάπτυξη πληροφοριακών συστημάτων καθ' όλη τη διάρκεια του κύκλου ζωής του λογισμικού. Η γλώσσα αυτή χρησιμοποιείται για την αναπαράσταση γνώσης για το περιβάλλον στο οποίο λειτουργεί ένα σύστημα, το είδος της πληροφορίας που αποθηκεύεται στο σύστημα και τη σημασία της πληροφορίας αυτής για το σύστημα, καθώς και γνώσεις κατά τις φάσεις ανάλυσης απαιτήσεων, σχεδιασμού και υλοποίησης του συστήματος. Η αρχική υλοποίηση της TELOS όπως περιγράφεται στο [**?**], έχει δυνατότητες χρονικής λογικής (time reasoning), εισαγωγής και ελέγχου περιορισμών και παρέχει ισχυρούς μηχανισμούς εννοιολογικής σχεδίασης όπως μηχανισμοί κατηγοριοποίησης και γενίκευσης-εξειδίκευσης.

Στην τρέχουσα υλοποίηση της TELOS [**?**] που χρησιμοποιείται στο ερευνητικό πρόγραμμα Ithaca δεν υπάρχουν οι μηχανισμοί χρονικής λογικής, εισαγωγής και επαλήθευσης περιορισμών και εξαγωγής συμπερασμάτων. Ο μηχανισμός χρονικής λογικής δεν χρειάζεται στο πλαίσιο του Ithaca, ενώ οι άλλοι δύο μηχανισμοί θα χειροτέρευαν την απόδοση του συστήματος. Αντί των περιορισμών ακεραιότητας (integrity constraints) χρησιμοποιούνται κλήσεις συναρτήσεων (triggers) οι οποίες καλούνται κατά την εισαγωγή ή διαγραφή οντοτήτων σε κάποια κλάση οντοτήτων.

3.2 Μοντέλο αναπαράστασης δεδομένων

Το μοντέλο αναπαράστασης δεδομένων στη γλώσσα TELOS ακολουθεί τις βασικές αρχές του οντοκεντρικού μοντέλου αναπαράστασης δεδομένων που περιγράφτηκε στο προηγούμενο κεφάλαιο. Έτσι, κάθε αντικείμενο του πραγματικού κόσμου αποτελεί μια ξεχωριστή οντότητα (*Object*). Όλες οι οντότητες που μπορούν να υπάρξουν σε μια βάση δεδομένων ταξινομούνται στην κλάση του συστήματος *Object*.

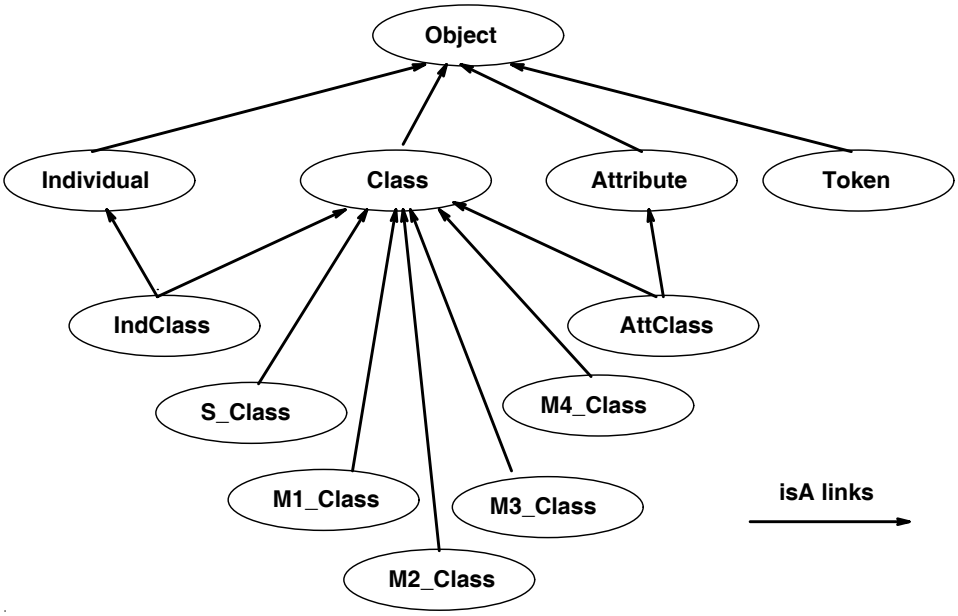
Η κλάση Object περιέχει τέσσερις υποκλάσεις: *Individual*, *Attribute*, *Class* και *Token*. Στη κλάση *Individual* ταξινομούνται οι οντότητες, οι κλάσεις οντοτήτων, οι κλάσεις από κλάσεις οντοτήτων κ.ο.κ. Η κλάση αυτή χρησιμοποιείται για την αναπαράσταση των διακριτών αντικειμένων του πραγματικού κόσμου. Η κλάση *Attribute* χρησιμοποιείται για την αναπαράσταση των γνωρισμάτων και των σχέσεων μεταξύ αντικειμένων του πραγματικού κόσμου. Στη κλάση αυτή ταξινομούνται οι σχέσεις, οι κλάσεις σχέσεων κ.ο.κ. Βέβαια τόσο οι οντότητες όσο και τα γνωρίσματα αποτελούν *οντότητες* (*Object*) στη γλώσσα TELOS και τυγχάνουν ομοιόμορφης διαχείρισης. Η κλάση *Class* χρησιμοποιείται για την ταξινόμηση όλων των κλάσεων οντοτήτων και κλάσεων γνωρισμάτων, καθώς και κλάσεων από κλάσεις. Οι ατομικές οντότητες και τα ατομικά γνωρίσματα που αποτελούν ουσιαστικά τον πληθυσμό της βάσης δεδομένων ταξινομούνται στη κλάση του συστήματος *Token*.

Για τις πρωτογενείς τιμές ακεραίων, πραγματικών αριθμών και ορμαθών χαρακτήρων χρησιμοποιούνται αντίστοιχα οι κλάσεις του συστήματος *Integer*, *Real* και *String*. Οι πρωτογενείς τιμές δεν αποτελούν ξεχωριστές οντότητες της βάσης και έτσι δεν μπορούν να δημιουργηθούν παρά μόνο να γίνει αναφορά σ'αυτές.

Οι κλάσεις που αναφέρθηκαν παραπάνω (εκτός από αυτές για πρωτογενείς τιμές) χρησιμοποιούνται για τη δημιουργία του εννοιολογικού σχήματος της βάσης δεδομένων. Στη γλώσσα TELOS το εννοιολογικό σχήμα αντιμετωπίζεται όπως και τα δεδομένα της βάσης και μπορεί να αλλάζει δυναμικά. Στο σχήμα **??** φαίνεται η ιεραρχία των κλάσεων συστήματος που χρησιμοποιούνται στη γλώσσα TELOS. Κάθε οντότητα μπορεί να αποτελεί περίπτωση περισσότερων από μιας από αυτές τις κλάσεις. Για παράδειγμα, μια κλάση οντοτήτων αποτελεί περίπτωση των κλάσεων *Individual* και *Class*, ενώ μια κλάση γνωρισμάτων αποτελεί περίπτωση των κλάσεων *Attribute* και *Class*.

3.2.1 Μηχανισμός ταξινόμησης

Στα οντοκεντρικά μοντέλα αναπαράστασης δεδομένων οι κλάσεις χρησιμοποιούνται για να ομαδοποιήσουν όλες τις οντότητες που έχουν κοινά χαρακτηριστικά. Έτσι, οι κλάσεις αποτελούν αφαιρετικούς τύπους δεδομένων και χρησιμοποιούνται για τη δημιουργία της

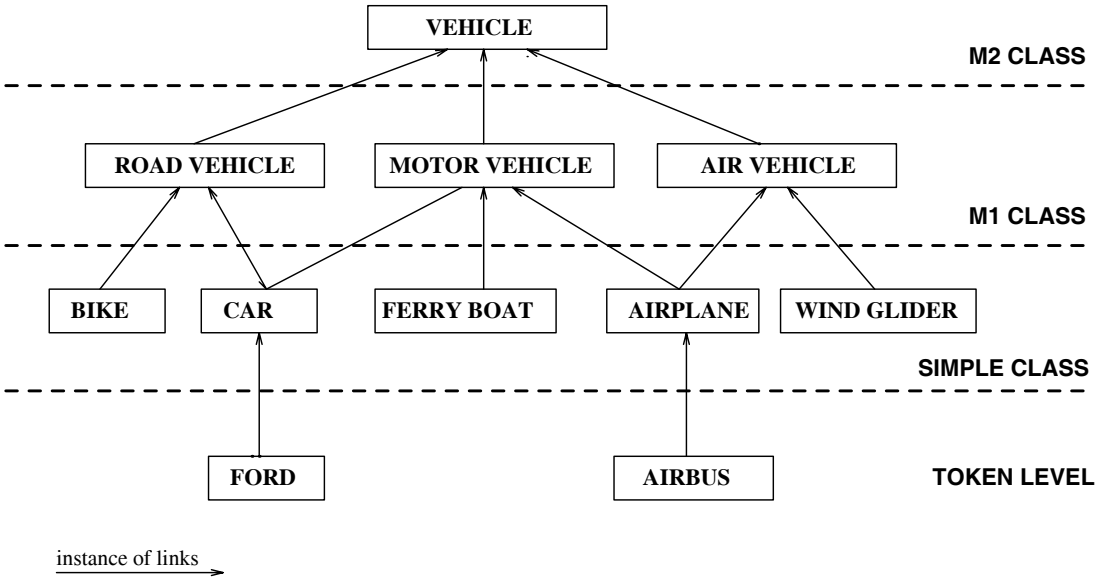


Σχήμα 3.1: Η ιεραρχία των κλάσεων του συστήματος

Όλες οι κλάσεις του συστήματος TELOS αποτελούν υποκλάσεις (εξειδικεύσεις) της κλάσης **Object**. Οι κλάσεις **Individual** και **Attribute** περιέχουν όλες τις ανεξάρτητες οντότητες και τις οντότητες που αποτελούν γνωρίσματα αντίστοιχα. Οι απλές οντότητες ανήκουν στη κλάση **Token** ενώ οι κλάσεις οντοτήτων μπορούν να βρίσκονται σε διάφορα επίπεδα ταξινόμησης που περιγράφονται από τις κλάσεις **S_Class**, **M1_Class**, **M2_Class**, **M3_Class** και **M4_Class**.

σχέσης ταξινόμησης.

Καθώς όμως κάθε κλάση οντοτήτων αποτελεί οντότητα στο οντοκεντρικό μοντέλο αναπαράστασης γνώσης, μπορεί με τη σειρά της να αποτελεί περίπτωση μιας άλλης γενικότερης κλάσης οντοτήτων. Δημιουργώντας νέες κλάσεις οντοτήτων που περιέχουν άλλες κλάσεις οντοτήτων, μπορεί να δημιουργηθεί μια μη φραγμένη ιεραρχία ταξινόμησης. Το πρώτο επίπεδο αυτής της ιεραρχίας αποτελούν οι ατομικές οντότητες (tokens), το επόμενο επίπεδο απλές κλάσεις οντοτήτων που ομαδοποιούν ατομικές οντότητες, το επόμενο επίπεδο οι μετακλάσεις που ομαδοποιούν κλάσεις οντοτήτων το επόμενο μετα-μετακλάσεις κ.ο.κ. Στη γλώσσα TELOS η ιεραρχία ταξινόμησης είναι φραγμένη και αποτελείται από τα εξής επίπεδα: *απλές οντότητες* (tokens), *απλές κλάσεις* (simple classes), *μετακλάσεις* (M1 classes) έως και M4 classes. Η επιλογή φραγμένης ιεραρχίας ταξινόμησης έγινε έχοντας υπόψη ότι τα επίπεδα αυτά αρκούν για την περιγραφή κάθε αντικειμένου στον πραγματικό κόσμο. Η φραγμένη ιεραρχία ταξινόμησης μαζί με άλλους μηχανισμούς αναπαράστασης που παρέχει η γλώσσα απλοποιεί τη διαχείριση του εννοιολογικού σχήματος. Οι κλάσεις που ορίζονται από το σύστημα TELOS χρησιμοποιούνται για τη ταξινόμηση ανεξάρτητων οντοτήτων (*Individuals*), αλλά και σχέσεων μεταξύ οντοτήτων ή και γνωρισμάτων (*Attributes*).



Σχήμα 3.2: Η ιεραρχία ταξινόμησης

Στο σχήμα αυτό φαίνονται τρία επίπεδα της ιεραρχίας ταξινόμησης. Οι οντότητες ενός επιπέδου κληρονομούν όλα τα χαρακτηριστικά των οντοτήτων στις οποίες ταξινομούνται στο αμέσως ανώτερο επίπεδο. Για παράδειγμα η κλάση οντοτήτων **CAR** αποτελεί περίπτωση των κλάσεων **ROAD VEHICLE** και **MOTOR VEHICLE** και κληρονομεί όλα τα χαρακτηριστικά τους.

Οι οντότητες που ορίζονται από κάποιο χρήστη ανήκουν σε ένα μοναδικό επίπεδο ταξινόμησης, αλλά μπορούν να αποτελούν περιπτώσεις περισσότερων από μιας κλάσεων οριζόμενων από το χρήστη. Ένα παράδειγμα φαίνεται στο σχήμα **??**. Η κλάση **CAR** κληρονομεί όλα τα χαρακτηριστικά από τις κλάσεις **ROAD VEHICLE** και **MOTOR VEHICLE** και βρίσκεται ένα επίπεδο ταξινόμησης πιο κάτω από αυτές. Ο μηχανισμός της ταξινόμησης χρησιμοποιείται για κληρονόμηση εγγενών ιδιοτήτων κλάσεων ενώ ο μηχανισμός απόδοσης γνωρίσματος που θα αναφέρουμε παρακάτω χρησιμοποιείται για την επίδοση ιδιαίτερων ιδιοτήτων.

3.2.2 Μηχανισμός γενίκευσης-εξειδίκευσης

Ο μηχανισμός της γενίκευσης στα εννοιολογικά μοντέλα χρησιμοποιείται για την ομαδοποίηση συνόλων οντοτήτων. Στην περίπτωση αυτή οι διαφορές ανάμεσα στα μέλη του συνόλου δεν λαμβάνονται υπόψη και δημιουργείται μια νέα γενικότερη κλάση που ομαδοποιεί τα μέλη αυτά σύμφωνα με τις κοινές τους ιδιότητες. Η νέα αυτή κλάση οντοτήτων αποτελεί τη *γενίκευση* όλων των μελών του συνόλου. Αντίστροφα, κάθε μέλος του συνόλου οντοτήτων αποτελεί *εξειδίκευση* της γενικής κλάσης οντοτήτων που ομαδοποιεί τα μέλη του συνόλου. Η σχέση *γενίκευσης* είναι η γνωστή *isA* σχέση. Στα οντοκεντρικά μοντέλα

αναπαράστασης δεδομένων η γενικότερη κλάση αναφέρεται ως *υπερκλάση* (superclass) των ειδικότερων κλάσεων που ομαδοποιεί, ενώ οι ειδικότερες κλάσεις ονομάζονται *υποκλάσεις* (subclasses) της γενικότερης κλάσης.

Η σχέση γενίκευσης μπορεί να εφαρμοστεί μονάχα σε κλάσεις οντοτήτων οι οποίες βρίσκονται στο ίδιο επίπεδο ταξινόμησης. Μια υποκλάση Β κάποιας κλάσης Α στην γλώσσα TELOS κληρονομεί όλα τα γνωρίσματα της κλάσης Α στα οποία μπορεί να περιορίζει τα πεδία τιμών και μπορεί να περιέχει επιπλέον δικά της γνωρίσματα. Επίσης, μια υποκλάση μπορεί να έχει περισσότερες από μία υπερκλάσεις κληρονομώντας έτσι γνωρίσματα από όλες τις υπερκλάσεις της σύμφωνα με την αρχή πολλαπλής κληρονόμησης. Δημιουργώντας νέες υπερκλάσεις για υπάρχουσες υπερκλάσεις ή νέες υποκλάσεις για υπάρχουσες υποκλάσεις, μπορεί να δημιουργηθεί μια *ιεραρχία κλάσεων* (ιεραρχία γενίκευσης) η οποία έχει τη μορφή κατευθυνόμενου γράφου. Η ιεραρχία γενίκευσης επιφέρει οικονομία στο εννοιολογικό σχήμα μιας βάσης δεδομένων αφού οι ορισμοί γνωρισμάτων για υποκλάσεις κληρονομούνται από τις υπερκλάσεις τους χωρίς να χρειάζεται να επαναληφθούν. Η ιεραρχία γενίκευσης-εξειδίκευσης είναι ορθογώνια προς την ιεραρχία ταξινόμησης, παρέχοντας έτσι ισχυρότερη εκφραστικότητα στη γλώσσα TELOS. Αυτό σημαίνει ότι οι δυο ιεραρχίες μπορεί να χρησιμοποιηθούν ανεξάρτητα η μια από την άλλη. Για παράδειγμα, μια ιεραρχία ειδίκευσης μπορεί να κληρονομείται σε διαφορετικά επίπεδα ταξινόμησης και ένα επίπεδο ταξινόμησης να περιέχει πολλές ιεραρχίες εξειδίκευσης.

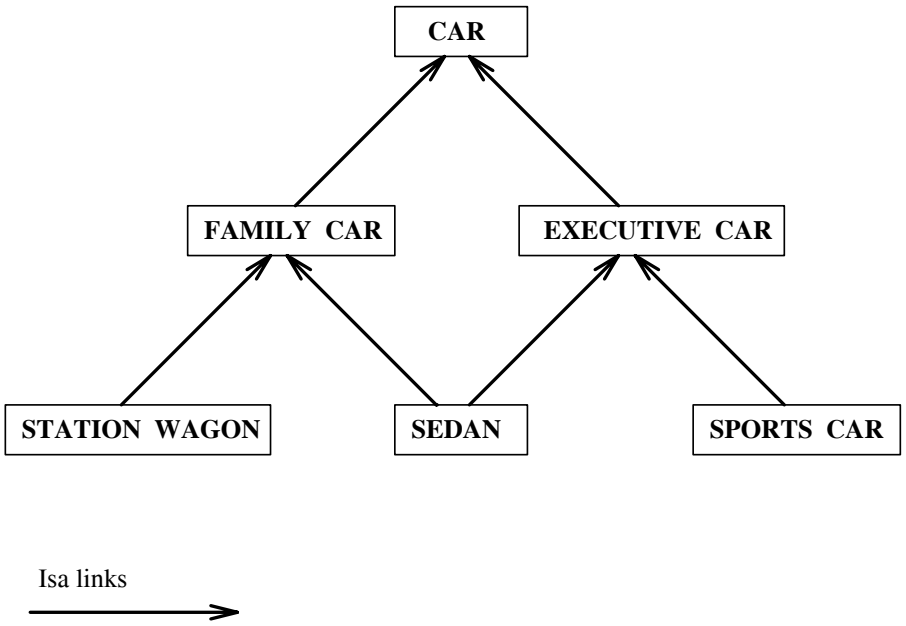
Στο σχήμα **??** παρουσιάζεται ένα παράδειγμα χρήσης του μηχανισμού γενίκευσης-εξειδίκευσης.

3.2.3 Μηχανισμός Απόδοσης Γνωρίσματος

Ο μηχανισμός απόδοσης γνωρίσματος στην TELOS χρησιμοποιείται για την υλοποίηση των μηχανισμών *σύνθεσης* (aggregation) και *σύστασης* (composition). Στην περίπτωση της σύνθεσης, μια οντότητα θεωρείται ως ένας αφαιρετικός τύπος δεδομένων ο οποίος ενσωματώνει ένα πλήθος από γνωρίσματα. Φυσικά τα γνωρίσματα αυτά αποτελούν επιμέρους οντότητες, τις οποίες όμως αντιλαμβανόμαστε ως σύνολο μέσω της οντότητας που τις περιέχει με το μηχανισμό της σύνθεσης.

Ο μηχανισμός της σύστασης είναι η γνωστή σχέση *part-of*. Στην περίπτωση αυτή, μια οντότητα σχηματίζεται από ένα σύνολο επιμέρους οντοτήτων, ακριβώς όπως και στην περίπτωση της σύνθεσης, αλλά η σημασία των επιμέρους οντοτήτων για την οντότητα που τις περιέχει είναι διαφορετική. Η οντότητα που τις περιέχει δεν θεωρείται αφαιρετικός τύπος, αλλά οι επιμέρους οντότητες αποτελούν ξεχωριστά τμήματα της οντότητας αυτής.

Τα γνωρίσματα στην TELOS θεωρούνται διμελείς σχέσεις μεταξύ οντοτήτων. Κάθε



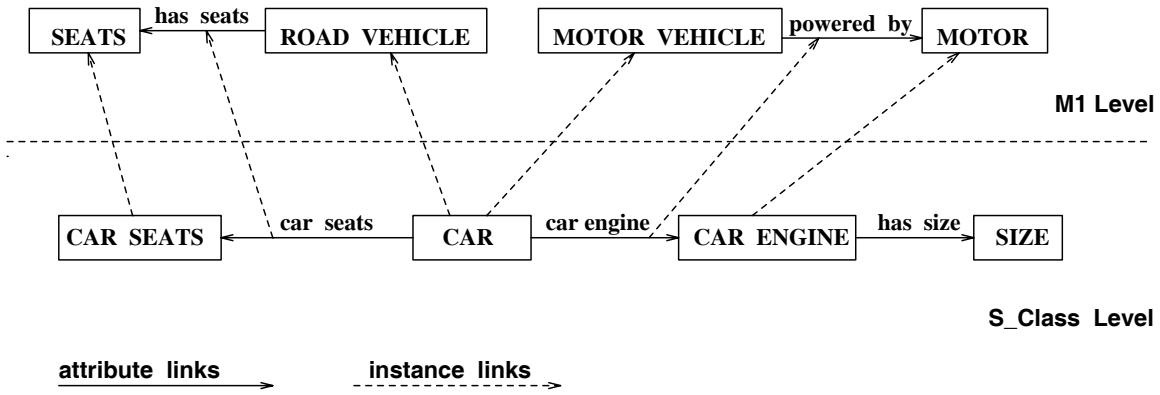
Σχήμα 3.3: Η ιεραρχία γενίκευσης

Η κλάση **SEDAN** αποτελεί άμεση εξειδίκευση των κλάσεων **FAMILY CAR** και **EXECUTIVE CAR** και έμμεση εξειδίκευση της κλάσης **CAR**. Η ιεραρχία αυτή βρίσκεται σε ένα επίπεδο ταξινόμησης.

γνώρισμα έχει ένα *πεδίο ορισμού* που είναι η οντότητα στην οποία αποδίδεται και ένα *πεδίο τιμών* που είναι η οντότητα την οποία προσδιορίζει. Επιπλέον, ο τύπος της οντότητας που προσδιορίζεται από το γνώρισμα αποτελεί και τον *τύπο τιμών* του γνωρίσματος. Σύμφωνα με το οντοκεντρικό μοντέλο αναπαράστασης δεδομένων, τα γνωρίσματα στην TELOS αποτελούν και αυτά οντότητες. Αντίθετα με το σχεσιακό μοντέλο, όπου τα πεδία τιμών των γνωρισμάτων αποτελούν οι πρωτογενείς τιμές (όπως ακέραιος, πραγματικός, ορμαθός χαρακτήρων), στα οντοκεντρικά μοντέλα τα πεδία τιμών των γνωρισμάτων μπορούν να είναι οποιαδήποτε κλάση οντοτήτων. Με τον τρόπο αυτό δημιουργούνται κλάσεις γνωρισμάτων, των οποίων οι περιπτώσεις βρίσκονται ένα επίπεδο πιο κάτω στην ιεραρχία ταξινόμησης. Έτσι, ο μηχανισμός ταξινόμησης όταν εφαρμόζεται στις κλάσεις και τα γνωρίσματά τους δημιουργεί ένα κατευθυνόμενο γράφο, που ονομάζεται ιεραρχία σύνθεσης-κλάσεων (class-composition hierarchy). Ένα παράδειγμα φαίνεται στο σχήμα **??**.

3.2.4 Ταυτότητα Οντοτήτων

Κάθε οντότητα διακρίνεται μέσω ενός παραγόμενου από το σύστημα TELOS *αναγνωριστικού οντότητας*. Το αναγνωριστικό αυτό χρησιμοποιείται αποκλειστικά από το σύστημα, ενώ οι χρήστες διακρίνουν τις οντότητες μέσω *λογικών ονομάτων*. Τα λογικά ονόματα δίνονται από τους χρήστες και είναι μοναδικά για κάθε ανεξάρτητη οντότητα. Τα λογικά ονόματα



Σχήμα 3.4: Ένα παράδειγμα της ιεραρχίας σύνθεσης

Η κλάση οντοτήτων **CAR** αποτελεί περίπτωση των κλάσεων **VEHICLE** και **MOTOR VEHICLE** και κληρονομεί από αυτές τις κλάσεις γνωρισμάτων **has seats** και **powered by** των οποίων περιπτώσεις είναι τα γνωρίσματα **car seats** και **car engine**. Το πεδία ορισμού και τιμών των γνωρισμάτων αυτών που βρίσκονται στο επίπεδο ταξινόμησης S_Class αποτελούν οι περιπτώσεις των πεδίων ορισμών και τιμών για τα γνωρίσματα **has seats** και **powered by** που βρίσκονται στο επίπεδο M1_Class.

για τα γνωρίσματα είναι μοναδικά μονάχα για την οντότητα μέσα στην οποία ορίζονται. Επίσης, ο χρήστης μπορεί να παραλείψει τον ορισμό λογικών ονομάτων για τα γνωρίσματα μιας οντότητας. Στην περίπτωση αυτή το σύστημα TELOS αναλαμβάνει τη δημιουργία ενός λογικού ονόματος για τα γνωρίσματα αυτά.

3.3 Υλοποίηση των Οντοτήτων για τη γλώσσα TELOS

Οι οντότητες στην TELOS ανάλογα με το επίπεδο ταξινόμησης στο οποίο βρίσκονται διακρίνονται σε *κλάσεις οντοτήτων* (classes) και *βασικές οντότητες* (tokens). Στο κατώτερο επίπεδο ταξινόμησης δεν επιτρέπεται η δημιουργία ιεραρχίας γενίκευσης και οι οντότητες του επιπέδου αυτού κληρονομούν τα χαρακτηριστικά της δομής τους από τις κλάσεις οντοτήτων σε S_Class επίπεδο, των οποίων αποτελούν περιπτώσεις. Καθώς τα *γνωρίσματα* (ή σχέσεις) αποτελούν και αυτά με τη σειρά τους οντότητες, διακρίνονται και αυτά σε *κλάσεις οντοτήτων* και *βασικές οντότητες*.

Κάθε οντότητα στην TELOS, έχει ένα μοναδικό αναγνωριστικό οντότητας, το οποίο και αποτελεί την ταυτότητα της οντότητας. Έτσι, είναι δυνατόν να διακρίνεται μοναδικά κάθε οντότητα μέσω αυτού του αναγνωριστικού. Καθώς το αναγνωριστικό αυτό παρέχεται από το σύστημα για κάθε οντότητα που δημιουργείται στο περιβάλλον TELOS, ονομάζεται SYSID (από τα αρχικά των λέξεων αναγνωριστικό συστήματος στην Αγγλική γλώσσα, system identifier) και αντιστοιχεί στα αναγνωριστικά OID (object identifier) των άλλων οντο-

κεντρικών συστημάτων. Καθώς το αναγνωριστικό αυτό είναι ουσιαστικά ένας φυσικός αριθμός, κάθε προσπέλαση σε οντότητες από το σύστημα γίνεται μέσω αυτού του αναγνωριστικού, αντί του λογικού ονόματος της οντότητας (που είναι ορμαθός χαρακτήρων) για λόγους βελτίωσης της απόδοσης του συστήματος. Επιπλέον, κάθε σχέση μεταξύ οντοτήτων μπορεί να περιγραφεί μέσω των αναγνωριστικών συστήματος για τις οντότητες και μόνο οι πρωτογενείς τιμές μπορούν να αναφέρονται άμεσα, χωρίς τη χρήση των αναγνωριστικών συστήματος.

3.3.1 Κλάσεις Οντοτήτων

Οι κλάσεις οντοτήτων στην TELOS μπορεί να είναι *κλάσεις γνωρισμάτων* ή *κλάσεις ανεξαρτήτων οντοτήτων* (Individuals). Το εννοιολογικό σχήμα μιας βάσης που έχει περιγραφεί σε TELOS έχει μορφή *σημασιολογικού δικτύου*, όπου οι κόμβοι του δικτύου αντιστοιχούν στις κλάσεις ανεξάρτητων οντοτήτων και τις απλές ανεξάρτητες οντότητες, ενώ οι σύνδεσμοι μεταξύ κόμβων αντιστοιχούν στις σχέσεις *ταξινόμησης* και *γενίκευσης-εξειδίκευσης* και στο μηχανισμό *σύνθεσης*, δηλαδή τις κλάσεις γνωρισμάτων και τα απλά γνωρίσματα. Στο υπόλοιπο μέρος αυτού του κεφαλαίου θεωρούμε ως *συνδέσμους* τα γνωρίσματα και τις κλάσεις γνωρισμάτων και ως *κόμβους* τις κλάσεις οντοτήτων και τις απλές οντότητες.

Στις κλάσεις οντοτήτων *κόμβων* και *συνδέσμων* μπορούν να χρησιμοποιηθούν όλοι οι μηχανισμοί δόμησης της TELOS, οδηγώντας έτσι σε πλήρεις περιγραφές του πραγματικού κόσμου. Κάθε εφαρμογή των μηχανισμών δόμησης περιλαμβάνει ακριβώς δυο οντότητες όπως φαίνεται στα προηγούμενα σχήματα του κεφαλαίου. Βέβαια οι σχέσεις που δημιουργούνται με τους μηχανισμούς δόμησης κληρονομούνται μέσω των ιεραρχιών ταξινόμησης, γενίκευσης και ιεραρχίας σύνθεσης κλάσεων από περισσότερες από μια οντότητες, αλλά οι άμεσοι σύνδεσμοι στους μηχανισμούς δόμησης περιλαμβάνουν πάντα, μονάχα δυο οντότητες (ή κλάσεις οντοτήτων). Έτσι, για την πλήρη περιγραφή μιας κλάσης οντοτήτων αρκεί να παρατεθούν στη δομή που την αναπαριστάνει μόνο τα γνωρίσματα ή οι οντότητες με τις οποίες συνδέεται άμεσα. Όλες οι υπόλοιπες συνδέσεις μπορούν να υπολογιστούν ως μεταβατικό κλειστό περίβλημα (closure) των συνδέσμων με αυτή την οντότητα για κάθε μηχανισμό δόμησης. Επίσης, χρειάζονται το *αναγνωριστικό συστήματος* για την οντότητα και η *περιγραφή του τύπου της*, δηλαδή το επίπεδο ταξινόμησης στο οποίο βρίσκεται και αν αποτελεί ανεξάρτητη οντότητα (individual) ή οντότητα γνώρισμα (attribute).

Κλάσεις κόμβων - class nodes

Στην κατηγορία αυτή ανήκουν οι κλάσεις ανεξάρτητων οντοτήτων (individual classes). Οι δομές αναπαράστασης και αποθήκευσης για τις οντότητες αυτές εκτός από το *αναγνωριστικό*

συστήματος και την *περιγραφή του τύπου* για κάθε τέτοια οντότητα περιλαμβάνει και ένα διάνυσμα από σύνολα αναγνωριστικών συστήματος που αναφέρονται σε όλες τις οντότητες με τις οποίες η οντότητα αυτή συνδέεται άμεσα μέσω των μηχανισμών δόμησης. Τα σύνολα αναγνωριστικών συστήματος διακρίνονται σε:

instance of: στο σύνολο αυτό περιλαμβάνονται όλες οι οντότητες των οποίων περίπτωση αποτελεί η συγκεκριμένη οντότητα,

instantiated: στο σύνολο αυτό περιλαμβάνονται όλες οι οντότητες που αποτελούν περίπτωση της συγκεκριμένης οντότητας,

isA: στο σύνολο αυτό περιλαμβάνονται όλες οι άμεσες υπερκλάσεις της συγκεκριμένης οντότητας,

subclasses: στο σύνολο αυτό περιλαμβάνονται όλες οι άμεσες υποκλάσεις της συγκεκριμένης οντότητας,

links: στο σύνολο αυτό περιλαμβάνονται όλοι οι σύνδεσμοι (γνωρίσματα) προς άλλες οντότητες που έχουν τη συγκεκριμένη οντότητα για πεδίο ορισμού,

linked by: στο σύνολο αυτό περιλαμβάνονται όλοι οι σύνδεσμοι για τους οποίους η τρέχουσα οντότητα αποτελεί πεδίο τιμών και

triggers: όλες οι οντότητες που υλοποιούν συναρτήσεις περιορισμών για τη συγκεκριμένη οντότητα.

Κλάσεις συνδέσμων - link classes

Στην κατηγορία αυτή ανήκουν οι κλάσεις γνωρισμάτων (attribute classes). Η δομή αναπαράστασης για τις οντότητες αυτές είναι αντίστοιχη με τη δομή αναπαράστασης για τις *κλάσεις κόμβων*, καθώς επιτρέπεται ορισμός γνωρισμάτων για τις κλάσεις γνωρισμάτων. Η δομή αναπαράστασης περιέχει δυο επιπλέον πεδία τα οποία απαραίτητα πρέπει να παίρνουν τιμές για να δημιουργηθεί η συγκεκριμένη οντότητα:

from-node: περιλαμβάνει το αναγνωριστικό συστήματος που περιγράφει την κλάση κόμβων που αποτελεί το πεδίο ορισμού για τη συγκεκριμένη κλάση συνδέσμων και

to-node: περιλαμβάνει το αναγνωριστικό συστήματος που περιγράφει την κλάση κόμβων που αποτελεί το πεδίο τιμών για τη συγκεκριμένη κλάση συνδέσμων.

3.3.2 Απλές οντότητες

Οι απλές οντότητες βρίσκονται στο κατώτερο επίπεδο ταξινόμησης (token level) και μπορεί να είναι απλές ανεξάρτητες οντότητες κόμβοι (individuals), ή απλές οντότητες σύνδεσμοι (attributes). Όντας στο κατώτερο επίπεδο ταξινόμησης, οι οντότητες αυτές δεν μπορεί να έχουν περιπτώσεις (instances). Επίσης, στο επίπεδο αυτό δεν επιτρέπεται η χρήση του μηχανισμού γενίκευσης-εξειδίκευσης. Έτσι, οι δομές αναπαράστασης για τις οντότητες αυτές δεν περιέχουν τα σύνολα αναγνωριστικών συστήματος *instantiated, isA* και *subclasses*, που έχουν νόημα μόνο για τις κλάσεις οντοτήτων. Επίσης, δεν περιλαμβάνουν το σύνολο αναγνωριστικών συστήματος trigger καθώς οι πιθανοί περιορισμοί για τις οντότητες αυτές ελέγχονται κατά τη δημιουργία ή διαγραφή τους μέσω της κλάσης οντοτήτων της οποίας αποτελούν περιπτώσεις.

Απλοί κόμβοι - token nodes

Στην κατηγορία αυτή ανήκουν οι απλές οντότητες (individual tokens). Η δομή αναπαράστασης για τις οντότητες αυτές είναι η απλούστερη δυνατή, μια και τέτοιες οντότητες συνδέονται άμεσα μονάχα με γνωρίσματα και τις κλάσεις οντοτήτων των οποίων αποτελούν περιπτώσεις.

Απλοί σύνδεσμοι - link tokens

Τα απλά γνωρίσματα (attribute tokens) που αποτελούν αυτή την κατηγορία οντοτήτων (link tokens), είναι γνωρίσματα των οποίων το πεδίο τιμών αποτελούν οι απλές ανεξάρτητες οντότητες και το πεδίο ορισμού οι απλές οντότητες (κόμβοι ή σύνδεσμοι).

3.3.3 Εγγενείς κλάσεις οντοτήτων για το σύστημα TELOS

Εκτός από τους προηγούμενους τύπους οντοτήτων, που χρησιμοποιούνται για την αναπαράσταση και αποθήκευση όλων των δεδομένων που ορίζει ο χρήστης στη γλώσσα TELOS, υπάρχει και ένα σύνολο από τύπους οντοτήτων που ορίζονται από το σύστημα και χρησιμοποιούνται για να περιγράφουν τη δομή κάθε βάσης που μπορεί να περιγραφεί σε TELOS. Οι κλάσεις αυτές αποτελούν τον αρχικό πληθυσμό κάθε βάσης δεδομένων στην TELOS και ομαδοποιούν όλες τις οντότητες που μπορεί να περιγράψει ένας χρήστης. Οι βασικότερες τέτοιες κλάσεις είναι η κλάση Object η οποία ομαδοποιεί όλες τις οντότητες που υπάρχουν σε μια βάση. Οι κλάσεις Individual και Attribute όντας υποκλάσεις της κλάσης Object ομαδοποιούν όλους τους *κόμβους* και τους *συνδέσμους* που ορίζει ένας χρήστης. Επίσης, υπάρχουν οι κλάσεις Class και Token που ομαδοποιούν όλες τις κλάσεις οντοτήτων και τις

απλές οντότητες που ορίζει ο χρήστης αντίστοιχα. Όλες αυτές οι κλάσεις του συστήματος TELOS δε βρίσκονται σε κάποιο συγκεκριμένο επίπεδο ταξινόμησης, μιας και οι περιπτώσεις τους μπορεί να βρίσκονται σε οποιοδήποτε επίπεδο ταξινόμησης.

Η δομή που τις αναπαριστάνει, δεν περιέχει σύνολα από αναγνωριστικά συστήματος για τις οντότητες που ορίζονται από το χρήστη και συνδέονται με αυτές. Έτσι, οι δομές αναπαράστασης γι' αυτές τις κλάσεις χρησιμοποιούνται προκειμένου να βρεθούν οι δυνατές συσχετίσεις μιας οποιασδήποτε οντότητας με κλάσεις του συστήματος. Η επιλογή να μη συνδέονται αυτές οι κλάσεις με οντότητες που ορίζει ο χρήστης, έγινε με σκοπό να μην προβάλλουν οι κλάσεις αυτές περιορισμούς στην απόδοση του συστήματος καθώς περιπτώσεις τους αποτελούν όλες οι οντότητες που ορίζει κάποιος χρήστης.

Ειδικές κλάσεις του συστήματος έχουν δημιουργηθεί για να αναπαραστήσουν τα δυνατά επίπεδα ιεραρχίας ταξινόμησης που μπορεί να περιλαμβάνει η σημασιολογική περιγραφή μιας βάσης σε TELOS. Οι κλάσεις αυτές είναι οι Token_Class, S_Class, M1_Class,..., M4_Class, μια και έξι επίπεδα ιεραρχίας ταξινόμησης είναι πρακτικά αρκετά για οποιαδήποτε σημασιολογική περιγραφή του πραγματικού κόσμου.

Τέλος για τις πρωτογενείς τιμές έχουν δημιουργηθεί οι κλάσεις Integer, Real και String. Κάθε πρωτογενής τιμή ταξινομείται σε μια από αυτές τις κλάσεις που βρίσκονται σε επίπεδο ταξινόμησης S_Class.

3.3.4 Μεγάλες οντότητες

Στην υλοποίηση που περιγράψαμε μέχρι στιγμής όλοι οι τύποι για την αναπαράσταση δεδομένων (class node, token node, link class και link token) αποτελούν κλάσεις υλοποιημένες στη γλώσσα προγραμματισμού C++. Το μέγεθος των συνόλων αναγνωριστικών συστήματος που χρησιμοποιούνται από αυτές τις κλάσεις είναι συγκεκριμένο[1] και έχει επιλεχθεί προκειμένου να είναι το πιο κατάλληλο για τις περισσότερες από τις οντότητες που ορίζονται από το χρήστη για λόγους οικονομίας χώρου και επιδόσεων του συστήματος.

Είναι φυσικό όμως κάποιες από τις οντότητες που ορίζει ένας χρήστης να έχουν περισσότερες άμεσες συνδέσεις με άλλες οντότητες απ' όσες μπορεί να καταγραφούν στα συγκεκριμένου μεγέθους σύνολα αναγνωριστικών συστήματος που χρησιμοποιούνται για κάθε οντότητα. Αν θεωρήσουμε ένα μοντέλο σε TELOS που περιγράφει λογισμικό τότε όλες οι συναρτήσεις κάποιου προγράμματος αποτελούν περιπτώσεις της κλάσης οντοτήτων που περιγράφει τις συναρτήσεις. Αν τώρα θεωρήσουμε ένα πολύ μεγάλο πρόγραμμα, είναι προφανές ότι το πλήθος των περιπτώσεων για την κλάση οντοτήτων που περιγράφει τις συναρτήσεις θα είναι ίσο με το πλήθος των συναρτήσεων του προγράμματος.

[1] Τα σύνολα αυτά υλοποιούνται με τη μορφή μονοδιάστατων πινάκων σταθερού μεγέθους.

Για το λόγο αυτό έχει δημιουργηθεί μια δομή αναπαράστασης συνόλων αναγνωριστικών συστήματος που ονομάζεται telos_oext. Η δομή αυτή χρησιμοποιείται για να δημιουργούμε πιθανές επεκτάσεις στα σύνολα αναγνωριστικών συστήματος για κάθε δομή αναπαράστασης οντοτήτων όταν τα σύνολα αυτά υπερχειλίζουν. Η δομή αυτή οργανώνεται σε απλά συνδεδεμένες λίστες με κόμβους που περιέχουν σταθερό πλήθος αναγνωριστικών συστήματος. Κάθε περίπτωση της δομής telos_oext, αποτελώντας κόμβο της συνδεδεμένης λίστας, εκτός από ένα σταθερό πίνακα αναγνωριστικών συστήματος, περιέχει ένα δείκτη στον επόμενο κόμβο της λίστας από επεκτάσεις για τη συγκεκριμένη οντότητα, καθώς και το αναγνωριστικό συστήματος για την οντότητα για λόγους ελέγχου της συνέπειας των δεδομένων στη βάση.

Οι δομές για τις οντότητες και τις επεκτάσεις τους υλοποιήθηκαν ως C++ κλάσεις από τον δρ. Martin Dörr και περιγράφονται στο [**?**].

3.3.5 Αναπαράσταση σχέσεων μεταξύ οντοτήτων

Όπως αναφέραμε, κάθε εφαρμογή μηχανισμών δόμησης στην TELOS περιλαμβάνει ακριβώς δύο οντότητες που συνδέονται με ένα σύνδεσμο στο σημασιολογικό δίκτυο που αντιστοιχεί στο μηχανισμό δόμησης που εφαρμόζεται. Στην περίπτωση *γνωρισμάτων*, ένας σύνδεσμος συνδέει την οντότητα με το γνώρισμα για το οποίο αποτελεί πεδίο ορισμού, ενώ ένας άλλος σύνδεσμος συνδέει το γνώρισμα με την οντότητα που αποτελεί πεδίο τιμών του γνωρίσματος.

Σε κάθε περίπτωση που δημιουργείται ένας σύνδεσμος στο σημασιολογικό δίκτυο, ο σύνδεσμος αυτός μπορεί να διασχιστεί και κατά τις δυο πιθανές του κατευθύνσεις. Η χρησιμοποίηση του μηχανισμού αυτού στην TELOS, κάνει πιο πιο αποδοτική τη διάσχιση συνδέσμων, αφού κάθε κόμβος στο σημασιολογικό δίκτυο επιτρέπει τη διάσχιση όλων των συνδέσμων που ξεκινούν ή φτάνουν σ' αυτόν.

Για την υλοποίηση του μηχανισμού αυτού, κάθε σύνδεσμος θεωρείται ότι έχει δυο κατευθύνσεις οι οποίες κρατούνται στην αναπαράσταση κάθε οντότητας. Θεωρώντας για παράδειγμα το σύνδεσμο isA μεταξύ των κλάσεων οντοτήτων FAMILY CAR και STATION WAGON (όπως παρουσιάζονται στο σχήμα **??**), η κλάση οντοτήτων FAMILY CAR είναι άμεση υπερκλάση της κλάσης οντοτήτων STATION WAGON, και το σύνολο των αναγνωριστικών συστήματος για τις υπερκλάσεις της κλάσης STATION WAGON περιλαμβάνει το αναγνωριστικό συστήματος για την κλάση FAMILY CAR. Αντίστροφα, το σύνολο αναγνωριστικών συστήματος για τις υποκλάσεις της κλάσης FAMILY CAR περιλαμβάνει το αναγνωριστικό συστήματος για την κλάση STATION WAGON.

Η επιλογή χρησιμοποίησης συνδέσμων δυο κατευθύνσεων αυξάνει το μέγεθος της βά-

σης δεδομένων, αλλά επειδή οι ερωτήσεις διάσχισης και προς τις δυο κατευθύνσεις των συνδέσμων είναι αρκετά συχνές προτιμάται η σπατάλη χώρου προκειμένου να βελτιωθεί η απόδοση του συστήματος.

Κεφάλαιο 4

Μηχανισμός διαχείρισης οντοτήτων

4.1 Εισαγωγή

Τα σχεσιακά συστήματα βάσεων δεδομένων χρησιμοποιούν μια μικρή βάση δεδομένων, η οποία περιέχει πληροφορίες για την εννοιολογική σχεδίαση κάθε βάσης δεδομένων που έχουν αποθηκεύσει. Η μικρή βάση δεδομένων περιέχει αντικείμενα τα οποία εξετάζει το σύστημα πριν από κάθε πιθανή προσπέλαση στα δεδομένα της βάσης [**?**].

Συνήθως η μικρή βάση δεδομένων αναφέρεται με το όνομα *κατάλογος συστήματος*. Ο *κατάλογος συστήματος* περιέχει πληροφορίες για τη δομή των διαφόρων σχέσεων που χρησιμοποιούνται και το χώρο που αυτές αποθηκεύονται, τα ονόματα των πιθανών χρηστών της βάσης ή τους δυνατούς τρόπους προσπέλασης στη βάση για κάθε χρήστη. Ο *κατάλογος συστήματος* αποτελείται ουσιαστικά από σχέσεις που περιλαμβάνουν κάθε λογής πληροφορία που είναι απαραίτητη στο σύστημα για τη σωστή λειτουργία του.

Τα περισσότερα οντοκεντρικά συστήματα διαχείρισης βάσεων δεδομένων χρησιμοποιούν σχεσιακά συστήματα αποθήκευσης στον δίσκο για τις διάφορες οντότητες που αποτελούν τη βάση δεδομένων. Η περιγραφή των τύπων οντοτήτων αποθηκεύεται στον κατάλογο συστήματος, ενώ οι οντότητες καθεαυτές αποτελούν πλειάδες που αποθηκεύονται σε κατάλληλα αρχεία.

Στο σύστημα βάσης δεδομένων για τη γλώσσα TELOS οι τύποι οντοτήτων και οι επιμέρους οντότητες θεωρούνται δεδομένα και η διαχείρισή τους είναι ομοιόμορφη. Ο κατάλογος συστήματος λοιπόν δεν περιέχει περιγραφές των τύπων οντοτήτων της βάσης δεδομένων, παρά μονάχα πληροφορίες για την αποθήκευση και την κατάσταση όλων των οντοτήτων της βάσης. Ο κατάλογος συστήματος μαζί με το μηχανισμό αποθήκευσης οντοτήτων που παρουσιάζονται στο κεφάλαιο αυτό αποτελούν το *μηχανισμό διαχείρισης οντοτήτων* για το σύστημα διαχείρισης βάσεων δεδομένων για τη γλώσσα TELOS.

Στα οντοκεντρικά συστήματα διαχείρισης βάσεων δεδομένων, ο *κατάλογος συστήματος* έχει ίδια δομή με το κατάλογο συστήματος σε σχεσιακά συστήματα βάσεων δεδομένων, περιλαμβάνοντας πληροφορίες για τη δομή των οντοτήτων και κάποιο υποκατάλογο που αντιστοιχεί τα αναγνωριστικά οντοτήτων στις διευθύνσεις στο δίσκο ή στη μνήμη όπου αποθηκεύονται οι αντίστοιχες οντότητες. Μερικά συστήματα αποφεύγουν τη χρήση του υποκαταλόγου μετάφρασης αναγνωριστικών οντοτήτων σε διευθύνσεις της μνήμης ή του δίσκου μεταφράζοντας άμεσα τα αναγνωριστικά οντοτήτων στις κατάλληλες διευθύνσεις.

4.2 Ο κατάλογος συστήματος

Η γλώσσα TELOS περιλαμβάνει ποικίλους μηχανισμούς αναπαράστασης, όπως αναφέραμε στο προηγούμενο κεφάλαιο, και χρησιμοποιείται για την αναπαράσταση σύνθετων εφαρμογών. Το εννοιολογικό σχήμα για τις εφαρμογές αυτές είναι συνήθως πολύπλοκο, και περιλαμβάνει πολλές κλάσεις οντοτήτων που σχετίζονται μεταξύ τους σε διάφορα επίπεδα ταξινόμησης.

Ο *κατάλογος συστήματος* για την TELOS δε χρειάζεται να περιέχει πληροφορία για το εννοιολογικό σχήμα της βάσης, αφού η πληροφορία αυτή ενσωματώνεται στις δομές αναπαράστασης για την κάθε οντότητα. Καθώς οι οντότητες για τις εγγενείς κλάσεις του συστήματος αποτελούν επιμέρους οντότητες στην TELOS, ενσωματώνουν και αυτές με τη σειρά τους κάθε δομική και εννοιολογική πληροφορία που τις αφορά.

Ο κατάλογος συστήματος όπως περιγράφεται στη συνέχεια του κεφαλαίου αποτελεί το μοναδικό μέσο για τη προσπέλαση των οντοτήτων από τις διάφορες εφαρμογές. Οι εφαρμογές αυτές καθορίζουν τις οντότητες που προσπελαύνουν μέσω των αναγνωριστικών οντοτήτων.

4.2.1 Αναγνωριστικά συστήματος

Τα *αναγνωριστικά οντοτήτων* (OID) που χρησιμοποιούνται σε κάθε οντοκεντρικό σύστημα διαχείρισης βάσεων δεδομένων, στο σύστημα TELOS ονομάζονται *αναγνωριστικά συστήματος* (system identifiers). Η επιλογή του ονόματος οφείλεται στο γεγονός ότι στο σύστημα TELOS τα *αναγνωριστικά οντοτήτων* χρησιμοποιούνται μονάχα από τις εφαρμογές της TELOS για την προσπέλαση των διαφόρων οντοτήτων. Οι χρήστες του συστήματος χρησιμοποιούν τα *λογικά ονόματα* των οντοτήτων για κάθε αναφορά που πραγματοποιούν σ'αυτές.

Στα συστήματα που παρουσιάστηκαν στο κεφάλαιο 2, το *αναγνωριστικό οντότητας* αποτελείται από ένα διάνυσμα της μορφής $< \textit{σελίδα}, \textit{θέση} >$, όπου το πεδίο *σελίδα* περιγράφει τη σελίδα δίσκου (disk page) στην οποία βρίσκεται η συγκεκριμένη οντότητα ενώ το πεδίο

θέση περιγράφει τη σχετική θέση της οντότητας στη σελίδα δίσκου, ως προς την αρχή της σελίδας. Με τη μέθοδο αυτή όταν κάποια οντότητα μεγαλώνει, αλλάζει η θέση στην οποία αποθηκεύεται στο δίσκο, αλλάζοντας έτσι και το αναγνωριστικό για τη συγκεκριμένη οντότητα. Υπάρχουν δύο τρόποι για να διατηρηθεί η συνέπεια της βάσης:

α. Όταν μια οντότητα αλλάζει θέση στο δίσκο, δεν αλλάζει το αναγνωριστικό της, αλλά η προηγούμενη θέση στην οποία βρισκόταν αποκτάει ένα δείκτη, ο οποίος περιέχει το νέο αναγνωριστικό για την οντότητα, δηλαδή την καινούρια θέση στην οποία βρίσκεται η οντότητα. Αυτό όμως σημαίνει ότι η προσπέλαση σε μια τέτοια οντότητα απαιτεί τουλάχιστο δυο προσπελάσεις στο δίσκο (disk accesses), μια για να βρεθεί το νέο αναγνωριστικό και μια για να γίνει ουσιαστικά η προσπέλαση στα δεδομένα της οντότητας. Η μέθοδος αυτή χρησιμοποιείται στο σύστημα O_2.

β. Όταν μια οντότητα αλλάζει θέση, αλλάζει και το αναγνωριστικό για τη συγκεκριμένη οντότητα. Αυτό σημαίνει ότι κάθε αναφορά στην οντότητα αυτή μέσω του αναγνωριστικού της πρέπει να αλλαχτεί και το νέο αναγνωριστικό να πάρει τη θέση της. Η αντιμετώπιση αυτή επιφέρει χειροτέρευση στην απόδοση του συστήματος, αφού αλλαγή της θέσης μιας οντότητας επιφέρει αλλαγές σε όλες τις άλλες οντότητες που την αναφέρουν.

Στο σύστημα TELOS τα αναγνωριστικά συστήματος παραμένουν αμετάβλητα καθ'όλη τη ζωή των οντοτήτων που προσδιορίζουν, όπως και στο σύστημα Mneme [**?**] . Αν και οι οντότητες αφότου δημιουργηθούν παραμένουν αποθηκευμένες στην ίδια πάντα θέση στο δίσκο, το αναγνωριστικό συστήματος για κάθε οντότητα δεν περιλαμβάνει τη διεύθυνση που αυτή βρίσκεται αποθηκευμένη. Με το τρόπο αυτό, όταν μια οντότητα διαγράφεται από τη βάση δεδομένων το αναγνωριστικό συστήματος που την προσδιόριζε μπορεί να ανατεθεί σε μια νέα οντότητα που θα δημιουργηθεί αργότερα και πιθανότατα θα αποθηκευτεί σε μια νέα θέση στο δίσκο.

Τα αναγνωριστικά συστήματος υλοποιούνται με τη μορφή ακεραίων αριθμών από τους οποίους χρησιμοποιούνται 30 bits. Έτσι, το μέγεθος της βάσης περιορίζεται από το μέγεθος των αναγνωριστικών οντοτήτων σε 2^{30} οντότητες συνολικά. Επειδή όμως η δομή για τα αναγνωριστικά συστήματος είναι διάφανη (transparent) για το σύστημα, μπορεί να αλλαχτεί πολύ εύκολα ώστε να υποστηρίζει περισσότερες οντότητες. Τα αναγνωριστικά συστήματος ανατίθενται στις οντότητες κατά αύξουσα σειρά, τη στιγμή που αυτές δημιουργούνται από το κατάλογο συστήματος.

4.2.2 Η αρχιτεκτονική του καταλόγου συστήματος

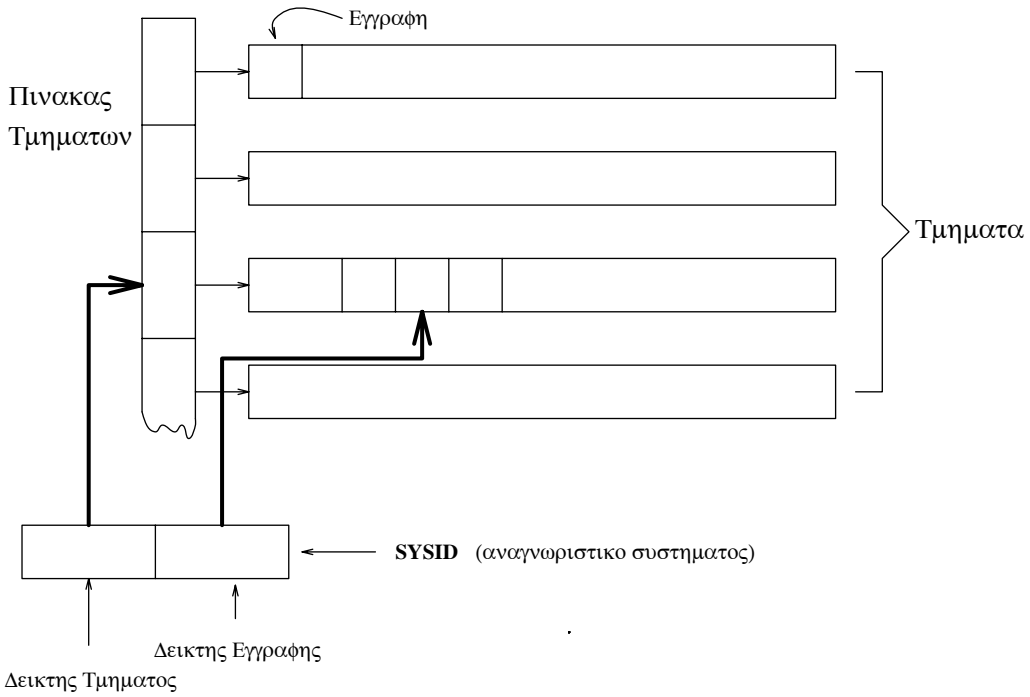
Ο κατάλογος συστήματος περιέχει πληροφορίες που αφορούν κάθε οντότητα που αποθηκεύεται στη βάση δεδομένων και οι οποίες δεν περιλαμβάνονται στις δομές αναπαράστασης για τις διάφορες οντότητες. Ο κατάλογος συστήματος χρησιμοποιώντας τη διασύνδεση (interface) που παρέχει ο μηχανισμός αποθήκευσης οντοτήτων εκτελεί τις παρακάτω λειτουργίες:

- δημιουργία νέων οντοτήτων
- διαγραφή οντοτήτων από τη βάση
- προσπέλαση στις οντότητες της βάσης
- ενημέρωση της βάσης με τις νέες οντότητες ή οντότητες που έχουν μεταβληθεί

Οι εφαρμογές που χρησιμοποιούν τη βάση TELOS μπορούν να προσπελάσουν τις οντότητες της βάσης μονάχα μέσω κλήσεων στο κατάλογο συστήματος. Έτσι, ο κατάλογος συστήματος αποτελεί τη μοναδική διασύνδεση των εφαρμογών αυτών με τα δεδομένα της βάσης.

Ο κατάλογος συστήματος είναι ένας *πίνακας από εγγραφές* (entries) για κάθε οντότητα της βάσης. Έτσι, ο κατάλογος συστήματος μπορεί να περιέχει μέχρι 2^{30} εγγραφές, καθώς αυτό είναι το μέγιστο πλήθος οντοτήτων για κάθε βάση δεδομένων στο σύστημα TELOS. Επειδή λοιπόν ο κατάλογος συστήματος μπορεί να είναι πολύ μεγάλος για να χωράει στη μνήμη κάποιου υπολογιστή, οργανώνεται εικονικά σαν *πίνακας από τμήματα* (segment table) με βαθμό εμμεσότητας 1 (single indirection). Κάθε τμήμα του πίνακα αυτού περιέχει ένα καθορισμένο αριθμό από εγγραφές. Καθώς όμως το πλήθος των τμημάτων αυτών είναι πολύ μεγάλο, δε χρησιμοποιείται πίνακας δεικτών για τα διάφορα τμήματα, αλλά τα τμήματα αυτά αποτελούν τις εγγραφές για το μηχανισμό κρυφής μνήμης του καταλόγου συστήματος. Η περιγραφή όλων των μηχανισμών κρυφής μνήμης για το σύστημα διαχείρισης οντοτήτων στην TELOS παρουσιάζεται στο κεφάλαιο 6.

Η προσπέλαση στις εγγραφές του καταλόγου συστήματος γίνεται μέσω του μοναδικού *αναγνωριστικού συστήματος* για την κάθε οντότητα. Για το σκοπό αυτό τα αναγνωριστικά συστήματος που υλοποιούνται με τη μορφή ακεραίων αριθμών, θεωρούνται ως διανύσματα της μορφής <*δείκτης τμήματος, δείκτης εγγραφής*>. Το πεδίο *δείκτης τμήματος* που αποτελείται από τα δυαδικά ψηφία υψηλής τάξης (high order bits) του αναγνωριστικού συστήματος καθορίζει το τμήμα του καταλόγου συστήματος στο οποίο βρίσκεται η εγγραφή για μια οντότητα. Το πεδίο *δείκτης εγγραφής* που αποτελείται από τα δυαδικά ψηφία χαμηλής τάξης (low order bits) του αναγνωριστικού συστήματος, καθορίζει τη συγκεκριμένη εγγραφή για



Σχήμα 4.1: Οργάνωση και δεικτοδότηση του καταλόγου συστήματος

Ο *Πίνακας Τμημάτων* που φαίνεται στο παραπάνω σχήμα είναι *εικονικός*, δηλαδή παρουσιάζεται μονάχα για να δείξει το τρόπο που γίνεται η δεικτοδότηση του καταλόγου συστήματος.

μια οντότητα στο τμήμα του καταλόγου συστήματος που την περιέχει. Ο τρόπος δεικτοδότησης του καταλόγου συστήματος και η οργάνωσή του ως πίνακα από τμήματα φαίνεται στο σχήμα **??**.

4.2.3 Εγγραφές του καταλόγου συστήματος

Σε κάθε οντότητα που ορίζεται σε κάποια βάση δεδομένων στο σύστημα TELOS αντιστοιχεί μια εγγραφή του καταλόγου συστήματος. Όπως έχουμε ήδη αναφέρει, η εγγραφή αυτή περιλαμβάνει πληροφορία που αφορά τη σταθερή κατάσταση (*persistency*) της οντότητας, όπως αυτή αποθηκεύεται στο δίσκο και την ενδιάμεση κατάσταση που μπορεί να βρίσκεται μια οντότητα που έχει φορτωθεί στη μνήμη.

Καθώς ο κατάλογος συστήματος αποτελεί το μέσο για την προσπέλαση στις οντότητες μιας βάσης, η σταθερή κατάσταση κάθε οντότητας περιλαμβάνει τις εξής πληροφορίες:

- θέση στο δίσκο όπου αποθηκεύεται η οντότητα,
- θέση στο δίσκο όπου αποθηκεύονται οι δομές επέκτασης για την οντότητα, αν το πλήθος των συνδέσεων της οντότητας αυτής με άλλες οντότητες υπερβαίνει το μέγεθος

του συγκεκριμένου τύπου που χρησιμοποιείται για την αναπαράσταση της οντότητας αυτής,

- το συνολικό μέγεθος των δομών επέκτασης για την οντότητα αυτή στο δίσκο, αν η αναπαράσταση της οντότητας περιλαμβάνει τέτοιες δομές και
- τον τύπο της οντότητας, ο οποίος περιγράφει το επίπεδο ταξινόμησης και τις κλάσεις του συστήματος στις οποίες ταξινομείται η οντότητα αυτή.

Για τις οντότητες που βρίσκονται στη μνήμη η εγγραφή του καταλόγου συστήματος περιλαμβάνει τις εξής πληροφορίες:

- τη θέση στη μνήμη όπου βρίσκεται η οντότητα,
- την κατάσταση της οντότητας. Μια οντότητα μπορεί να έχει μόλις δημιουργεί, διαγραφεί, να έχει υποστεί αλλαγές, ή τίποτα από τα προηγούμενα,
- τη χρησιμότητα της οντότητας. Μια οντότητα μπορεί να παραμένει μονίμως στη μνήμη ή να μην χρειάζεται πλέον, οπότε μπορεί να απελευθερωθεί ο χώρος της μνήμης που καταλαμβάνει.

4.2.4 Οργάνωση του τμήματος εγγραφών

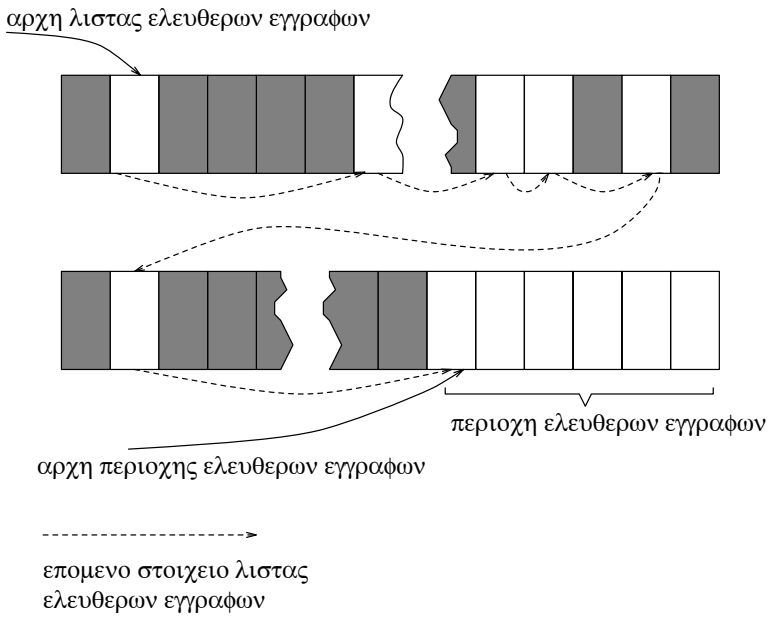
Είναι φανερό ότι μόνο η πληροφορία για τη σταθερή κατάσταση κάθε οντότητας είναι απαραίτητο να αποθηκεύεται στο δίσκο, ενώ η υπόλοιπη πληροφορία που περιέχεται στις εγγραφές του καταλόγου συστήματος είναι απαραίτητη μονάχα για τις οντότητες που βρίσκονται στη μνήμη.

Για το λόγο αυτό τα *τμήματα εγγραφών* χωρίζονται σε δύο πίνακες, οι οποίοι περιέχουν την πληροφορία για τη μόνιμη κατάσταση των οντοτήτων και για τις οντότητες που βρίσκονται στη μνήμη αντίστοιχα. Έτσι, στο δίσκο αποθηκεύονται μονάχα τα περιεχόμενα του πρώτου πίνακα αποφεύγοντας τη σπατάλη χώρου.

4.2.5 Διαχείριση των εγγραφών του καταλόγου συστήματος

Οι αρχικές εγγραφές του καταλόγου συστήματος χρησιμοποιούνται από τις εγγενείς οντότητες που περιλαμβάνονται σε κάθε βάση. Οι οντότητες αυτές παραμένουν μόνιμα στη μνήμη και δεν μπορούν να διαγραφούν αφού ομαδοποιούν όλες τις άλλες οντότητες που ορίζονται από το χρήστη.

Όταν δημιουργείται μια νέα οντότητα, ο μηχανισμός διαχείρισης του καταλόγου συστήματος επιλέγει μια διαθέσιμη εγγραφή του καταλόγου συστήματος και δίνει το αναγνωριστικό συστήματος που καθορίζει την εγγραφή αυτή ως αναγνωριστικό για τη νέα οντότητα.



Σχήμα 4.2: Η λίστα ελεύθερων εγγραφών και η περιοχή ελεύθερων εγγραφών

Οι γραμμοσκιασμένες εγγραφές έχουν ανατεθεί σε οντότητες, ενώ οι λευκές αποτελούν ελεύθερες εγγραφές.

Καθώς όμως τα περιεχόμενα μιας βάσης αλλάζουν, είναι δυνατό διάφορες οντότητες να διαγράφονται και έτσι να απελευθερώνονται οι εγγραφές του καταλόγου συστήματος που καταλαμβάνουν οι οντότητες αυτές. Οι ελεύθερες εγγραφές του καταλόγου συστήματος βρίσκονται σε μια απλά συνδεδεμένη *λίστα ελεύθερων εγγραφών*. Κάθε φορά που δημιουργείται μια νέα οντότητα, το αναγνωριστικό συστήματος που παραχωρείται σ'αυτήν είναι το πρώτο στοιχείο της λίστας ελεύθερων εγγραφών. Επειδή οι διαγραφές οντοτήτων στις εφαρμογές που έχουν χρησιμοποιήσει μέχρι στιγμής το περιβάλλον TELOS δεν είναι συχνές, ο κατάλογος του συστήματος είναι πυκνός και η λίστα ελεύθερων εγγραφών περιέχει λίγα στοιχεία. Το τελευταίο στοιχείο της λίστας ελεύθερων εγγραφών αποτελεί την αρχή της *περιοχής ελεύθερων εγγραφών*. Η περιοχή αυτή είναι πάντα συνεχής και αποτελεί το τέλος του καταλόγου συστήματος. Το σχήμα **??** δείχνει τη λίστα ελεύθερων εγγραφών και τη περιοχή ελεύθερων εγγραφών για τον κατάλογο συστήματος.

4.2.6 Σύστημα αποθήκευσης

Οι εγγραφές του καταλόγου συστήματος αποθηκεύονται σε ένα δυαδικό αρχείο που δημιουργείται και διαχειρίζεται μέσω κλήσεων στις ρουτίνες συστήματος (system calls) του συστήματος αρχείων που υποστηρίζει το λειτουργικό σύστημα UNIX. Στην αρχή του αρχείου βρίσκεται μία δομή που ονομάζεται επικεφαλίδα αρχείου (file header), όπου αποθηκεύονται

ο δείκτης του πρώτου στοιχείου της λίστας ελεύθερων εγγραφών, ο δείκτης στην αρχή της περιοχής ελεύθερων εγγραφών, το μέγεθος του καταλόγου συστήματος και το μέγεθος του αρχείου.

Τα τμήματα εγγραφών έχουν μέγεθος ίσο με το μέγεθος των σελίδων δίσκου (disk pages) και κάθε τμήμα εγγραφών καταλαμβάνει ακριβώς μια σελίδα δίσκου αποφεύγοντας εξωτερικό κατακερματισμό (external fragmentation) των σελίδων δίσκου. Έτσι, κάθε προσπέλαση σε μία εγγραφή του καταλόγου συστήματος απαιτεί το πολύ μια προσπέλαση στο δίσκο, αν το τμήμα εγγραφών που την περιέχει δεν βρίσκεται στην κρυφή μνήμη του καταλόγου συστήματος.

Η εγγραφή στο δίσκο τμημάτων εγγραφών, των οποίων τα περιεχόμενα έχουν μεταβληθεί, γίνεται μονάχα στο τέλος της δοσοληψίας που προκάλεσε τις πιθανές αλλαγές. Έτσι, όταν μια δοσοληψία αποτυγχάνει, δε γίνεται καμμιά αλλαγή στα περιεχόμενα του αρχείου. Στην περίπτωση αυτή, διαγράφονται τα τμήματα εγγραφών από τη μνήμη βελτιστοποιώντας το χρόνο απόκρισης του συστήματος με την τεχνική no rollback.

4.3 Διαχείριση Οντοτήτων

Σε αντίθεση με τα άλλα οντοκεντρικά συστήματα διαχείρισης βάσεων δεδομένων που χρησιμοποιούν σχεσιακά συστήματα αποθήκευσης στο δίσκο για την αποθήκευση των διαφόρων οντοτήτων, στο σύστημα διαχείρισης αποθήκευσης για τη γλώσσα TELOS οι δομές που αναπαριστάνουν τις οντότητες στη μνήμη και το δίσκο είναι οι ίδιες.

Η μέθοδος αυτή βελτιώνει την απόδοση του συστήματος διαχείρισης οντοτήτων, εφόσον δε χρειάζεται μετάφραση των δομών αναπαράστασης των οντοτήτων από την αναπαράστασή τους στο δίσκο στην αναπαράστασή τους στη μνήμη και αντίστροφα. Έτσι, η προσπέλαση σε κάθε οντότητα που βρίσκεται στο δίσκο γίνεται με πιο αποδοτικό και γρήγορο τρόπο απ'ότι στα συστήματα Iris, POSTGRES και O_2 που έχουν περιγραφεί.

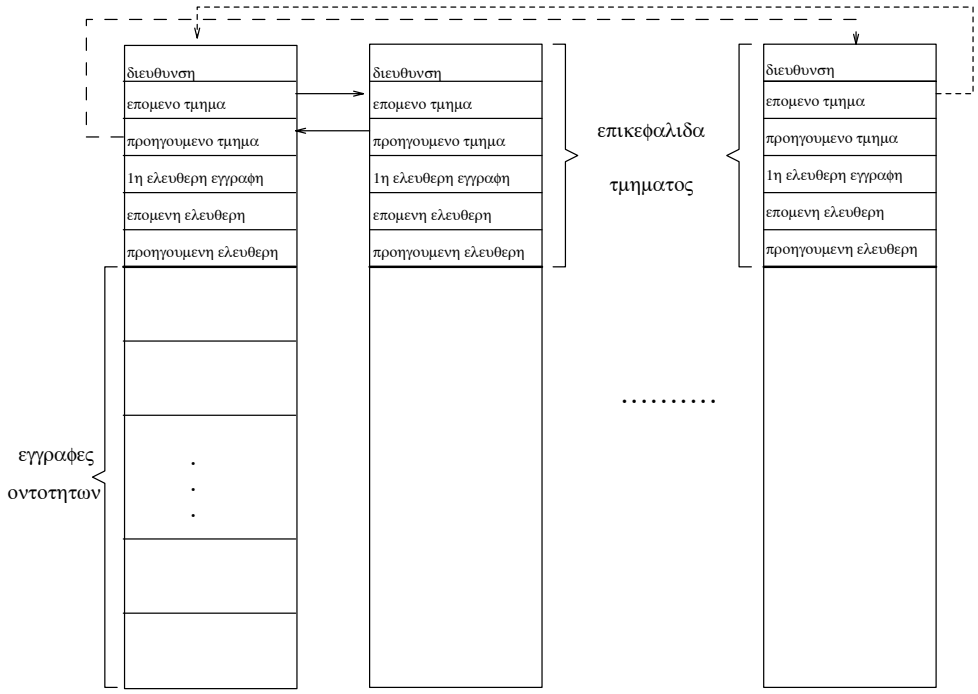
4.3.1 Τμήματα οντοτήτων (object blocks)

Οι οντότητες οργανώνονται σε *τμήματα οντοτήτων* και τα τμήματα οντοτήτων αποτελούν τη μονάδα μεταφοράς από το δίσκο στη μνήμη και αντίστροφα. Τα τμήματα οντοτήτων έχουν σταθερό μέγεθος (ίσο με το μέγεθος της σελίδας δίσκου) και περιέχουν πάντοτε οντότητες ενός μονάχα τύπου. Ο τύπος αυτός αντιστοιχεί στις δομές αναπαράστασης οντοτήτων που περιγράφηκαν στο τρίτο κεφάλαιο, δηλαδή τις δομές *class node*, *link class*, *token node* και *link node*. Επίσης, οι δομές επέκτασης (*extensions*) για τις διάφορες οντότητες αποθηκεύονται σε χωριστά τμήματα οντοτήτων.

Καθώς οι διαφορετικές δομές για την αναπαράσταση των οντοτήτων έχουν διαφορετικά μεγέθη η καθεμιά, το πλήθος των οντοτήτων που περιέχονται σε κάθε τμήμα οντοτήτων εξαρτάται από το μέγεθος των δομών αποθήκευσης για τις οντότητες αυτές. Τα τμήματα οντοτήτων περιέχουν στο τέλος τους αχρησιμοποίητο χώρο, γιατί δεν επιτρέπουμε να βρίσκονται οι δομές αποθήκευσης ανάμεσα σε δυο διαφορετικές σελίδες δίσκου. Έτσι, υπάρχει κάποια μικρή σπατάλη χώρου που κυμαίνεται από 1% έως 4.7% σε κάθε τμήμα οντοτήτων. Η σπατάλη αυτή όμως βελτιώνει τις επιδόσεις του συστήματος διαχείρισης οντοτήτων, αφού για την πρόσβαση σε κάθε οντότητα αρκεί το πολύ μια προσπέλαση στο δίσκο. Αν όμως δεν υπήρχε αυτή η σπατάλη χώρου, τότε η πρόσβαση σε κάποιες οντότητες θα απαιτούσε δύο προσπελάσεις στο δίσκο, οι οποίες είναι τρεις τάξεις μεγέθους αργότερες από τις προσβάσεις στη μνήμη. Επειδή η μεταφορά από το δίσκο στη μνήμη γίνεται σε μονάδες τμημάτων οντοτήτων, η πρόσβαση σε μια οντότητα που βρίσκεται στο δίσκο οδηγεί σε προανάκληση (prefetching) όλων των άλλων οντοτήτων που βρίσκονται στο ίδιο τμήμα οντοτήτων με αυτή. Επίσης, η αποθήκευση οντοτήτων ίδιου τύπου στο ίδιο τμήμα οντοτήτων, δημιουργεί έναν απλό μηχανισμό ομαδοποίησης (clustering) καθώς οι οντότητες αποθηκεύονται σε κοντινές θέσεις στο δίσκο ή στη μνήμη.

Τα τμήματα οντοτήτων που περιέχουν οντότητες του ίδιου τύπου οργανώνονται σε διπλά συνδεδεμένες λίστες τμημάτων οντοτήτων. Στο σύστημα διαχείρισης οντοτήτων για τη γλώσσα TELOS υπάρχουν 5 διαφορετικές λίστες τμημάτων οντοτήτων που αντιστοιχούν στις 5 διαφορετικές δομές αποθήκευσης οντοτήτων. Η δομή των τμημάτων οντοτήτων και η λίστα που δημιουργείται από τα τμήματα οντοτήτων για μια δομή αποθήκευσης παρουσιάζονται στο σχήμα **??**.

Κάθε τμήμα οντοτήτων περιέχει μια επικεφαλίδα που περιέχει τους δείκτες με τις απόλυτες διευθύνσεις στο δίσκο του επόμενου και προηγούμενου τμήματος στη διπλά συνδεδεμένη λίστα τμημάτων οντοτήτων και τη διεύθυνση στο δίσκο του ίδιου του τμήματος. Επίσης, περιέχει τους δείκτες για μια διπλά συνδεδεμένη λίστα αχρησιμοποίητων δομών αποθήκευσης μέσα στο ίδιο το τμήμα οντοτήτων. Η λίστα αυτή είναι διπλά συνδεδεμένη προκειμένου να γίνεται αποδοτικότερα η διαγραφή και εισαγωγή δομών αποθήκευσης. Η δομή της επικεφαλίδας για κάθε τμήμα οντοτήτων παρουσιάζεται στο σχήμα **??**. Οι δείκτες της λίστας αχρησιμοποίητων δομών αποθήκευσης είναι σχετικοί ως προς την απόλυτη διεύθυνση της αρχής του τμήματος οντοτήτων στο οποίο αναφέρονται. Η μέθοδος αυτή βελτιστοποιεί τη διαγραφή και εισαγωγή δομών αποθήκευσης γιατί υλοποιείται μέσω προσπελάσεων στη μνήμη σε μονοδιάστατο πίνακα. Η διπλά συνδεδεμένη λίστα τμημάτων οντοτήτων βελτιστοποιεί τη διεργασία δημιουργίας νέων τμημάτων οντοτήτων όταν η βάση μεγαλώνει, και τη διεργασία διαγραφής τμημάτων οντοτήτων όταν διαγράφονται όλες οι δομές αποθήκευσης

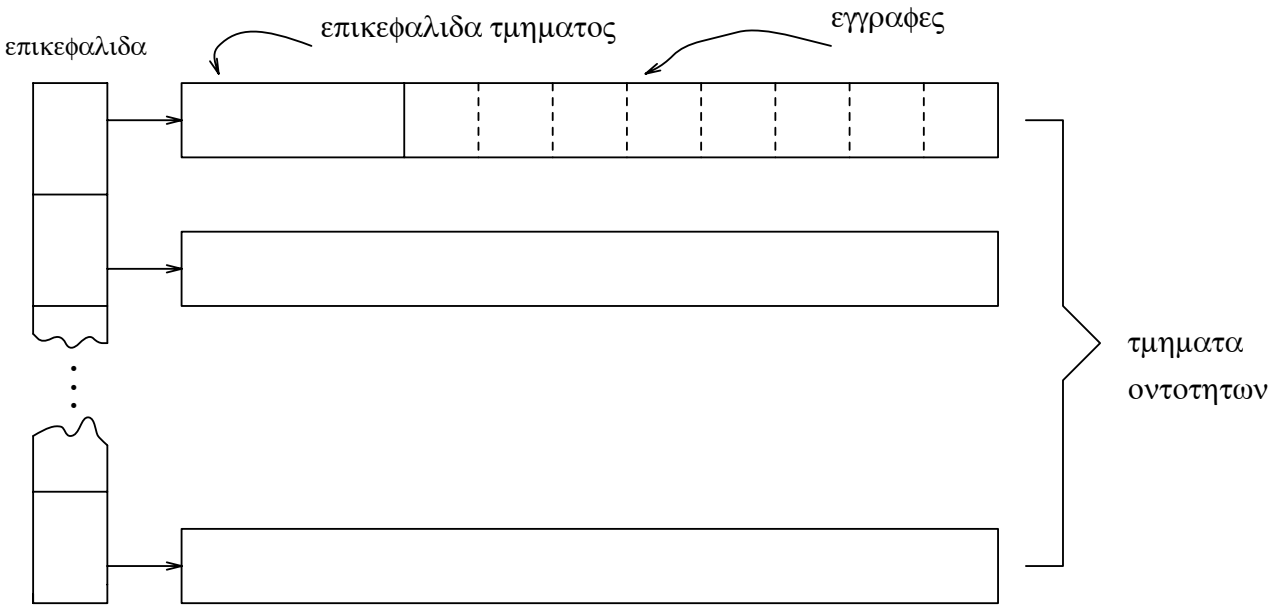


Σχήμα 4.3: Σύνδεση και περιγραφή των *τμημάτων οντοτήτων* για μια από τις δομές αποθήκευσης

που περιέχουν τα τμήματα αυτά.

Προκειμένου να βελτιστοποιηθεί ο χρόνος προσπέλασης στις οντότητες, χρησιμοποιούνται μηχανισμοί κρυφής μνήμης, οι οποίοι περιέχουν τα τμήματα οντοτήτων που βρίσκονται στη μνήμη. Οι μηχανισμοί αυτοί και οι διάφοροι αλγόριθμοι αντικατάστασης περιγράφονται στο έκτο κεφάλαιο.

Όταν απαιτείται πρόσβαση σε μια οντότητα στη μνήμη, η δομή αναπαράστασης για την οντότητα αντιγράφεται από το τμήμα οντοτήτων που την περιέχει σε μια νέα θέση στη μνήμη, μέσω του καταλόγου συστήματος. Η πρόσβαση θα μπορούσε να γίνει απευθείας στο τμήμα οντοτήτων που περιέχει τη δομή αναπαράστασης για κάθε οντότητα, χωρίς την ανάγκη δημιουργίας των αντιγράφων. Με τον τρόπο αυτό θα βελτιωνόταν η απόδοση του συστήματος και θα απαιτούνταν λιγότερος αποθηκευτικός χώρος. Επίσης, θα γινόταν πιο απλή η διαχείριση των οντοτήτων αφού δε θα γινόταν διάκριση ανάμεσα σε οντότητες σε σταθερή κατάσταση (persistent objects) και στα αντίγραφά τους στη μνήμη (non-persistent objects), όπως συμβαίνει στο σύστημα αποθήκευσης Cricket [**?**]. Δυστυχώς όμως η έκδοση της γλώσσας C++ που χρησιμοποιείται για την τρέχουσα υλοποίηση παρουσιάζει προβλήματα συμβατότητας σε διαφορετικά περιβάλλοντα.



Σχήμα 4.4: Η οργάνωση του αρχείου με τα *τμήματα εγγραφών*

4.3.2 Σύστημα αποθήκευσης

Τα περιεχόμενα των τμημάτων οντοτήτων μεταβάλλονται μονάχα όταν μια δοσοληψία τελειώνει επιτυχώς. Χρησιμοποιώντας την πολιτική write back, τα τμήματα οντοτήτων που έχουν μεταβληθεί κατά τη διάρκεια μιας δοσοληψίας γράφονται στο δίσκο όταν διαγράφονται από την κρυφή μνήμη. Έτσι, οι μεταβολές που συμβαίνουν κατά τη διάρκεια μιας δοσοληψίας παραμένουν στη μνήμη μέχρι να τελειώσει η δοσοληψία.

Όλα τα τμήματα οντοτήτων αποθηκεύονται σε ένα δυαδικό αρχείο στο σύστημα αρχείων του λειτουργικού συστήματος UNIX. Εναλλακτικά, θα μπορούσε να χρησιμοποιηθεί ένα αρχείο για κάθε λίστα από τμήματα οντοτήτων που αντιστοιχούν σε μια από τις δομές αποθήκευσης. Η δεύτερη λύση όμως θα δημιουργούσε πολλά αρχεία και θα ήταν λιγότερο αποδοτική λόγω περιορισμών στις διεργασίες από το λειτουργικό σύστημα UNIX.

Το αρχείο με τα τμήματα οντοτήτων οργανώνεται ανάλογα με τον κατάλογο συστήματος όπως φαίνεται στο σχήμα **??**. Η προσπέλαση στο δίσκο γίνεται μέσω των πιο σημαντικών bits από τη διεύθυνση μιας οντότητας στο δίσκο. Τα bits αυτά καθορίζουν τη θέση που είναι αποθηκευμένο το τμήμα εγγραφών που περιέχει τη συγκεκριμένη οντότητα.

Η πρώτη σελίδα δίσκου του αρχείου αποτελεί την επικεφαλίδα του αρχείου. Εκεί περιλαμβάνονται δείκτες για το πρώτο στοιχείο της λίστας τμημάτων οντοτήτων για κάθε τύπο αποθήκευσης, καθώς και τα πεδία με το μέγεθος του αρχείου και το σύνολο των τμημάτων εγγραφών.

Οι εγγενείς οντότητες αναπαριστάνονται μέσω ειδικών τύπων αποθήκευσης, οι οποίοι αποθηκεύονται στο υπόλοιπο τμήμα της πρώτης σελίδας δίσκου. Η προσπέλαση σε αυτές γίνεται μέσω της απόλυτης διεύθυνσής τους στο δίσκο.

4.4 Διαχείριση δοσοληψιών

Ένας από τους βασικούς μηχανισμούς που διαθέτουν όλα τα συστήματα διαχείρισης βάσεων δεδομένων είναι ο μηχανισμός ταυτόχρονης προσπέλασης στη βάση από πολλές διεργασίες που μπορούν να δημιουργούν διάφοροι χρήστες.

Η διαχείριση δοσοληψιών (transaction management) είναι ένας μηχανισμός που καθορίζει τον τρόπο με τον οποίο διάφορα προγράμματα μπορούν να μοιράζονται μια κοινή βάση δεδομένων. Συνήθως μια δοσοληψία θεωρείται ως η μονάδα ταυτόχρονης προσπέλασης (concurrency) και η μονάδα επανάκτησης (recovery) του συστήματος διαχείρισης βάσεων δεδομένων. Αφού θεωρείται μονάδα ταυτόχρονης προσπέλασης, τα βήματα από διάφορες δοσοληψίες μπορούν να αναμειχθούν μεταξύ τους χωρίς να επιδρούν το ένα στο άλλο, ενώ ως μονάδα επανάκτησης του συστήματος, μια δοσοληψία επιτυγχάνει πλήρως και όλες οι μεταβολές που πραγματοποιεί γράφονται στη βάση δεδομένων, ή αποτυγχάνει πλήρως χωρίς να επιφέρει καμιά αλλαγή στη βάση δεδομένων.

Μια δοσοληψία αποτελείται από μια σειρά προσπελάσεων ανάγνωσης και εγγραφής στη βάση δεδομένων και έχει δυο ιδιότητες: *ατομικότητα* (atomicity) και *σειριακότητα* (serializability). Σύμφωνα με την πρώτη ιδιότητα η ακολουθία προσπελάσεων ανάγνωσης και εγγραφών στη βάση που πραγματοποιούνται από μια δοσοληψία θεωρούνται ως ατομική πράξη στη βάση δεδομένων, η οποία επιτυγχάνει ή αποτυγχάνει πλήρως. Η δεύτερη ιδιότητα σημαίνει ότι τα αποτελέσματα πολλών δοσοληψιών που συμβαίνουν ταυτόχρονα στη βάση είναι ίδια με τα αποτελέσματα των ίδιων δοσοληψιών αν αυτές εκτελεστούν σειριακά και εξασφαλίζει ότι τα αποτελέσματα κάθε δοσοληψίας είναι ανεξάρτητα από τα αποτελέσματα οποιασδήποτε άλλης δοσοληψίας.

4.4.1 Διαχείριση δοσοληψιών στην TELOS

Ο μηχανισμός διαχείρισης δοσοληψιών για τη βάση δεδομένων που δημιουργείται με περιγραφές σε γλώσσα TELOS, είναι ένας απλός μηχανισμός που χρησιμοποιεί το πρωτόκολλο κλειδώματος δύο φάσεων (two phase locking protocol). Η επιλογή του συγκεκριμένου μηχανισμού έγινε προκειμένου να εξασφαλίζεται η ταυτόχρονη προσπέλαση στη βάση δεδομένων από πολλές διεργασίες, οι οποίες μπορεί απλώς να διαβάζουν τα περιεχόμενά της ή και να τα μεταβάλλουν. Επίσης, ο μηχανισμός αυτός μπορεί να υλοποιηθεί εύκολα

παρέχοντας ασφαλή μεταβολή στη βάση χωρίς απώλειες δεδομένων.

Ο μηχανισμός κλειδώματος δύο φάσεων, όπως περιγράφεται από το όνομά του, λειτουργεί σε δύο στάδια. Στο πρώτο στάδιο (φάση) που ονομάζεται φάση ανάπτυξης (growing phase) η δοσοληψία μπορεί να κλειδώνει διάφορα δεδομένα της βάσης αλλά δεν μπορεί να ξεκλειδώνει δεδομένα που έχει ήδη κλειδώσει, ενώ στη δεύτερη φάση που ονομάζεται φάση συρρίκνωσης (shrinking phase) η δοσοληψία ξεκλειδώνει δεδομένα αλλά δεν μπορεί να κλειδώνει δεδομένα. Στην τρέχουσα υλοποίηση το κλείδωμα εφαρμόζεται σε ολόκληρη τη βάση και όχι σε τμήμα της.

Όταν μια δοσοληψία πρόκειται να αλλάξει τα περιεχόμενα της βάσης, η δοσοληψία αυτή αρχικά κλειδώνει τη βάση σε μορφή ανάγνωσης και κατόπιν σε μορφή εγγραφής. Τότε πραγματοποιούνται όλες οι μεταβολές στα αντίγραφα των δεδομένων της μνήμης και μονάχα όταν ολοκληρωθεί ο έλεγχος της συνέπειας των δεδομένων που δημιουργήθηκαν από τη δοσοληψία μπορούν αυτά να γραφτούν στη βάση. Αφού τα νέα δεδομένα γραφτούν στη βάση, η δοσοληψία απελευθερώνει το κλείδωμα σε μορφή εγγραφής και κατόπιν το κλείδωμα σε μορφή ανάγνωσης, επιτρέποντας έτσι σε άλλες δοσοληψίες την αποκλειστική προσπέλαση στη βάση για εγγραφές. Ο μηχανισμός αυτός εξασφαλίζει και τις δυο ιδιότητες των δοσοληψιών, δηλαδή την ατομικότητα και σειριακότητα, με επιτυχία.

Όταν μια διεργασία πρόκειται να διαβάσει κάποια δεδομένα της βάσης μέσω κάποιας πράξης, για παράδειγμα πραγματοποιώντας μια ερώτηση, εξασφαλίζεται το κλείδωμα για διάβασμα στη βάση πριν από οποιαδήποτε προσπέλαση στη βάση, το οποίο απελευθερώνεται μόλις ολοκληρωθεί η πράξη.

Η υλοποίηση των δυο τρόπων προσπέλασης στη βάση πραγματοποιείται μέσω κλήσεων σε συναρτήσεις συστήματος (UNIX) που επιδρούν σ' ένα ειδικό αρχείο και εξασφαλίζουν τις δυο επιθυμητές μορφές κλειδώματος. Έτσι, τα υπόλοιπα αρχεία που περιλαμβάνουν τη βάση δεν κλειδώνονται αλλά παραμένουν προσπελάσιμα από όλες τις διεργασίες. Πριν όμως από κάθε προσπέλαση στα δεδομένα της βάσης, ελέγχεται αν η συγκεκριμένη διεργασία μπορεί να αποκτήσει κλείδωμα διαβάσματος στη βάση. Είναι δυνατό κατά τη διάρκεια της ταυτόχρονης εκτέλεσης διαφόρων διεργασιών, τα περιεχόμενα της βάσης να αλλάξουν από κάποιες δοσοληψίες. Έτσι, όποτε αλλάζουν τα περιεχόμενα της βάσης, ενημερώνεται ένας μετρητής στο ειδικό αρχείο που χρησιμοποιείται για το μηχανισμό διαχείρισης δοσοληψιών και μέσω αυτού του μετρητή ενημερώνονται με τη σειρά τους όλες οι διεργασίες, που εκτελούνται ταυτόχρονα για την αλλαγή της βάσης. Στην περίπτωση αυτή, οι διεργασίες που χρησιμοποιούν δεδομένα από την προηγούμενη κατάσταση της βάσης, ακυρώνουν τα περιεχόμενα της κρυφής μνήμης που χρησιμοποιούν για τις διάφορες δομές αποθήκευσης και διαβάζουν τη νέα κατάσταση της βάσης από το δίσκο.

Κάθε διεργασία περιλαμβάνει δικό της αντίγραφο των δομών διαχείρισης οντοτήτων και προσπελαύνει απευθείας τα περιεχόμενα των διαφόρων αρχείων. Εναλλακτικά θα μπορούσε να υπήρχε μια διεργασία εξυπηρέτησης, η οποία θα αποτελούσε και τη μοναδική διεργασία που θα πραγματοποιούσε κάθε προσπέλαση στο δίσκο, και η οποία θα χρησιμοποιούσε μηχανισμό message passing για την επικοινωνία με τις διάφορες άλλες διεργασίες πελάτη που θα επιθυμούσαν προσπέλαση στη βάση δεδομένων.

Αν και η απόδοση του συστήματος αποθήκευσης και διαχείρισης οντοτήτων κρίνεται ικανοποιητική, θα μπορούσε να χρησιμοποιηθεί η αρχιτεκτονική *πελάτη-εξυπηρετητή* για τη δημιουργία πιο εξελιγμένων μηχανισμών διαχείρισης δοσοληψιών. Σε μια τέτοια περίπτωση, θα παρέμενε ως έχει ο μηχανισμός διαχείρισης οντοτήτων, και απλώς θα χρειαζόταν να προστεθεί ένα επίπεδο με το μηχανισμό διαχείρισης δοσοληψιών στη διεργασία εξυπηρέτησης.

Κεφάλαιο 5

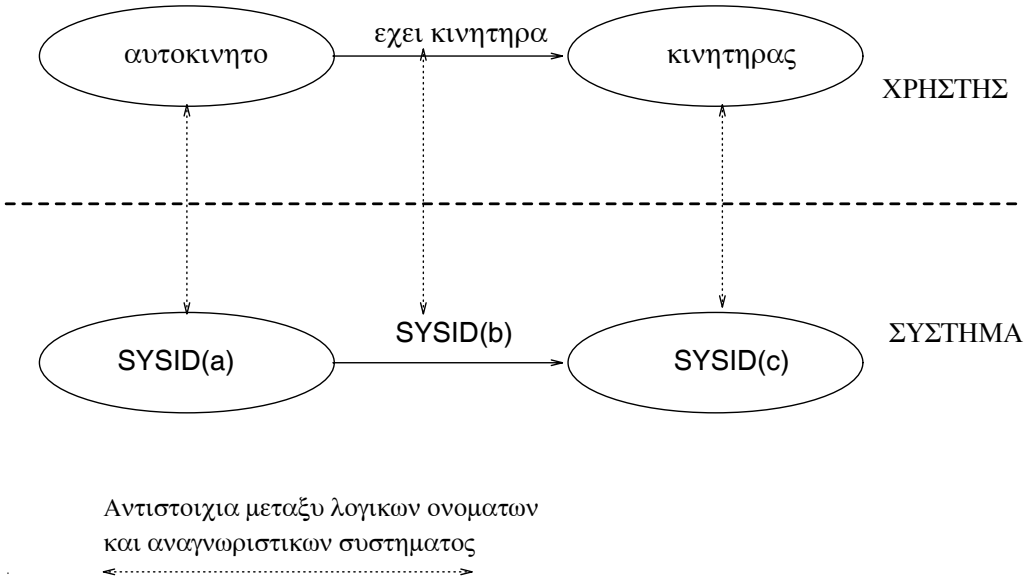
Ο κατάλογος λογικών ονομάτων

5.1 Εισαγωγή

Κάθε οντότητα στη γλώσσα TELOS μπορεί να διακριθεί με μοναδικό τρόπο μέσω του *αναγνωριστικού συστήματος* που της παραχωρείται από το σύστημα τη στιγμή που δημιουργείται η οντότητα αυτή. Τα αναγνωριστικά συστήματος χρησιμοποιούνται μονάχα από το σύστημα για κάθε αναφορά που γίνεται σε κάποια οντότητα.

Όταν όμως κάποιος χρήστης αναφέρεται σε κάποια οντότητα ή επιθυμεί να εισάγει κάποια οντότητα στη βάση δεδομένων χρησιμοποιεί διαφορετικά αναγνωριστικά για κάθε οντότητα. Τα αναγνωριστικά αυτά, τα οποία ονομάζονται *λογικά ονόματα*, είναι ουσιαστικά κάποια ονόματα (μια ή περισσότερες λέξεις) που περιγράφουν τις οντότητες αυτές και αντιστοιχούν στα πραγματικά αντικείμενα ή σχέσεις που οι οντότητες αυτές αναπαριστούν. Για παράδειγμα το λογικό όνομα που θα χρησιμοποιούσε ένας χρήστης για να αναγνωρίσει κάποια κλάση οντοτήτων που περιγράφει αυτοκίνητα θα ήταν η λέξη *αυτοκίνητο*, ενώ το σύστημα θα χρησιμοποιούσε κάποιο αναγνωριστικό συστήματος με τη μορφή που περιγράφηκε στο τέταρτο κεφάλαιο. Αντίστοιχα, για να περιγράψει ο χρήστης την οντότητα που αντιστοιχεί στον κινητήρα κάποιου αυτοκινήτου θα χρησιμοποιούσε τα λογικά ονόματα που αντιστοιχούν στις κλάσεις οντοτήτων *αυτοκίνητο* και *κινητήρας*. Το σύστημα της TELOS με τη σειρά του για να αναφερθεί στο κινητήρα κάποιου αυτοκινήτου πρέπει να αναφερθεί πρώτα στην οντότητα *αυτοκίνητο* μέσω του αναγνωριστικού συστήματος που της αντιστοιχεί και κατόπιν στην οντότητα *κινητήρας* μέσω του αντίστοιχου αναγνωριστικού συστήματος που περιέχεται στην περιγραφή της οντότητας *αυτοκίνητο*. Η αντιστοιχία στη χρήση των λογικών ονομάτων από το χρήστη και των αναγνωριστικών συστήματος από το σύστημα παρουσιάζεται στο σχήμα **??**.

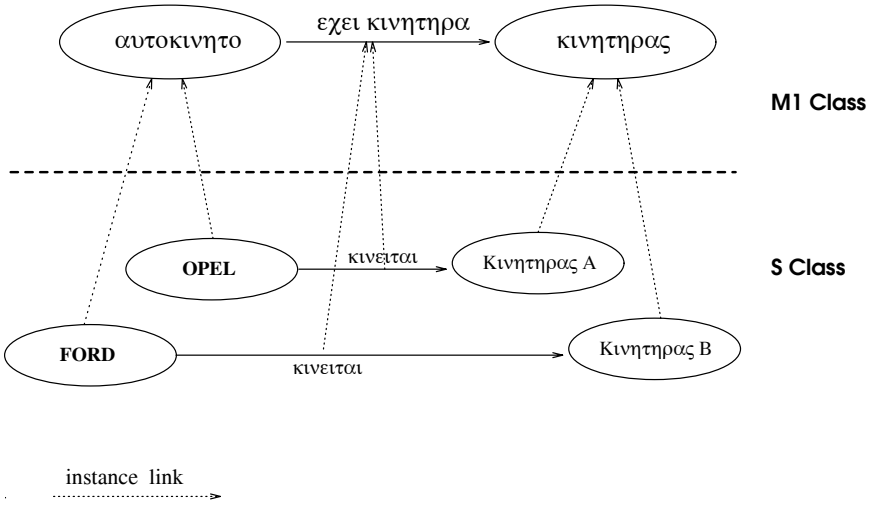
Ο μηχανισμός ταξινόμησης (instantiation) οδηγεί στην κληρονόμηση των γνωρισμάτων



Σχήμα 5.1: Αντιστοιχία *λογικών ονομάτων* και *αναγνωριστικών συστήματος*

Ο χρήστης αναφέρεται στις διάφορες οντότητες μέσω των λογικών ονομάτων *αυτοκίνητο*, *έχει κινητήρα* και *κινητήρας*, ενώ το σύστημα χρησιμοποιεί αντίστοιχα τα αναγνωριστικά οντοτήτων *a*, *b* και *c*.

κάποιας κλάσης οντοτήτων σε κατώτερα επίπεδα ταξινόμησης. Έτσι, τα γνωρίσματα που κληρονομούνται από μια κλάση οντοτήτων στις περιπτώσεις της, έχουν το ίδιο λογικό όνομα για όλες τις περιπτώσεις (εκτός αν ο χρήστης θελήσει να αλλάξει αυτό το λογικό όνομα). Για παράδειγμα, τα αυτοκίνητα *FORD* και *OPEL* στο σχήμα **??** χρησιμοποιούν το ίδιο λογικό όνομα *έχει κινητήρα* για το γνώρισμα που συνδέει τις συγκεκριμένες περιπτώσεις με αντίστοιχες περιπτώσεις της κλάσης οντοτήτων *κινητήρας*. Τα λογικά ονόματα για τις *ανεξάρτητες οντότητες* (individuals) είναι μοναδικά για όλη τη βάση δεδομένων, ενώ τα λογικά ονόματα για τα *γνωρίσματα* μπορούν να επαναλαμβάνονται. Σε μια τέτοια περίπτωση, τα γνωρίσματα διακρίνονται μέσω του μοναδικού αναγνωριστικού συστήματος που τους ανατίθεται, ή μέσω του λογικού ονόματος της οντότητας στην οποία ορίζονται μαζί με το λογικό όνομα για το συγκεκριμένο γνώρισμα. Στην περίπτωση αυτή, το λογικό όνομα της οντότητας (ή το αναγνωριστικό συστήματος της οντότητας, μια και είναι και τα δυο μοναδικά) δημιουργούν κάποιο μηχανισμό εμβέλειας λογικών ονομάτων. Έτσι, η εμβέλεια στην οποία ένα λογικό όνομα είναι μοναδικό συμπίπτει με το σύνολο των λογικών ονομάτων της βάσης δεδομένων, αν το λογικό όνομα αναφέρεται σε ανεξάρτητη οντότητα. Όταν ένα λογικό όνομα αναφέρεται σε γνώρισμα κάποιας ανεξάρτητης οντότητας, η εμβέλεια του λογικού ονόματος προσδιορίζεται από το σύνολο των λογικών ονομάτων για τα γνωρίσματα της συγκεκριμένης οντότητας. Με άλλα λόγια, κάθε γνώρισμα μιας οντότητας έχει διαφορετικό λογικό όνομα από όλα τα άλλα γνωρίσματα της συγκεκριμένης οντότητας.



Σχήμα 5.2: Εμβέλεια λογικών ονομάτων

Το αυτοκίνητο *FORD* και το αυτοκίνητο *OPEL* έχουν το ίδιο λογικό όνομα *κινείται* για τα γνωρίσματά τους, αλλά τα γνωρίσματα αυτά αποτελούν χωριστές οντότητες

Ο μηχανισμός για την εμβέλεια των λογικών ονομάτων επιτρέπει στο χρήστη τον αυστηρό καθορισμό για κάθε οντότητα που επιθυμεί να προσπελάσει στη βάση. Έτσι, οι ανεξάρτητες οντότητες καθορίζονται μέσω του μοναδικού τους λογικού ονόματος, ενώ τα γνωρίσματα ως συνδυασμός του λογικού ονόματος για την οντότητα που τα περιέχει μαζί με το συγκεκριμένο λογικό όνομα του συγκεκριμένου γνωρίσματος. Έτσι, στο σχήμα **??** ο συνδυασμός *FORDH κινητήρας* αναφέρεται στο γνώρισμα *κινητήρας* για την οντότητα *FORD*.

Όπως έχουμε αναφέρει, το σύστημα διαχείρισης οντοτήτων χρησιμοποιεί τα αναγνωριστικά συστήματος για να προσπελαύνει τις οντότητες της βάσης. Έτσι, χρειάζεται ένας μηχανισμός, ο οποίος μεταφράζει τα λογικά ονόματα των διαφόρων οντοτήτων στα αναγνωριστικά συστήματος που τους αντιστοιχούν. Καθώς ο χρήστης αντιλαμβάνεται τις οντότητες μέσω των λογικών ονομάτων τους, το σύστημα διαχείρισης οντοτήτων πρέπει να παρέχει έναν ακόμα μηχανισμό, ο οποίος μεταφράζει τα αναγνωριστικά συστήματος για τις διάφορες οντότητες στα λογικά ονόματα που τους αντιστοιχούν, ώστε να παρουσιάζει τα λογικά ονόματα των οντοτήτων στο χρήστη. Αντικείμενο του τρέχοντος κεφαλαίου αποτελεί ο μηχανισμός αποθήκευσης λογικών ονομάτων και μετάφρασης λογικών ονομάτων σε αναγνωριστικά συστήματος και αντίστροφα. Ο μηχανισμός αυτός ονομάζεται *κατάλογος λογικών ονομάτων* και συντίθεται από τον *υποκατάλογο αναγνωριστικών συστήματος* και τον *υποκατάλογο λογικών ονομάτων*.

5.2 Υποκατάλογος αναγνωριστικών συστήματος

Ο κατάλογος αυτός χρησιμοποιείται από τις εφαρμογές της TELOS προκειμένου να ανακληθεί το λογικό όνομα μιας οντότητας για την οποία είναι γνωστό το αναγνωριστικό συστήματος που την προσδιορίζει. Επειδή το εννοιολογικό μοντέλο για τη γλώσσα TELOS επιτρέπει τον ορισμό γνωρισμάτων σε γνωρίσματα, το λογικό όνομα ενός γνωρίσματος συντίθεται από τα λογικά ονόματα της ανεξάρτητης οντότητας στην οποία ορίζεται το γνώρισμα και των γνωρισμάτων που συνθέτουν το μονοπάτι στην ιεραρχία σύνθεσης μέσω του οποίου συνδέεται το συγκεκριμένο γνώρισμα με την ανεξάρτητη οντότητα. Έτσι, το λογικό όνομα για τα γνωρίσματα μπορεί να αποτελείται από τα λογικά ονόματα όλων των οντοτήτων που βρίσκονται στο συγκεκριμένο μονοπάτι της ιεραρχίας σύνθεσης. Ένα τέτοιο λογικό όνομα δε χρειάζεται να αποθηκεύεται καθώς μπορεί να δημιουργηθεί συνενώνοντας (χρησιμοποιώντας κατάλληλα σημεία στίξης για διαχωρισμό) τα λογικά ονόματα όλων των οντοτήτων που συμμετέχουν στο καθορισμό του. Ένα τέτοιο λογικό όνομα ονομάζεται *πλήρες λογικό όνομα γνωρίσματος*, μια και μόνο γνωρίσματα μπορούν να έχουν τέτοια λογικά ονόματα.

Ο καθορισμός του λογικού ονόματος για κάποια οντότητα της οποίας το αναγνωριστικό συστήματος είναι γνωστό είναι μια πολύ συχνή πράξη που επιτελείται πριν από οποιαδήποτε παρουσίαση κάποιας οντότητας στο χρήστη από τις διάφορες εφαρμογές. Ο μηχανισμός λοιπόν που πραγματοποιεί αυτή τη πράξη πρέπει να είναι πολύ αποδοτικός, επιτρέποντας γρήγορη ανάκληση των λογικών ονομάτων οντοτήτων ακόμα και για βάσεις με πολύ μεγάλο πλήθος οντοτήτων.

5.2.1 Αρχιτεκτονική του υποκαταλόγου αναγνωριστικών συστήματος

Ο μηχανισμός για τη διαχείριση των λογικών ονομάτων για τις οντότητες εκτός από τις διάφορες μεθόδους που επιτρέπουν τη προσπέλαση, εισαγωγή, διαγραφή ή αλλαγή των λογικών ονομάτων για κάθε οντότητα περιλαμβάνει και μια δομή αποθήκευσης, η οποία συσχετίζει τα λογικά ονόματα με τα αναγνωριστικά συστήματος και επιτρέπει την εφαρμογή των διαφόρων μεθόδων αναζήτησης του *υποκαταλόγου αναγνωριστικών συστήματος*.

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, τα αναγνωριστικά συστήματος παραμένουν μοναδικά καθ'όλη τη διάρκεια ζωής των οντοτήτων που προσδιορίζουν. Επίσης, η δομή των αναγνωριστικών συστήματος επιτρέπει την προσπέλαση των εγγραφών του καταλόγου συστήματος με ελάχιστο και σταθερό κόστος. Καθώς οι προσπελάσεις στον υποκατάλογο αναγνωριστικών συστήματος γίνονται μέσω των αναγνωριστικών συστήματος που προσδιορίζουν τα αντίστοιχα λογικά ονόματα για τις οντότητες, ο υποκατάλογος αναγνωριστικών

συστήματος μπορεί να οργανωθεί αντίστοιχα με τον κατάλογο συστήματος.

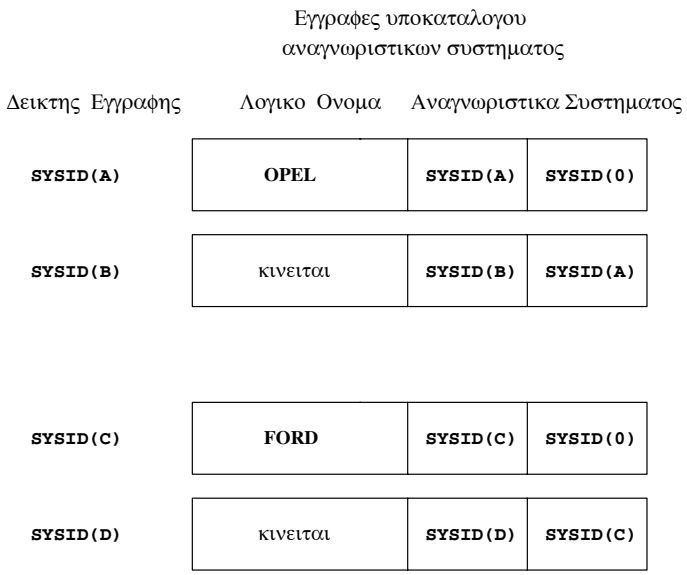
Έτσι, ο υποκατάλογος αναγνωριστικών συστήματος είναι ένας *πίνακας από εγγραφές* για κάθε οντότητα της βάσης. Κάθε εγγραφή σχετίζει ένα αναγνωριστικό συστήματος κάποιας οντότητας με το λογικό όνομα της οντότητας και περιγράφεται στη συνέχεια του κεφαλαίου. Φυσικά το μέγεθος του καταλόγου αυτού προσδιορίζεται από το μέγιστο επιτρεπτό πλήθος οντοτήτων σε μια βάση. Ο κατάλογος αυτός στη περίπτωση πολύ μεγάλων βάσεων δεν χωράει στη μνήμη του υπολογιστή και οργανώνεται σαν πίνακας από τμήματα (segment table). Ο βαθμός εμμεσότητας για τον πίνακα αυτό είναι 1 (single indirection) και κάθε τμήμα του πίνακα αυτού περιέχει καθορισμένο πλήθος εγγραφών. Κάθε τμήμα αποτελεί και μια εγγραφή για τη κρυφή μνήμη του υποκαταλόγου αναγνωριστικών συστήματος που περιγράφεται στο κεφάλαιο 6.

Τα αναγνωριστικά συστήματος αποτελούν τους δείκτες μέσω των οποίων γίνεται η προσπέλαση στον υποκατάλογο αναγνωριστικών συστήματος για κάθε αναφορά σε λογικά ονόματα. Καθώς ο κατάλογος αυτός οργανώνεται σαν έμμεσος πίνακας, τα αναγνωριστικά συστήματος θεωρούνται ως διανύσματα της μορφής <*δείκτης τμήματος, δείκτης εγγραφής*>, όπου ο *δείκτης τμήματος* καθορίζει το τμήμα στο οποίο βρίσκεται μια συγκεκριμένη εγγραφή, ενώ το πεδίο *δείκτης εγγραφής* καθορίζει τη θέση της εγγραφής αυτής μέσα στο τμήμα του υποκαταλόγου αναγνωριστικών συστήματος που την περιέχει. Αξίζει να αναφέρουμε στο σημείο αυτό ότι ενώ ο τρόπος δεικτοδότησης του υποκαταλόγου αναγνωριστικών συστήματος είναι ίδιος με τον τρόπο δεικτοδότησης του καταλόγου συστήματος όπως φαίνεται στο σχήμα **??**, το πλήθος των δυαδικών ψηφίων που καθορίζουν τους δείκτες τμήματος και εγγραφής διαφέρουν στους δυο καταλόγους καθώς το μέγεθος των εγγραφών για κάθε κατάλογο είναι διαφορετικό.

5.2.2 Εγγραφές του υποκαταλόγου αναγνωριστικών συστήματος

Οι εγγραφές του υποκαταλόγου αναγνωριστικών συστήματος χρησιμοποιούνται για την αντιστοίχιση του αναγνωριστικού συστήματος κάθε οντότητας στο λογικό όνομα που περιγράφει την οντότητα αυτή. Αν η εγγραφή αυτή αναφέρεται σε γνώρισμα, το *πλήρες λογικό όνομα γνωρίσματος* που αντιστοιχεί στο γνώρισμα δεν κρατιέται στην εγγραφή, αλλά δημιουργείται μέσω της ιεραρχίας σύνθεσης για το συγκεκριμένο γνώρισμα. Σε μια τέτοια περίπτωση, ο χρήστης μπορεί να παραλείψει την ανάθεση λογικού ονόματος στο γνώρισμα και τότε το λογικό όνομα για το γνώρισμα αυτό δημιουργείται αυτόματα από τον κατάλογο λογικών ονομάτων, συνθέτοντας το λογικό όνομα loginam με το αναγνωριστικό συστήματος που προσδιορίζει το συγκεκριμένο γνώρισμα.

Επιπλέον, οι εγγραφές του υποκαταλόγου αναγνωριστικών συστήματος περιλαμβάνουν



Σχήμα 5.3: Ένα παράδειγμα της χρήσης των εγγραφών για την εύρεση λογικού ονόματος για κάποια οντότητα της οποίας το αναγνωριστικό είναι γνωστό

Στο αριστερό τμήμα φαίνονται τα μοναδικά αναγνωριστικά συστήματος μέσω των οποίων γίνεται η προσπέλαση στις αντίστοιχες εγγραφές.
Η εγγραφή για την οντότητα με το λογικό όνομα **OPEL** περιέχει, εκτός από το λογικό όνομα και το αναγνωριστικό συστήματος, ένα άκυρο αναγνωριστικό συστήματος το **SYSID(0)** που δηλώνει ότι η οντότητα είναι ανεξάρτητη οντότητα. Για το γνώρισμα **κινείται** της οντότητας αυτής η εγγραφή περιέχει και το αναγνωριστικό συστήματος **SYSID(A)** που δηλώνει ότι η συγκεκριμένη οντότητα αποτελεί γνώρισμα της οντότητας με αναγνωριστικό **SYSID(A)**. Η εγγραφή για το αντίστοιχο γνώρισμα **κινείται** της οντότητας **FORD** περιέχει το μοναδικό αναγνωριστικό της οντότητας αυτής, δηλαδή το **SYSID(C)**.

άλλο ένα πεδίο με ένα αναγνωριστικό συστήματος που χρησιμοποιείται για τη δημιουργία του μηχανισμού εμβέλειας λογικών ονομάτων. Στην περίπτωση γνωρισμάτων, το αναγνωριστικό αυτό προσδιορίζει την ανεξάρτητη οντότητα της οποίας αποτελεί γνώρισμα η οντότητα στην οποία αντιστοιχεί η συγκεκριμένη εγγραφή. Το πεδίο αυτό χρησιμοποιείται για τη δημιουργία του *πλήρους λογικού ονόματος γνωρίσματος* όπως φαίνεται στο παράδειγμα του σχήματος **??**.

5.2.3 Διαχείριση εγγραφών

Σε αντίθεση με το μηχανισμό διαχείρισης του καταλόγου συστήματος, στον υποκατάλογο αναγνωριστικών συστήματος δεν διατηρούνται λίστες ελεύθερων εγγραφών ούτε περιοχή ελεύθερων εγγραφών, αλλά οι ελεύθερες εγγραφές περιέχουν μια άκυρη τιμή στο πεδίο για το αναγνωριστικό συστήματος για τη συγκεκριμένη εγγραφή. Η εισαγωγή ή διαγραφή μιας οντότητας στη βάση, γίνεται απ'ευθείας από τον κατάλογο συστήματος, ο οποίος με τη σειρά του χρησιμοποιεί την κατάλληλη μέθοδο του υποκαταλόγου αναγνω-

ριστικών συστήματος για την ανάθεση ή ελευθέρωση μιας εγγραφής στον υποκατάλογο αναγνωριστικών συστήματος αντίστοιχα. Η μέθοδος αυτή απλοποιεί τη διαχείριση του υποκαταλόγου αναγνωριστικών συστήματος καθώς απαλλάσει το σύστημα από διατήρηση επιπλέον πληροφορίας που είναι ήδη γνωστή.

5.2.4 Σύστημα αποθήκευσης

Ο υποκατάλογος αναγνωριστικών συστήματος καταλαμβάνει ένα δυαδικό αρχείο η διαχείριση του οποίου πραγματοποιείται μέσω κλήσεων στις ρουτίνες συστήματος για τη διαχείριση αρχείων που παρέχει το λειτουργικό σύστημα UNIX. Κάθε σελίδα δίσκου (disk page) καταλαμβάνεται από ένα τμήμα εγγραφών, καθώς το μέγεθος των τμημάτων αυτών είναι ίδιο με το μέγεθος της σελίδας δίσκου. Τα τμήματα εγγραφών βρίσκονται στο δίσκο ταξινομημένα ως προς αύξον αναγνωριστικό συστήματος για τις εγγραφές που περιέχουν, και έτσι η διεύθυνση κάποιου τμήματος εγγραφών υπολογίζεται άμεσα χρησιμοποιώντας τα πιο σημαντικά δυαδικά ψηφία του αναγνωριστικού συστήματος για οποιαδήποτε εγγραφή περιλαμβάνεται στο συγκεκριμένο τμήμα.

Για τις δοσοληψίες που αποτυγχάνουν χρησιμοποιείται η τεχνική no rollback και έτσι το αρχείο αποθήκευσης δεν μεταβάλλεται καθόλου σε τέτοιες περιπτώσεις, παρά μονάχα ακυρώνονται τα περιεχόμενα της κρυφής μνήμης για τον υποκατάλογο αναγνωριστικών συστήματος. Όταν μια δοσοληψία επιτυγχάνει, στο δίσκο γράφονται μονάχα τα νέα τμήματα εγγραφών και τα τμήματα εγγραφών που έχουν υποστεί αλλαγές κατά τη δοσοληψία.

5.3 Ο υποκατάλογος λογικών ονομάτων

Ο υποκατάλογος αναγνωριστικών συστήματος επιτρέπει την ανάκληση του λογικού ονόματος μιας οντότητας, όταν είναι γνωστό το αναγνωριστικό συστήματος που την καθορίζει, εξυπηρετώντας έτσι την παρουσίαση των λογικών ονομάτων στο χρήστη.

Ο χρήστης για τον καθορισμό μιας οντότητας, που προσπελαύνεται από το σύστημα, χρησιμοποιεί το λογικό όνομα της οντότητας. Όπως έχουμε αναφέρει, το λογικό όνομα για τις ανεξάρτητες οντότητες είναι μοναδικό στο σύνολο λογικών ονομάτων για τις οντότητες κάποιας βάσης, ενώ τα λογικά ονόματα για τα γνωρίσματα μπορούν να επαναλαμβάνονται.

Ο καθορισμός μιας οντότητας από το χρήστη μέσω του λογικού της ονόματος δεν είναι τόσο συχνή πράξη όσο ο καθορισμός της οντότητας μέσω του αναγνωριστικού της, καθώς οι λειτουργίες του συστήματος χρησιμοποιούν τα αναγνωριστικά των οντοτήτων για την πραγματοποίησή τους. Η γλώσσα εισαγωγής δεδομένων (Data Entry Language) [**?**] που χρησιμοποιείται για εισαγωγή, μεταβολή ή διαγραφή των διαφόρων οντοτήτων χρησιμοποιεί

αποκλειστικά λογικά ονόματα για τον καθορισμό των οντοτήτων. Έτσι, ο μηχανισμός μετάφρασης λογικών ονομάτων σε αναγνωριστικά συστήματος πρέπει να είναι αρκετά αποδοτικός, ώστε να μην επιβαρύνει το μηχανισμό εισαγωγής δεδομένων στη βάση.

Για τον σχεδιασμό και την υλοποίηση του μηχανισμού μετάφρασης λογικών ονομάτων στα αντίστοιχα αναγνωριστικά συστήματος, αξιολογήθηκαν διάφοροι μηχανισμοί κατακερματισμού (hashing) και δέντρων αναζήτησης και τελικά επιλέχθηκε ο μηχανισμός γραμμικού κατακερματισμού (linear hashing). Στη συνέχεια του κεφαλαίου γίνεται μια συνοπτική παρουσίαση των μηχανισμών που μελετήθηκαν και κατόπιν παρουσιάζεται ο μηχανισμός που επιλέχθηκε, καθώς και ο σχεδιασμός και υλοποίηση που πραγματοποιήθηκαν για τον *υποκατάλογο λογικών ονομάτων*.

5.3.1 Μηχανισμοί αναζήτησης

Καθώς η αναζήτηση των εγγραφών του υποκαταλόγου λογικών ονομάτων χρησιμοποιεί τα λογικά ονόματα για την εύρεση των εγγραφών, αντικείμενο αυτής της παραγράφου αποτελούν οι μηχανισμοί αναζήτησης και αποθήκευσης ονομάτων. Συνήθως οι μηχανισμοί αυτοί, που είναι οι διάφορες μορφές των δέντρων αναζήτησης και των καταλόγων κατακερματισμού (hash tables), υποθέτουν ότι τα κλειδιά αναζήτησης είναι μοναδικά. Η μοναδικότητα των κλειδιών αναζήτησης εξασφαλίζει ικανοποιητική απόδοση στους μηχανισμούς αυτούς. Στην περίπτωση των λογικών ονομάτων για την TELOS, τα κλειδιά αυτά δεν είναι μοναδικά.

Αν και οι μηχανισμοί αναζήτησης που προτείνονται από την βιβλιογραφία είναι πολλοί και έχουν μελετηθεί αρκετά ως προς την απόδοσή τους, δεν υπάρχει ένας μηχανισμός ο οποίος να έχει την καλύτερη απόδοση και να ταιριάζει καλύτερα σε κάθε εφαρμογή. Συνήθως κάθε μηχανισμός είναι καταλληλότερος από άλλους μονάχα για συγκεκριμένες εφαρμογές. Κάθε μηχανισμός αναζήτησης αξιολογείται σύμφωνα με τα παρακάτω κριτήρια, όσον αφορά τις διάφορες λειτουργίες του και το χώρο που απαιτεί για την αναπαράσταση των δεδομένων που περιέχει:

Χρόνος προσπέλασης: Ο χρόνος που απαιτείται για να βρεθεί μια συγκεκριμένη μονάδα πληροφορίας χρησιμοποιώντας το συγκεκριμένο μηχανισμό. Στην περίπτωση του καταλόγου λογικών ονομάτων, η αναζήτηση ισοδυναμεί με την εύρεση μιας εγγραφής που περιέχει ένα συγκεκριμένο λογικό όνομα μιας οντότητας, σε κάποια δομή που περιλαμβάνει εγγραφές αντιστοίχισης λογικών ονομάτων και αναγνωριστικών συστήματος.

Χρόνος εισαγωγής: Ο χρόνος που απαιτείται για την εισαγωγή μιας νέας εγγραφής αντιστοίχισης λογικών ονομάτων και αναγνωριστικών συστήματος στη δομή αναζήτησης.

Κατά τη λειτουργία αυτή, ο συνολικός χρόνος περιλαμβάνει το χρόνο για την εύρεση της θέσης όπου θα εισαχθεί η νέα εγγραφή, καθώς επίσης και το χρόνο που απαιτείται για την ενημέρωση της δομής αναζήτησης ώστε να περιλαμβάνει τη νέα εγγραφή.

Χρόνος διαγραφής: Ο χρόνος που απαιτείται για τη διαγραφή μιας συγκεκριμένης εγγραφής από τη δομή αναζήτησης. Κατά τη λειτουργία αυτή, ο συνολικός χρόνος περιλαμβάνει το *χρόνο προσπέλασης* για τη συγκεκριμένη εγγραφή, καθώς επίσης και το χρόνο που απαιτείται για την ενημέρωση των δομών αναζήτησης ώστε να μην περιλαμβάνουν πλέον την εγγραφή που διαγράφτηκε.

Απαιτούμενος χώρος: Ο χώρος που χρειάζεται για την αποθήκευση των δομών δεδομένων για το συγκεκριμένο μηχανισμό αναζήτησης. Ο χώρος που απαιτείται για την αποθήκευση των εγγραφών αντιστοίχισης λογικών ονομάτων και αναγνωριστικών συστήματος είναι ανεξάρτητος από το μηχανισμό αναζήτησης. Η οργάνωση όμως της δομής αναζήτησης χρειάζεται επιπλέον χώρο για άλλες δομές που χρησιμοποιούνται για την υλοποίηση των διαφόρων λειτουργιών. Το μέγεθος του επιπλέον χώρου που απαιτείται συνήθως δεν είναι σημαντικό, οπότε πολλές φορές είναι προτιμότερη η επιλογή ενός μηχανισμού που καταναλώνει περισσότερο χώρο, αλλά έχει καλύτερη χρονική απόδοση από άλλους. Ένα μέγεθος που δείχνει την αποδοτικότητα ως προς τον απαιτούμενο χώρο είναι ο *βαθμός χρήσης αποθηκευτικού χώρου* (storage utilization) που ορίζεται ως ο λόγος του πλήθους εγγραφών που πραγματικά χρησιμοποιούνται προς το πλήθος των εγγραφών που αποθηκεύονται, καθώς οι διάφοροι μηχανισμοί αναζήτησης χρησιμοποιούν *ομάδες εγγραφών* (buckets) ως τη μονάδα ανάθεσης αποθηκευτικού χώρου. Δηλαδή, κάθε φορά που χρειάζεται μια εγγραφή, οι μηχανισμοί αυτοί είναι δυνατόν να δημιουργούν μια ολόκληρη ομάδα εγγραφών για την αποθήκευσή της, ενώ κάποιες από τις ομάδες εγγραφών που χρησιμοποιούνται περιέχουν ήδη ελεύθερες εγγραφές.

Β-δέντρα αναζήτησης

Οι δομές Β-δέντρων αναζήτησης [**?**] προσφέρουν ένα αποτελεσματικό μηχανισμό για οργάνωση και διατήρηση δομών αναζήτησης. Επειδή όμως η απόδοση του μηχανισμού αυτού όταν συμβαίνουν εισαγωγές και διαγραφές δεν είναι ικανοποιητική, έχουν προταθεί στη βιβλιογραφία διάφορες παραλλαγές των Β-δέντρων με ικανοποιητική απόδοση. Ο περιορισμός της απόδοσης των Β-δέντρων προκύπτει από την οργάνωση των κόμβων τους. Έτσι, κάθε κόμβος περιέχει εγγραφές με δεδομένα και εγγραφές με δείκτες στους άμεσους απόγονούς του. Επίσης, το μέγεθος κάθε κόμβου είναι σταθερό και έτσι ο κόμβος μπορεί

να περιέχει μέχρι ένα μέγιστο αριθμό εγγραφών δεδομένων και δεικτών. Ο αριθμός αυτός ονομάζεται *χωρητικότητα* του B-δέντρου και συμβολίζεται με το σύμβολο k. Κάθε κόμβος μπορεί να περιέχει μέχρι k εγγραφές δεδομένων και μέχρι $k+1$ δείκτες σε άλλους κόμβους. Επίσης, όλοι οι κόμβοι του B-δέντρου, εκτός από τη ρίζα, περιέχουν τουλάχιστον $\lfloor \frac{k}{2} \rfloor$ εγγραφές με δεδομένα.

Το ύψος h ενός B-δέντρου που περιέχει N εγγραφές με δεδομένα είναι:

$$log_{k+1}(N+1) \leq h \leq log_{\lfloor \frac{k}{2} \rfloor + 1}(\frac{N+1}{2})$$

Επειδή κάθε κόμβος του B-δέντρου περιέχει εγγραφές με δεδομένα, και κάθε πράξη στα B-δέντρα χρησιμοποιεί την πράξη της αναζήτησης για την πραγματοποίησή της, όλες οι πράξεις εξαρτώνται από το ύψος του δέντρου. Προκειμένου λοιπόν να παρέχει ικανοποιητική απόδοση για μεγάλο πλήθος εγγραφών η δομή B-δέντρου, πρέπει το ύψος του δέντρου να παραμένει μικρό όπως παρατηρείται στο [**?**].

Μια λύση είναι η αύξηση της χωρητικότητας k των κόμβων του δέντρου. Η λύση αυτή όμως δεν είναι ικανοποιητική καθώς αύξηση του μεγέθους των κόμβων προκαλεί αύξηση του κόστους επεξεργασίας για τον κάθε κόμβο αφενός, και στο κόστος αποθήκευσης και ανάκλησης κάθε κόμβου καθώς οι συσκευές αποθήκευσης παρέχουν δυνατότητες προσπέλασης συγκεκριμένων ποσοτήτων αποθηκευμένης πληροφορίας σε κάθε πρόσβαση σ'αυτές.

Η αποτελεσματική λύση που προτείνεται είναι τα B^+-δέντρα [**?**]. Στα B^+-δέντρα οι εγγραφές πληροφορίας βρίσκονται μονάχα στα φύλλα, ενώ οι υπόλοιποι κόμβοι των δέντρων περιέχουν δείκτες στους απογόνους τους και τιμές δεικτοδότησης που επιτρέπουν τη σωστή επιλογή του κατάλληλου απογόνου σε κάθε προσπέλαση στο δέντρο. Η προσπέλαση για κάθε εγγραφή με δεδομένα απαιτεί μια προσπέλαση στο δίσκο αν ολόκληρο το υποδέντρο με τους δείκτες βρίσκεται στην μνήμη. Η εισαγωγή ή διαγραφή εγγραφών όμως στο δέντρο μπορεί να οδηγεί σε επαναδιατάξεις των φύλλων όταν αυτά υπερχειλίζουν (overflow) ή υποχειλίζουν (underflow), δημιουργώντας έτσι διακυμάνσεις στην απόδοση αλλά και στο βαθμό χρησιμοποίησης αποθηκευτικού χώρου. Ένας κόμβος υπερχειλίζει όταν πρέπει να εισαχθεί σε αυτόν κάποια εγγραφή και ο κόμβος περιέχει ήδη k εγγραφές και υποχειλίζει όταν ο κόμβος περιέχει το ελάχιστο δυνατόν πλήθος εγγραφών (δηλαδή $\lfloor \frac{k}{2} \rfloor$) και κάποια από τις εγγραφές του διαγράφεται.

Στην εργασία [**?**] παρουσιάζεται μια αποδοτικότερη μορφή B^+-δέντρων. Στα δέντρα αυτά που ονομάζονται B^+-δέντρα με μερικές επεκτάσεις (B^+-trees with partial expansions), τα φύλλα μπορούν να μεγαλώνουν σε μέγεθος όταν υπερχειλίζουν αντί να αναδιατάσσονται τα περιεχόμενά τους σε άλλα φύλλα. Το κόστος των πράξεων σε τέτοιες δομές είναι ίδιο με το κόστος για τα απλά B^+-δέντρα, αλλά ο βαθμός χρησιμοποίησης αποθηκευτικού χώρου

είναι υψηλότερος πλησιάζοντας την τιμή 81 %. Το μόνο μειονέκτημα του μηχανισμού αυτού είναι ότι απαιτεί πολύπλοκους αλγορίθμους για τη διαχείριση του αποθηκευτικού χώρου.

Μηχανισμοί κατακερματισμού (hash mechanisms)

Οι μηχανισμοί κατακερματισμού αποτελούν μια αποτελεσματική και δημοφιλή μέθοδο για οργάνωση δομών αναζήτησης. Στους μηχανισμούς αυτούς ο αποθηκευτικός χώρος μοιράζεται σε ένα πλήθος από *ομάδες εγγραφών* (buckets) σταθερού μεγέθους (όπως και τα φύλλα των B^+-δέντρων) και σε κάθε ομάδα εγγραφών ανατίθεται μια μοναδική *διεύθυνση*, μέσω της οποίας γίνεται κάθε προσπέλαση στην ομάδα αυτή. Χρησιμοποιώντας ένα μετασχηματισμό (εφαρμόζοντας δηλαδή κάποια συνάρτηση) στο κλειδί αναζήτησης, δημιουργείται η *διεύθυνση* της ομάδας εγγραφών όπου περιέχεται η εγγραφή με το συγκεκριμένο κλειδί αναζήτησης. Για παράδειγμα, προκειμένου να βρεθεί η εγγραφή που περιέχει την αντιστοιχία κάποιου λογικού ονόματος και αναγνωριστικού συστήματος πραγματοποιείται ένας μετασχηματισμός στο λογικό όνομα, καθορίζοντας έτσι την ομάδα εγγραφών όπου βρίσκεται η ζητούμενη εγγραφή. Για την αναζήτηση της εγγραφής μέσα στην ομάδα εγγραφών που την περιέχει χρησιμοποιείται ξανά η τιμή της συνάρτησης μετασχηματισμού. Οι μηχανισμοί κατακερματισμού είναι οι αποδοτικότεροι μηχανισμοί αναζήτησης μια και έχουν πολυπλοκότητα ανάκλησης δεδομένων της τάξεως του O(1) σε σύγκριση με τους μηχανισμούς δέντρων αναζήτησης όπου η πολυπλοκότητα είναι O($logN$) όπου N είναι το πλήθος των εγγραφών που αποθηκεύουν.

Οι μηχανισμοί κατακερματισμού χωρίζονται σε στατικούς και δυναμικούς. Η πρώτη κατηγορία χρησιμοποιείται όταν το σύνολο των κλειδιών αναζήτησης παραμένει σταθερό, ενώ η δεύτερη κατηγορία χρησιμοποιείται για εφαρμογές όπου το πλήθος των κλειδιών αναζήτησης μεταβάλλεται δυναμικά. Οι περισσότεροι μηχανισμοί δυναμικού κατακερματισμού χρησιμοποιούν κάποια δομή δεικτοδότησης, μέσω της οποίας γίνεται η επιλογή της κατάλληλης ομάδας εγγραφών. Στην κατηγορία αυτή ανήκουν μηχανισμοί όπως το composite perfect hashing [**?**], bounded disorder file organization [**?**] και virtual hashing [**?**]. Οι μηχανισμοί της κατηγορίας αυτής μπορούν να παρέχουν ανάκληση εγγραφών μέσω μονάχα μιας προσπέλασης στο δίσκο, αλλά ο μέσος χρόνος ανάκλησης επηρεάζεται από την ύπαρξη της δομής δεικτοδότησης. Επίσης, όπως παρατηρείται στο [**?**] η διαδικασία κατά την οποία μεγαλώνει ο κατάλογος κατακερματισμού όταν υπάρχουν συχνές εισαγωγές εγγραφών, είναι σχετικά αργή γιατί συνήθως περιλαμβάνει διπλασιασμό του καταλόγου κατακερματισμού. Επίσης, το μέγεθος των δομών δεικτοδότησης εξαρτάται από το πλήθος των εγγραφών που αποθηκεύονται αυξάνοντας το χρόνο της διάσχισης για πολύ μεγάλα σύνολα εγγραφών.

Η μέθοδος του *γραμμικού κατακερματισμού* (linear hashing) που αναπτύχθηκε από τον

W. Litwin [**?**] και αναλύεται στο [**?**] δεν χρησιμοποιεί δομές δεικτοδότησης και μπορεί να λειτουργήσει αποδοτικά για πολύ μεγάλα σύνολα εγγραφών, παρέχοντας ταυτόχρονα υψηλό *βαθμό χρήσης αποθηκευτικού χώρου.*

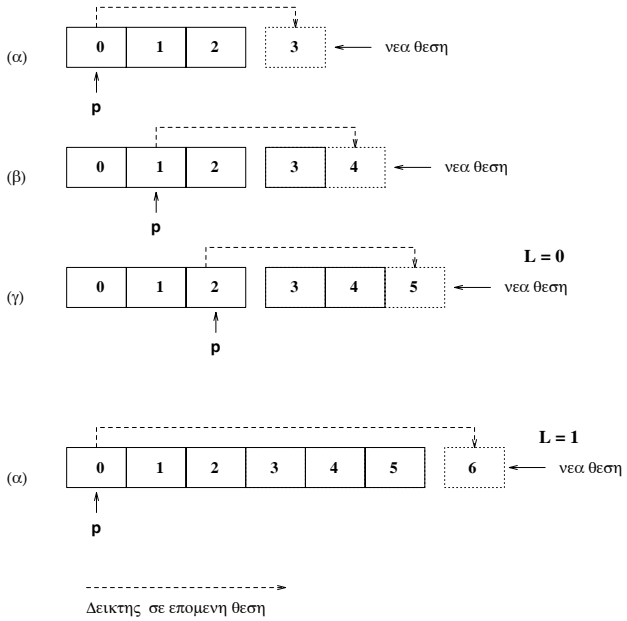
5.4 Οργάνωση του υποκαταλόγου λογικών ονομάτων

Η μέθοδος αναζήτησης που επιλέχθηκε για την οργάνωση του υποκαταλόγου λογικών ονομάτων είναι ο μηχανισμός γραμμικού κατακερματισμού, καθώς παρέχει ικανοποιητική απόδοση για τις διάφορες πράξεις και είναι αποτελεσματικός ως προς τη χρησιμοποίηση του αποθηκευτικού χώρου.

Στη συνέχεια, αφού παρουσιάσουμε το μηχανισμό γραμμικού κατακερματισμού, αναλύουμε τον τρόπο που εφαρμόζεται στον υποκατάλογο λογικών ονομάτων. Επειδή τα λογικά ονόματα που αποτελούν κλειδιά αναζήτησης για τον κατάλογο αυτό δεν είναι μοναδικά, χρειάστηκε να σχεδιαστεί και να υλοποιηθεί ένας ειδικός μηχανισμός αναζήτησης για εγγραφές που έχουν κοινό κλειδί αναζήτησης.

5.4.1 Γραμμικός κατακερματισμός

Ο μηχανισμός γραμμικού κατακερματισμού αναπτύχθηκε από τον W. Litwin το 1980 προκειμένου να χρησιμοποιηθεί για οργάνωση εξωτερικών αρχείων στα οποία χρησιμοποιούνται κλειδιά αναζήτησης. Ας θεωρήσουμε ένα κατάλογο κατακερματισμού (hash table) που αποτελείται από N ομάδες εγγραφών (buckets) με εικονικές διευθύνσεις 0, 1, ..., $N-1$. Ο μηχανισμός γραμμικού κατακερματισμού αυξάνει το μέγεθος του καταλόγου κατακερματισμού διασπώντας τις ομάδες εγγραφών σε μια προκαθορισμένη σειρά. Έτσι, πρώτα διασπάται η ομάδα εγγραφών με διεύθυνση 0 σε δύο νέες ομάδες εγγραφών με διευθύνσεις 0 και N, και κατόπιν όλες οι υπόλοιπες κατά αύξουσα σειρά εικονικής διεύθυνσης μέχρι και την ομάδα εγγραφών $N-1$. Κατά τη διάσπαση μιας ομάδας εγγραφών συνήθως μεταφέρονται περίπου οι μισές εγγραφές που περιέχει σε μια νέα ομάδα εγγραφών. Ένας δείκτης p χρησιμοποιείται για να δείχνει την επόμενη ομάδα εγγραφών που πρόκειται να διασπαστεί. Όταν οι N ομάδες εγγραφών έχουν διασπαστεί, το μέγεθος του καταλόγου κατακερματισμού διπλασιάζεται σε $2N$. Τότε ο *δείκτης επόμενης ομάδας εγγραφών προς διάσπαση* p γίνεται 0 και ξαναρχίζει η διαδικασία διάσπασης ομάδων εγγραφών από την ομάδα εγγραφών με διεύθυνση 0. Τη φορά αυτή η διαδικασία διάσπασης εφαρμόζεται στις ομάδες εγγραφών 0 έως $2N-1$, και όταν ολοκληρωθεί το μέγεθος του καταλόγου κατακερματισμού έχει γίνει $4N$. Στο σχήμα **??** παρουσιάζεται η ακολουθία διάσπασης των ομάδων εγγραφών για $N=3$ και $p=0$.



Σχήμα 5.4: Διαδικασία αύξησης του καταλόγου για γραμμικό κατακερματισμό

Στον κατάλογο αυτό αρχικά είναι $N = 3$, $p = 0$ και $L = 0$ αφού δεν έχει γίνει καμία πλήρης αύξηση του καταλόγου. Στις φάσεις (α), (β) και (γ) επιτελείται η πλήρης αύξηση και στην επόμενη φάση (α), έχει γίνει $L = 1$ και το μέγεθος του καταλόγου πλέον είναι N $\times$ $2^1 = 6$

Ο δείκτης νέας θέσης δείχνει στην θέση που δημιουργείται στον κατάλογο μόλις διασπαστεί η θέση p.

Η συνάρτηση κατακερματισμού η οποία υπολογίζει σε ποια ομάδα βρίσκεται κάποιο κλειδί αναζήτησης K όταν το μέγεθος του καταλόγου κατακερματισμού είναι N έχει τη μορφή:

$$h_0(K) = K mod N$$

ενώ όταν έχει διπλασιαστεί το μέγεθος του καταλόγου κατακερματισμού, η συνάρτηση έχει τη μορφή:

$$h_1(K) = K mod (2 \times N)$$

Όταν όμως το μέγεθος του καταλόγου κατακερματισμού δεν έχει διπλασιαστεί, ο υπολογισμός της ομάδας εγγραφών όπου βρίσκεται κάποιο κλειδί, γίνεται ως εξής: Υπολογίζουμε την τιμή $h_0(K)$ για το κλειδί αναζήτησης K. Αν η τιμή $h_0(K)$ είναι μικρότερη από την τρέχουσα τιμή του p, αυτό σημαίνει ότι η ομάδα εγγραφών που περιέχει την εγγραφή για το συγκεκριμένο κλειδί, έχει ήδη διασπαστεί. Στην περίπτωση αυτή, η εγγραφή βρίσκεται σε μια από τις ομάδες εγγραφών $h_0(K)$ ή $h_0(K) + N$. Η ομάδα εγγραφών τότε καθορίζεται από τη συνάρτηση $h_1(K)$.

Η γενική συνάρτηση για τον υπολογισμό της ομάδας εγγραφών που περιέχει την εγγραφή

με κλειδί αναζήτησης K είναι η παρακάτω:

$$h_j(K) = g(K) mod(N \times 2^j), j = 0, 1, \ldots$$

όπου N είναι το ελάχιστο μέγεθος του καταλόγου κατακερματισμού και το j περιγράφει πόσες φορές έχει διπλασιαστεί πλήρως ο κατάλογος κατακερματισμού. Η συνάρτηση $g(K)$, που ονομάζεται συνάρτηση μετασχηματισμού, είναι μια συνάρτηση που μετασχηματίζει το κλειδί K σε μια διεύθυνση που βρίσκεται στο διάστημα [0, M] όπου το M είναι αρκετά μεγάλο (συνήθως M > 2^{20}).

Η κατάσταση του καταλόγου κατακερματισμού μπορεί να περιγραφεί πλήρως με τις εξής παραμέτρους:

L : πόσες φορές έχει διπλασιαστεί το μέγεθος του καταλόγου κατακερματισμού (αρχικά έχει μέγεθος N).

p : δείκτης στην επόμενη ομάδα εγγραφών που πρόκειται να διασπαστεί. Για το p ισχύει η σχέση: $0 \leq p \leq N \times 2^L$.

Ο κατάλογος κατακερματισμού μπορεί να μεγαλώνει ή να μικραίνει δυναμικά. Η διάσπαση μιας ομάδας εγγραφών ή η συνένωση δύο ομάδων εγγραφών, καθορίζεται από το *βαθμό πληρότητας* (load factor) για τις ομάδες εγγραφών. Ο βαθμός πληρότητας καθορίζεται από τη σχέση:

$$\text{βαθμός πληρότητας} = \frac{\text{σύνολο εγγραφών στο κατάλογο κατακερματισμού}}{\text{πλήθος ομάδων εγγραφών} \times \text{χωρητικότητα ομάδας εγγραφών}}$$

όπου:

χωρητικότητα ομάδας εγγραφών είναι το πλήθος των εγγραφών που μπορούν να υπάρξουν σε μια ομάδα εγγραφών.

πλήθος ομάδων εγγραφών $= 2^L \times N + (p - 1)$ δηλαδή το μέγεθος του καταλόγου κατακερματισμού.

Ρυθμίζοντας μέγιστα και ελάχιστα όρια για το βαθμό πληρότητας μπορούμε να καθορίσουμε πότε αυξάνεται το μέγεθος του καταλόγου κατακερματισμού. Έτσι, το μέγεθος του καταλόγου κατακερματισμού αυξάνεται κάθε φορά που ο βαθμός πληρότητας υπερβαίνει το μέγιστο όριο και μειώνεται κάθε φορά που ο βαθμός πληρότητας γίνεται μικρότερος από το ελάχιστο όριο. Ο βαθμός πληρότητας αντιστοιχεί στο βαθμό χρήσης αποθηκευτικού χώρου που ορίσαμε για τους μηχανισμούς αναζήτησης.

Επειδή η μεταβολή του μεγέθους του πίνακα κατακερματισμού εξαρτάται από το βαθμό πληρότητας και όχι από το αν κάποια ομάδα εγγραφών υπερχειλίζει ή δεν περιέχει καθόλου

εγγραφές, είναι δυνατό σε κάποια απο τις θέσεις του καταλόγου κατακερματισμού να υπάρχουν περισσότερες εγγραφές από όσες μπορεί να περιλαμβάνει μια ομάδα εγγραφών. Στην περίπτωση αυτή, η ομάδα εγγραφών υπερχειλίζει και χρησιμοποιείται κάποιος μηχανισμός σύνδεσης ομάδων για τις ομάδες εγγραφών που αντιστοιχούν στη συγκεκριμένη θέση του καταλόγου κατακερματισμού. Στις περιπτώσεις αυτές, μπορεί να χρειάζονται περισσότερες από μια προσπελάσεις στο δίσκο για την εύρεση της εγγραφής που αντιστοιχεί σε ένα κλειδί αναζήτησης που βρίσκεται στη συγκεκριμένη θέση του καταλόγου κατακερματισμού. Για όλα τα άλλα κλειδιά αρκεί μια προσπέλαση στο δίσκο. Ο βαθμός πληρότητας συνήθως κυμαίνεται από 67 % έως 85 %.

5.4.2 Δομή του καταλόγου κατακερματισμού

Ο κατάλογος κατακερματισμού για τα λογικά ονόματα οργανώνεται με τη μορφή πίνακα από τμήματα (segment table). Κάθε τμήμα του καταλόγου αυτού αντιστοιχεί σε συγκεκριμένο πλήθος από ομάδες εγγραφών, ενώ κάθε ομάδα εγγραφών αντιστοιχεί σε ακριβώς μια θέση του καταλόγου κατακερματισμού.

Η επιλογή του μεγέθους των τμημάτων έγινε ώστε να συμπίπτει με τη μονάδα μεταφοράς πληροφορίας μεταξύ δίσκου και μνήμης υπολογιστή. Επίσης, το αρχικό μέγεθος του καταλόγου κατακερματισμού έχει την τιμή N = 8. Επειδή ο αριθμός 8 είναι ακέραια δύναμη του 2, όλα τα δυνατά μεγέθη του καταλόγου κατακερματισμού μετά από κάθε διπλασιασμό του είναι ακέραιες δυνάμεις του 2. Έτσι, η συνάρτηση υπολογισμού της ομάδας εγγραφών όπου βρίσκεται κάποιο κλειδί K, $h_j(K)$ υλοποιείται πολύ αποδοτικά παίρνοντας τα $3 + j$ λιγότερο σημαντικά δυαδικά ψηφία της τιμής της συνάρτησης μετασχηματισμού $g(K)$. Καθώς $N = 8 = 2^3$ η συνάρτηση επιλογής ομάδας εγγραφών παίρνει τη μορφή $h_j(K) = g(K) mod (8 \times 2^j) = g(K) mod (2^3 \times 2^j) = g(K) mod (2^{3+j})$ όπου το j είναι η τιμή της παραμέτρου L.

Το κόστος της συνάρτησης μετασχηματισμού $g(K)$ δεν μπορεί να αποφευχθεί με κανέναν από τους άλλους μηχανισμούς αναζήτησης (εκτός τα απλά B-δέντρα που δεν παρέχουν τόσο καλή απόδοση) μια και όλοι χρησιμοποιούν κάποιο μηχανισμό μετασχηματισμού του κλειδιού K σε κάποια τιμή που χρησιμοποιείται για τη διάσχιση των διαφόρων μορφών δεικτοδότησης. Η συνάρτηση μετασχηματισμού παρουσιάζεται στο παράρτημα **??**.

Η προσπέλαση στις εγγραφές οντοτήτων γίνεται με τρόπο αντίστοιχο με την προσπέλαση στις πλειάδες του υποκαταλόγου αναγνωριστικών συστήματος. Τα πιο σημαντικά δυαδικά ψηφία της τιμής $h_j(K)$ επιλέγουν το τμήμα στο οποίο βρίσκεται η ζητούμενη ομάδα εγγραφών, ενώ τα λιγότερο σημαντικά ψηφία επιλέγουν την ομάδα εγγραφών στο τμήμα.

Τα τμήματα του καταλόγου κατακερματισμού προσπελαύνονται απευθείας μέσω μηχα-

νισμού κρυφής μνήμης και έτσι δεν χρειάζεται να διατηρείται πίνακας δεικτών στα τμήματα αυτά.

5.4.3 Ομάδες εγγραφών

Οι ομάδες εγγραφών αποτελούν τις δομές αποθήκευσης των εγγραφών που αντιστοιχούν λογικά ονόματα σε αναγνωριστικά συστήματος. Κάθε ομάδα εγγραφών μπορεί να περιέχει μέχρι ένα καθορισμένο πλήθος από εγγραφές, δηλαδή οι ομάδες εγγραφών έχουν σταθερό μέγεθος. Επειδή το πλήθος των εγγραφών που βρίσκονται σε μια συγκεκριμένη θέση του καταλόγου κατακερματισμού μπορεί να είναι μεγαλύτερο από το πλήθος εγγραφών που περιέχει μια ομάδα εγγραφών, οι ομάδες εγγραφών μπορούν να συνδέονται σχηματίζοντας λίστες από ομάδες εγγραφών. Τότε η συγκεκριμένη θέση του καταλόγου κατακερματισμού περιέχει την πρωτεύουσα ομάδα εγγραφών (primary bucket), η οποία και αποτελεί την αρχή της λίστας ομάδων εγγραφών για τη θέση αυτή.

Κάθε ομάδα εγγραφών εκτός από τον πίνακα εγγραφών περιέχει μια επικεφαλίδα που αναφέρει πόσες εγγραφές βρίσκονται αποθηκευμένες σ'αυτή, και ένα δείκτη σε επόμενη ομάδα εγγραφών. Ο δείκτης αυτός χρησιμοποιείται για τη δημιουργία των λιστών ομάδων εγγραφών. Οι επιπλέον ομάδες εγγραφών (εκτός από την πρωτεύουσα) διαχειρίζονται από το *μηχανισμό βοηθητικών ομάδων εγγραφών*.

Κάθε εγγραφή περιλαμβάνει δύο πεδία. Το πρώτο περιέχει την τιμή $g(K)$ για κάποιο συγκεκριμένο λογικό όνομα K και το δεύτερο περιέχει το αναγνωριστικό συστήματος για την οντότητα η οποία χαρακτηρίζεται από το λογικό όνομα K. Προτιμήθηκε να περιέχουν οι εγγραφές τις τιμές της συνάρτησης $g(K)$ αντί των λογικών ονομάτων επειδή οι τιμές της συνάρτησης είναι ακέραιοι αριθμοί και έτσι έχουν πολύ μικρότερο μέγεθος από τα λογικά ονόματα που αποτελούνται από πολλούς (έως 96) χαρακτήρες.

Οι εγγραφές μιας ομάδας ταξινομούνται ως προς την τιμή του πεδίου που περιέχει το αποτέλεσμα της συνάρτησης $g()$. Έτσι, η εύρεση μιας συγκεκριμένης εγγραφής γίνεται μέσω δυαδικής αναζήτησης (binary search) στον πίνακα εγγραφών της ομάδας.

5.4.4 Μη μοναδικά ονόματα

Η λειτουργία των συμβατικών μορφών πινάκων κατακερματισμού προϋποθέτει ότι τα κλειδιά αναζήτησης και οι τιμές της συνάρτησης μετασχηματισμού $g()$ είναι μοναδικές. Στην περίπτωση του υποκαταλόγου ονομάτων, όπου κλειδί αναζήτησης είναι το λογικό όνομα, τα λογικά ονόματα για τα γνωρίσματα των οντοτήτων δεν είναι πάντα μοναδικά, δίνοντας έτσι ίδια τιμή στη συνάρτηση μετασχηματισμού. Επίσης, επειδή το σύνολο των πιθανών λογικών ονομάτων είναι πολύ μεγαλύτερο από το σύνολο των πιθανών τιμών της συνάρτησης

μετασχηματισμού, είναι δυνατόν οι τιμές της συνάρτησης αυτής για δυο διαφορετικά λογικά ονόματα να συμπίπτουν.

Έτσι, όταν σε περισσότερες από μία οντότητες (με ίδια ή διαφορετικά λογικά ονόματα) αντιστοιχεί ίδια τιμή της συνάρτησης μετασχηματισμού πρέπει να υπάρχει κάποιος τρόπος ώστε να αποθηκεύονται διαφορετικές εγγραφές για κάθε μία από τις οντότητες αυτές.

Στη περίπτωση αυτή, δημιουργείται μια εγγραφή η οποία περιέχει την τιμή της συνάρτησης μετασχηματισμού στο ένα της πεδίο και ένα δείκτη σε βοηθητική ομάδα εγγραφών. Η βοηθητική ομάδα εγγραφών έχει διαφορετική μορφή για τις εγγραφές της. Έτσι, οι εγγραφές πλέον δεν περιέχουν την τιμή της συνάρτησης μετασχηματισμού στο ένα τους πεδίο, αλλά το αναγνωριστικό συστήματος για την οντότητα της οποίας αποτελούν γνωρίσματα οι οντότητες με κοινό λογικό όνομα (και άρα κοινή τιμή της συνάρτησης μετασχηματισμού). Αν οι οντότητες έχουν κοινή τιμή για τη συνάρτηση μετασχηματισμού και δεν είναι γνωρίσματα, τότε χρησιμοποιείται μια άκυρη τιμή για αυτό το πεδίο.

Μέθοδοι αναζήτησης

Ο υποκατάλογος λογικών ονομάτων παρέχει δυο μεθόδους αναζήτησης για οντότητες των οποίων το λογικό όνομα είναι γνωστό. Οι μέθοδοι αυτές είναι: αναζήτηση μέσω του λογικού ονόματος και αναζήτηση μέσω του λογικού ονόματος για κάποιο γνώρισμα και του αναγνωριστικού συστήματος για την οντότητα στην οποία ορίζεται το γνώρισμα. Για κάθε μια από αυτές τις μεθόδους αναζήτησης, το κλειδί μέσω του οποίου υπολογίζεται η τιμή της *συνάρτησης μετασχηματισμού* είναι ο μετασχηματισμός $g()$ του λογικού ονόματος στο σύνολο ακεραίων $\{0, .., 2^{31}\}$.

Κατά την αναζήτηση μέσω του λογικού ονόματος K, επιλέγονται οι εγγραφές του καταλόγου κατακερματισμού για τις οποίες το αποθηκευμένο κλειδί αναζήτησης συμπίπτει με το μετασχηματισμό $g(K)$. Τα αναγνωριστικά συστήματος για τις οντότητες που έχουν λογικά ονόματα για τα οποία η συνάρτηση μετασχηματισμού έχει τιμή που ταυτίζεται με το $g(K)$ τοποθετούνται σε ένα προσωρινό σύνολο αναγνωριστικών συστήματος. Κατόπιν χρησιμοποιείται ο υποκατάλογος αναγνωριστικών συστήματος, μέσω του οποίου γίνεται ανάκληση των πραγματικών λογικών ονομάτων για τις οντότητες που τα αναγνωριστικά τους βρίσκονται στο προσωρινό σύνολο. Τα λογικά αυτά ονόματα συγκρίνονται με το λογικό όνομα για το οποίο γίνεται η αναζήτηση. Το αποτέλεσμα της αναζήτησης είναι τα αναγνωριστικά συστήματος για τις οντότητες που το λογικό τους όνομα συμπίπτει με το δοθέν λογικό όνομα. Στο σημείο αυτό πρέπει να αναφέρουμε ότι αν το λογικό όνομα αντιστοιχεί σε *ανεξάρτητη οντότητα*, το αποτέλεσμα είναι ένα μοναδικό αναγνωριστικό συστήματος, ενώ αν το λογικό όνομα αντιστοιχεί σε γνώρισμα, το αποτέλεσμα είναι το

σύνολο αναγνωριστικών συστήματος για όλα τα γνωρίσματα που έχουν ίδιο λογικό όνομα με το λογικό όνομα για το οποίο γίνεται η αναζήτηση.

Όταν αναζητείται το αναγνωριστικό συστήματος για κάποιο γνώρισμα και δίνονται το λογικό όνομα του γνωρίσματος και το αναγνωριστικό συστήματος για την οντότητα στην οποία ορίζεται το γνώρισμα, το αποτέλεσμα της αναζήτησης είναι πάντοτε το μοναδικό αναγνωριστικό συστήματος του γνωρίσματος. Η αναζήτηση στον κατάλογο κατακερματισμού γίνεται ξανά μέσω της τιμής της συνάρτησης μετασχηματισμού $g(K)$, αλλά το προσωρινό σύνολο αναγνωριστικών συστήματος περιέχει ζεύγη της μορφής (μοναδικό αναγνωριστικό συστήματος, αναγνωριστικό συστήματος για την οντότητα όπου ορίζεται το γνώρισμα) όταν το λογικό όνομα του γνωρίσματος ή η τιμή $g(K)$ δεν είναι μοναδικά. Επειδή όμως το λογικό όνομα του γνωρίσματος μαζί με το αναγνωριστικό για την οντότητα όπου αυτό ορίζεται είναι μοναδικά για όλη τη βάση, ουσιαστικά το προσωρινό σύνολο περιέχει μονάχα ένα μέλος το οποίο περιλαμβάνει το ζητούμενο αναγνωριστικό συστήματος για το γνώρισμα του οποίου το λογικό όνομα δόθηκε στην αρχή της αναζήτησης.

Όταν το λογικό όνομα του γνωρίσματος και η τιμή $g(K)$ είναι μοναδικά, η αναζήτηση γίνεται όπως και στην προηγούμενη περίπτωση, και αφού πραγματοποιηθεί αναζήτηση και στον υποκατάλογο αναγνωριστικών συστήματος συγκρίνονται το αναγνωριστικό για την μοναδική οντότητα και το λογικό όνομα που προκύπτουν με τα δοθέντα, προκειμένου η απάντηση να περιέχει το μοναδικό αναγνωριστικό συστήματος που βρέθηκε από τον κατάλογο κατακερματισμού.

5.4.5 Μηχανισμός βοηθητικών ομάδων εγγραφών

Όπως έχει αναφερθεί, οι εγγραφές που ανατίθενται σε μια συγκεκριμένη θέση του καταλόγου κατακερματισμού είναι δυνατόν να είναι περισσότερες από τη χωρητικότητα της πρωτεύουσας ομάδας εγγραφών για τη θέση αυτή, καθώς το κριτήριο για τη διάσπαση μιας ομάδας εγγραφών δεν εξαρτάται από το αν αυτή υπερχειλίζει, αλλά από το *βαθμό πληρότητας* του καταλόγου κατακερματισμού. Έτσι, οι επιπλέον εγγραφές για κάποια ομάδα εγγραφών που υπερχειλίζει ανατίθενται σε βοηθητικές ομάδες εγγραφών, οι οποίες συνδέονται με μορφή συνδεδεμένης λίστας με την πρωτεύουσα ομάδα εγγραφών για τη θέση του καταλόγου κατακερματισμού όπου αυτές ανήκουν.

Ακόμα, η τιμή της συνάρτησης μετασχηματισμού $g(K)$ για ένα λογικό όνομα K μπορεί να μην είναι μοναδική στο σύνολο $\{g(K) : K \in$ σύνολο λογικών ονομάτων της βάσης$\}$ γιατί είναι δυνατόν διαφορετικές οντότητες να έχουν το ίδιο λογικό όνομα, αλλά και επειδή το σύνολο των πιθανών λογικών ονομάτων της βάσης που αποτελεί πεδίο ορισμού της συνάρτησης μετασχηματισμού είναι πολύ μεγαλύτερο από το πεδίο τιμών της. Στην

περίπτωση που οι τιμές της συνάρτησης μετασχηματισμού συμπίπτουν για περισσότερα από δυο λογικά ονόματα, οι εγγραφές για τις οντότητες στις οποίες αντιστοιχούν αυτά τα λογικά ονόματα τοποθετούνται σε βοηθητικές ομάδες εγγραφών. Οι ομάδες αυτές συνδέονται με τη πρωτεύουσα ομάδα εγγραφών για τη θέση στον κατάλογο κατακερματισμού όπου ανήκουν σχηματίζοντας γραμμικές λίστες.

Ο μηχανισμός διαχείρισης βοηθητικών ομάδων εγγραφών αναλαμβάνει τη διάθεση τέτοιων ομάδων όποτε αυτές χρειάζονται, καθώς επίσης και κάθε προσπέλαση σε αυτές. Ο μηχανισμός αυτός περιλαμβάνει έναν *κατάλογο βοηθητικών ομάδων εγγραφών*, ο οποίος οργανώνεται και λειτουργεί αντίστοιχα με τον *κατάλογο συστήματος* που παρουσιάστηκε στο τέταρτο κεφάλαιο. Η προσπέλαση στον κατάλογο βοηθητικών ομάδων εγγραφών γίνεται μέσω της *διεύθυνσης βοηθητικής ομάδας εγγραφών* και ο κατάλογος αποτελείται από τμήματα. Κάθε τμήμα περιέχει ένα συγκεκριμένο πλήθος *βοηθητικών ομάδων εγγραφών* και έχει μέγεθος ίσο με το μέγεθος της σελίδας δίσκου. Έτσι, οι διευθύνσεις μέσω των οποίων γίνεται κάθε προσπέλαση στον κατάλογο αυτό, έχουν τη μορφή του διανύσματος <*τμήμα*, *θέση*> όπου το πεδίο *τμήμα* καθορίζει τη διεύθυνση του τμήματος και το πεδίο *θέση* καθορίζει τη θέση της βοηθητικής ομάδας εγγραφών μέσα στο τμήμα που την περιέχει. Τα τμήματα εγγραφών στη μνήμη αποθηκεύονται και προσπελαύνονται μέσω μηχανισμού κρυφής μνήμης.

Οι δείκτες για τις λίστες βοηθητικών ομάδων εγγραφών περιέχουν τη διεύθυνση για τις συγκεκριμένες ομάδες και έτσι μπορούν να μετασχηματίζονται άμεσα σε διευθύνσεις στη μνήμη ή το δίσκο. Ο κατάλογος βοηθητικών ομάδων εγγραφών περιλαμβάνει *λίστα ελεύθερων ομάδων εγγραφών* και *περιοχή ελεύθερων ομάδων εγγραφών* κάνοντας αποδοτικότερη την ανάθεση ή ελευθέρωση ομάδων εγγραφών.

5.4.6 Σύστημα αποθήκευσης

Ο κατάλογος κατακερματισμού και ο κατάλογος βοηθητικών ομάδων εγγραφών αποθηκεύονται σε δύο δυαδικά αρχεία του λειτουργικού συστήματος UNIX. Τα αρχεία αυτά περιέχουν επικεφαλίδες στις οποίες κρατείται πληροφορία για το μέγεθος των αρχείων αυτών, αλλά και οι παράμετροι που καθορίζουν την κατάσταση των δυο καταλόγων αντίστοιχα.

Οι σελίδες δίσκου ανατίθενται αντίστοιχα σε τμήματα του καταλόγου κατακερματισμού και τμήματα από βοηθητικές ομάδες εγγραφών. Ο καθορισμός της σελίδας δίσκου που περιέχει κάποιο τμήμα γίνεται άμεσα χρησιμοποιώντας τη διεύθυνση της πρωτεύουσας ή βοηθητικής ομάδας εγγραφών αντίστοιχα.

Για τα δύο αρχεία χρησιμοποιείται η τεχνική no rollback, και έτσι στο τέλος κάποιας δοσοληψίας που πετυχαίνει γράφονται στο δίσκο μονάχα τα τμήματα ομάδων εγγραφών που

έχουν μεταβληθεί κατά τη διάρκεια της δοσοληψίας.

Κεφάλαιο 6

Μηχανισμός κρυφής μνήμης

6.1 Εισαγωγή

Όλα τα συστήματα διαχείρισης βάσεων δεδομένων χρησιμοποιούν μαγνητικούς δίσκους για την αποθήκευση των δεδομένων για τις βάσεις που δημιουργούνται. Οι μαγνητικοί δίσκοι αποτελούν μια μονάδα με μικρό κόστος ανά αποθηκευμένο byte πληροφορίας και όντας ταυτόχρονα σταθερά αποθηκευτικά μέσα, εξασφαλίζουν μονιμότητα για την πληροφορία που αποθηκεύουν. Στα σημερινά συστήματα υπολογιστών όμως, η πληροφορία που αποθηκεύεται στο δίσκο μπορεί να χρησιμοποιηθεί και να μεταβληθεί μονάχα στην μνήμη υπολογιστών. Έτσι, οποτεδήποτε μια δοσοληψία επενεργεί στα δεδομένα κάποιας βάσης αποθηκευμένης σε μαγνητικό δίσκο, κάποιο τμήμα της βάσης δεδομένων πρέπει να μεταφερθεί πρώτα στη μνήμη του υπολογιστή και κατόπιν να γραφεί στο δίσκο μετά την πιθανή τροποποίησή του από τη δοσοληψία. Τα συστήματα διαχείρισης βάσεων δεδομένων περιλαμβάνουν ένα σύστημα ενταμιευτών (buffers) στη μνήμη του υπολογιστή, στους οποίους αποθηκεύονται προσωρινά τα δεδομένα της βάσης που μεταφέρονται από το δίσκο στη μνήμη.

Αν και τα λειτουργικά συστήματα διαθέτουν συστήματα διαχείρισης ενταμιευτών για μεταφορά δεδομένων μεταξύ δίσκου και μνήμης, τα συστήματα διαχείρισης βάσεων δεδομένων διαθέτουν πάντα δικά τους συστήματα διαχείρισης ενταμιευτών στο τμήμα της μνήμης που χρησιμοποιούν οι διεργασίες των χρηστών της βάσης δεδομένων. Η μεταφορά δεδομένων από και προς το δίσκο γίνεται σε βασικές μονάδες που ονομάζονται σελίδες δίσκου (disk pages). Τα περιεχόμενα της βάσης δεδομένων οργανώνονται σε ομάδες ίσες με τις σελίδες δίσκου που ονομάζονται επίσης σελίδες και μπορούν να μεταφερθούν από το δίσκο στη μνήμη με μια μονάχα προσπέλαση στο δίσκο. Το μέγεθος των σελίδων δίσκου σε διαφορετικά συστήματα ποικίλει μεταξύ 512 και 4096 bytes. Οι ενταμιευτές των συστημάτων διαχείρισης

βάσεων δεδομένων χωρίζονται σε τμήματα, κάθε ένα από τα οποία έχει μέγεθος ίσο με το μέγεθος των σελίδων δίσκου. Κάθε φορά που μεταφέρονται δεδομένα μεταξύ της μνήμης και του δίσκου η ποσότητα που μεταφέρεται είναι ίση με το μέγεθος των τμημάτων των ενταμιευτών. Συνήθως το μέγεθος των ενταμιευτών κυμαίνεται μεταξύ 16 kbytes και 16 Mbytes.

Η προσπέλαση σε μια σελίδα της βάσης δεδομένων που βρίσκεται στο δίσκο συνήθως είναι τρεις με τέσσερις τάξεις μεγέθους αργότερη από την προσπέλαση μιας σελίδας που βρίσκεται στη μνήμη [**?**]. Έτσι, κύριος στόχος του συστήματος διαχείρισης ενταμιευτών (buffer manager) είναι η ελαχιστοποίηση των πράξεων μεταφοράς δεδομένων μεταξύ δίσκου και μνήμης για ένα συγκεκριμένο μέγεθος ενταμιευτών.

Ο μηχανισμός διαχείρισης ενταμιευτών προσφέρει κάποια επαφή ζεύξης (interface) μέσω της οποίας πραγματοποιούνται όλες οι προσπελάσεις σε δεδομένα που βρίσκονται στους ενταμιευτές από τις διάφορες εφαρμογές του συστήματος διαχείρισης βάσεων δεδομένων. Κάθε πράξη ανάκλησης δεδομένων της βάσης ονομάζεται *λογική αναφορά* (logical reference). Μόλις συμβεί μια λογική αναφορά, το σύστημα διαχείρισης ενταμιευτών εφαρμόζει κάποιο αλγόριθμο αναζήτησης προκειμένου να διαπιστώσει αν η ζητούμενη *σελίδα* βρίσκεται σε κάποιο ενταμιευτή. Ο αλγόριθμος αυτός εξαρτάται από την οργάνωση του ενταμιευτή, και καθώς οι λογικές αναφορές είναι πολύ συχνές (κάθε πράξη προσπέλασης σε δεδομένα, προκαλεί μια λογική αναφορά), η αναζήτηση πρέπει να υλοποιείται πολύ αποδοτικά. Οι δομές πινάκων ή δέντρων με δείκτες δεν είναι αρκετά αποδοτικές, γιατί η πλοκή για τους αλγόριθμους αναζήτησης εξαρτάται από το μέγεθος του ενταμιευτή (δηλαδή το πλήθος N τμημάτων που αυτός περιέχει), ή επειδή οι πράξεις εισαγωγής και διαγραφής σελίδων επιφέρουν επιπλέον υπολογιστικό κόστος.

Ο πιο αποτελεσματικός τρόπος για την οργάνωση των ενταμιευτών είναι οι *κατάλογοι κατακερματισμού* όπως παρατηρείται στο [**?**] μια και η απόδοση τέτοιων μηχανισμών αναζήτησης είναι σταθερή και ανεξάρτητη από το μέγεθος N των ενταμιευτών.

Αν κατά την επεξεργασία μιας *λογικής αναφοράς* διαπιστωθεί ότι η ζητούμενη σελίδα δε βρίσκεται στον κατάλληλο ενταμιευτή, τότε η σελίδα αυτή πρέπει να διαβαστεί από το δίσκο. Κάθε προσπέλαση στο δίσκο ονομάζεται *φυσική αναφορά* (physical reference) και αποτελεί μια από τις πιο ακριβές πράξεις στα συστήματα διαχείρισης βάσεων δεδομένων. Οι φυσικές αναφορές είναι δύο ειδών: αναφορές ανάγνωσης και αναφορές εγγραφής αντίστοιχα. Οι αναφορές ανάγνωσης συμβαίνουν οποτεδήποτε η σελίδα στην οποία αντιστοιχεί μια λογική αναφορά δεν βρίσκεται στον ενταμιευτή. Οι αναφορές εγγραφής συμβαίνουν όταν έχει ολοκληρωθεί μια δοσοληψία και οι σελίδες που έχουν μεταβληθεί κατά τη διάρκειά της πρέπει να αποθηκευτούν σε σταθερή μνήμη (δηλαδή στο δίσκο), αλλά και όταν πρέπει να

πραγματοποιηθεί μια αναφορά ανάγνωσης και όλα τα τμήματα του ενταμιευτή περιέχουν σελίδες που έχουν υποστεί αλλαγές. Τότε πρέπει κάποια σελίδα να γραφτεί στο δίσκο παραχωρώντας τη θέση της στον ενταμιευτή στη σελίδα που πρέπει να διαβαστεί. Ο καθορισμός της σελίδας η οποία παραχωρεί τη θέση της όταν συμβαίνει κάποια αναφορά ανάγνωσης και όλες οι θέσεις του ενταμιευτή είναι πλήρεις γίνεται από τον *αλγόριθμο αντικατάστασης* (replacement algorithm).

Ο αλγόριθμος αντικατάστασης πρέπει να λειτουργεί ανεξάρτητα από την οργάνωση των ενταμιευτών και να έχει απλή πλοκή ώστε να μην επηρεάζει την απόδοση του συστήματος διαχείρισης ενταμιευτών. Επίσης πρέπει, αν είναι δυνατόν, να μην αντικαθιστά συχνά σελίδες που έχουν μεταβληθεί, μια και αντικατάσταση τέτοιων σελίδων περιλαμβάνει τουλάχιστον δύο εγγραφές στο δίσκο (εγγραφή της ίδιας της σελίδας και καταγραφή της προηγούμενής της κατάστασης σε κάποιο αρχείο καταγραφής μεταβολών στη βάση (log file). Οι αλγόριθμοι αντικατάστασης μπορεί να χρησιμοποιούν πληροφορία σχετικά με τη συχνότητα των λογικών αναφορών (όπως ο αλγόριθμος LFU), σχετικά με το χρόνο που συμβαίνουν οι λογικές αναφορές (όπως οι αλγόριθμοι LRU και FIFO), ή να μην λαμβάνουν υπόψη καμμία πληροφορία (όπως ο αλγόριθμος RANDOM). Η απόδοση κάποιου αλγορίθμου αντικατάστασης σε κάποιο σύστημα διαχείρισης βάσεων δεδομένων εξαρτάται από τη συγκεκριμένη εφαρμογή. Έτσι, η συνολική απόδοση ενός συστήματος μπορεί να βελτιωθεί αν χρησιμοποιηθεί κάποιος μηχανισμός που λαμβάνει υπόψη το ‘‘περιβάλλον’’ μιας εφαρμογής και επιλέγει τον πιο κατάλληλο αλγόριθμο αντικατάστασης για κάθε εφαρμογή.

6.2 Μηχανισμός κρυφής μνήμης για το σύστημα διαχείρισης οντοτήτων

Ο μηχανισμός διαχείρισης ενταμιευτών για το σύστημα διαχείρισης οντοτήτων στη γλώσσα TELOS δεν αποτελεί ένα ενιαίο σύστημα, αλλά διαχωρίζεται σε υπομηχανισμούς, κάθε ένας από τους οποίους αναλαμβάνει τη διαχείριση κάποιου μηχανισμού ενταμιευτών για μια από τις δομές αποθήκευσης και αναπαράστασης που περιγράψαμε στο τέταρτο και πέμπτο κεφάλαιο. Η χρήση επιμέρους μηχανισμών προσφέρει τη δυνατότητα να ρυθμίζονται τα μεγέθη των ενταμιευτών και οι αλγόριθμοι αναζήτησης για κάθε μηχανισμό ξεχωριστά. Με το τρόπο αυτό το σύστημα γίνεται πιο ευέλικτο μια και είναι δυνατόν να ρυθμιστεί ξεχωριστά η συμπεριφορά για κάθε μηχανισμό διαχείρισης ενταμιευτών.

Οι επιμέρους αυτοί μηχανισμοί έχουν ανάλογη δομή και το μόνο που αλλάζει σε αυτούς είναι η δομή των περιεχομένων των τμημάτων για κάθε ενταμιευτή και ο τρόπος προσπέλασης για τους ενταμιευτές. Το μέγεθος των τμημάτων είναι κοινό για όλους τους ενταμιευτές και συμπίπτει με το μέγεθος των *σελίδων δίσκου* για το υποσύστημα διαχείρισης αρχείων του

λειτουργικού συστήματος UNIX.

Προτιμήσαμε την ονομασία *μηχανισμός κρυφής μνήμης* αντί για τη συνήθη ονομασία *μηχανισμός διαχείρισης ενταμιευτών* καθώς οι *κατάλογοι ενταμιευτών* για κάθε επιμέρους μηχανισμό οργανώνονται όπως οι δομές *συσχετιστικής κρυφής μνήμης* (set associative cache memory) που παρουσιάζονται στην εργασία [**?**]. Επίσης δημιουργούν μια ιεραρχία μνήμης, η προσπέλαση στην οποία γίνεται πολύ πιο γρήγορα από κάθε προσπέλαση στο δίσκο. Τέλος, οι επιμέρους μηχανισμοί κρυφής μνήμης είναι το μόνο αποθηκευτικό μέσο στη μνήμη του υπολογιστή, όπου μπορούν να βρίσκονται πιθανά δεδομένα της βάσης.

6.2.1 Οργάνωση καταλόγου κρυφής μνήμης

Ο κατάλογος κρυφής μνήμης οργανώνεται με τη μορφή καταλόγου κατακερματισμού (hash table). Επειδή ο συνολικός χώρος που διατίθεται για τις διάφορες σελίδες που αποτελούν τις εγγραφές για τις διάφορες θέσεις του καταλόγου αυτού είναι σταθερός (ίσος με το συνολικό μέγεθος του εκάστοτε ενταμιευτή τμημάτων για τους επιμέρους μηχανισμούς κρυφής μνήμης), ο κατάλογος κατακερματισμού έχει πάντα σταθερό μέγεθος.

Το κλειδί αναζήτησης για τον κατάλογο κατακερματισμού είναι η *λογική θέση* της εκάστοτε σελίδας. Όπως έχουμε αναφέρει στα προηγούμενα κεφάλαια, για κάθε δομή αποθήκευσης η φυσική διεύθυνση στο δίσκο όπου αποθηκεύονται τα τμήματα της δομής καθορίζεται άμεσα από τη λογική διεύθυνση του κάθε τμήματος. Επίσης κάθε αναφορά, από τις διάφορες εφαρμογές, σε κάποιο τμήμα, γίνεται μέσω της λογικής διεύθυνσης του τμήματος, ενώ η φυσική διεύθυνση του τμήματος χρησιμοποιείται μονάχα όταν το τμήμα διαβάζεται/γράφεται από/στο δίσκο.

Η τιμή της συνάρτησης κατακερματισμού υπολογίζεται από τα λιγότερο σημαντικά ψηφία της λογικής διεύθυνσης του κάθε τμήματος. Έτσι, αν K είναι η λογική διεύθυνση κάποιου τμήματος και N το μέγεθος του καταλόγου κατακερματισμού, η συνάρτηση κατακερματισμού ορίζεται ως εξής:

$$h(K) = K \ mod \ N$$

Επειδή όμως το μέγεθος του καταλόγου κατακερματισμού είναι ακέραια δύναμη του 2 (δηλαδή ισχύει πάντα η σχέση $N = 2^l$ όπου l φυσικός αριθμός) η συνάρτηση κατακερματισμού υπολογίζεται (χρησιμοποιώντας την αριθμητική πράξη and σε δυαδικούς αριθμούς) ως:

$$h(K) = K \ and \ (l - 1)$$

Δηλαδή η τιμή της συνάρτησης $h(K)$ είναι ίση με τα $(l - 1)$ τελευταία ψηφία της δυαδικής αναπαράστασης για τη λογική διεύθυνση K κάθε τμήματος.

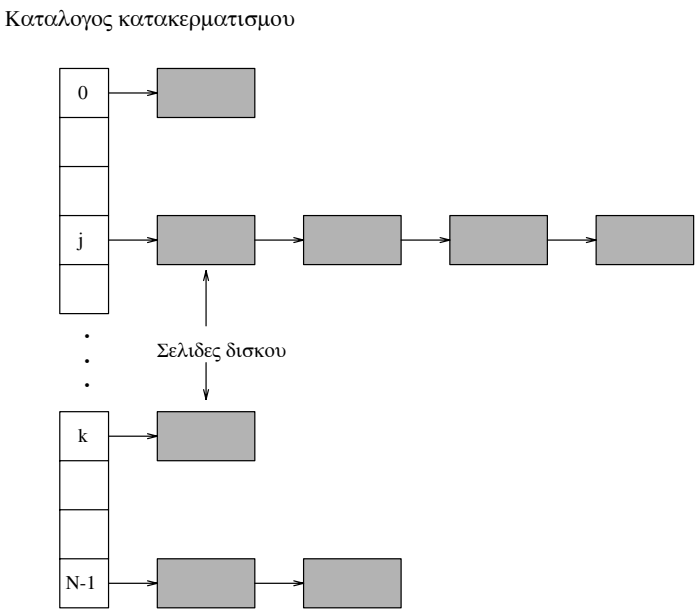
Είναι φανερό ότι όλα τα τμήματα κάποιας δομής αποθήκευσης για τα οποία τα $(l - 1)$ τελευταία ψηφία της δυαδικής αναπαράστασης για τη λογική τους διεύθυνση ταυτίζονται, τοποθετούνται στην ίδια θέση του καταλόγου κατακερματισμού. Για το λόγο αυτό, κάθε θέση του καταλόγου κατακερματισμού μπορεί να περιέχει περισσότερα από ένα τμήματα, επιτρέποντας στις δοσοληψίες να αναφέρονται σε δεδομένα τα οποία βρίσκονται σε λογικές (ή φυσικές) διευθύνσεις οσοδήποτε απομακρυσμένες μεταξύ τους. Αν κάθε θέση του καταλόγου κατακερματισμού μπορούσε να περιέχει μονάχα μια σελίδα, τότε κάθε φορά που θα αποτύγχανε κάποια λογική αναφορά σε μια σελίδα επειδή η θέση αυτή του καταλόγου ήταν κατειλλημένη από άλλη σελίδα, θα έπρεπε να αντικατασταθεί η σελίδα στη θέση αυτή από τη νέα σελίδα που προκαλεί την αποτυχία της λογικής αναφοράς. Με το σχήμα που ακολουθείται όμως, μπορούν και οι δύο αυτές σελίδες να βρίσκονται συγχρόνως στη κρυφή μνήμη χωρίς να αποτυγχάνουν οι διαδοχικές αναφορές σε αυτές.

Το μέγιστο πλήθος σελίδων που μπορούν να βρίσκονται σε μια θέση του καταλόγου κατακερματισμού κυμαίνεται μεταξύ 4 και 16 σελίδων και εξαρτάται από την δομή αποθήκευσης στην οποία αναφέρονται. Το αντίστοιχο αυτής της οργάνωσης είναι οι συσχετιστικές κρυφές μνήμες που συναντώνται στο χώρο της αρχιτεκτονικής υπολογιστών και αποτελούν την αποδοτικότερη μορφή οργάνωσης για κρυφή μνήμη που υλοποιείται μέσω hardware. Η δομή του καταλόγου κατακερματισμού παρουσιάζεται στο σχήμα **??**. Ο χώρος για τις σελίδες που αποθηκεύονται στον κατάλογο αυτό δημιουργείται κατά την εισαγωγή μιας σελίδας σε κάποια θέση του καταλόγου. Με τη μέθοδο αυτή ο χώρος μνήμης που απαιτεί το σύστημα για να λειτουργήσει εξαρτάται από το πλήθος των λογικών αναφορών που προκαλεί η κάθε εφαρμογή. Η εκκίνηση του συστήματος είναι πολύ γρήγορη καθώς δεν χρησιμοποιείται επιπλέον χώρος από τη μνήμη, αλλά τα μεγέθη των ενταμιευτών (που αντιστοιχούν στο πλήθος σελίδων σε κάποιο κατάλογο κατακερματισμού) αυξάνονται δυναμικά μονάχα όταν κάτι τέτοιο είναι απαραίτητο.

Το συνολικό πλήθος σελίδων που μπορούν να βρίσκονται ταυτόχρονα στον κατάλογο κατακερματισμού, δηλαδή το μέγεθος του ενταμιευτή για κάποια δομή αποθήκευσης, μπορεί να είναι μικρότερο από το μέγεθος του καταλόγου κατακερματισμού και μπορεί να καθοριστεί μέσω παραμέτρων τη στιγμή μετάφρασης του κώδικα για τους μηχανισμούς κρυφής μνήμης. Έτσι, δίνεται η δυνατότητα καθορισμού των συνολικών απαιτήσεων σε μνήμη των διαφόρων εφαρμογών για τη γλώσσα TELOS.

6.2.2 Αλγόριθμοι αντικατάστασης

Όταν μια *λογική αναφορά* σε κάποια σελίδα αποτυγχάνει επειδή η σελίδα δε βρίσκεται στον κατάλληλο ενταμιευτή (κατάλογο κατακερματισμού) και ο ενταμιευτής είναι πλήρης,



Σχήμα 6.1: Δομή του καταλόγου κατακερματισμού για την κρυφή μνήμη

Ανάλογα με τις λογικές αναφορές που συμβαίνουν σε κάποια εφαρμογή είναι δυνατόν ο κατάλογος κατακερματισμού να έχει κάποιες θέσεις που περιέχουν περισσότερες από μια σελίδες, ενώ άλλες θέσεις του είναι κενές.

Οι σελίδες που βρίσκονται στην ίδια θέση του καταλόγου ταξινομούνται με βάση τη λογική τους διεύθυνση.

κάποια σελίδα στον ενταμιευτή πρέπει να παραχωρήσει τη θέση της στη νέα σελίδα, η οποία στη συνέχεια διαβάζεται από τη θέση στο δίσκο όπου είναι αποθηκευμένη. Η επιλογή της σελίδας η οποία αντικαθίσταται από τον ενταμιευτή γίνεται από τον αλγόριθμο αντικατάστασης. Στην περίπτωση των συστημάτων διαχείρισης βάσεων δεδομένων, είναι δυνατόν κάποιες σελίδες να μην μπορούν να αντικατασταθούν, γιατί πρέπει να είναι πάντα δυνατή η προσπέλαση σ'αυτές. Τέτοιες συνήθως είναι οι σελίδες που περιέχουν πληροφορία για την δομή της βάσης δεδομένων, αλλά και οι σελίδες όπου περιέχονται αλλαγές ή καινούριες οντότητες που δημιουργούνται κατά την διάρκεια μιας δοσοληψίας. Επίσης όταν συμβαίνουν συχνές προσπελάσεις στα δεδομένα μιας σελίδας, η σελίδα αυτή πρέπει να βρίσκεται συνεχώς στον ενταμιευτή κατά τη διάρκεια αυτών των προσπελάσεων ώστε το σύστημα να παρουσιάζει καλή απόδοση.

Οι αλγόριθμοι αντικατάστασης μπορούν να χωριστούν σε δυο κατηγορίες. Στην πρώτη κατηγορία ανήκουν οι αλγόριθμοι που επιτελούν προανάκληση σελίδων, δηλαδή όταν χρησιμοποιούνται, εκτός από τη σελίδα στην οποία έγινε η λογική αναφορά που απέτυχε, τοποθετούν στον ενταμιευτή και κάποιες άλλες σελίδες στις οποίες είναι πιθανό να γίνουν λογικές αναφορές αργότερα. Έτσι, μεταφέροντας m διαδοχικές σελίδες από το δίσκο στον ενταμιευτή, οι αλγόριθμοι αυτοί αποφεύγουν το κόστος για m διαδοχικές προσπελάσεις στο

δίσκο αργότερα. Στην περίπτωση όμως που οι σελίδες αυτές δε χρησιμοποιούνται, η τεχνική αυτή αυξάνει το κόστος σε πράξεις ανάγνωσης από το δίσκο. Όπως αναφέρεται στο [**?**] οι τεχνικές προανάκλησης για ειδικές περιπτώσεις προσπελάσεων βελτιώνουν σημαντικά την απόδοση των συστημάτων διαχείρισης βάσεων δεδομένων.

Στη δεύτερη κατηγορία αλγορίθμων αντικατάστασης ανήκουν οι αλγόριθμοι που ανακαλούν σελίδες από το δίσκο μονάχα όταν αυτό τους ζητηθεί (demand paging algorithms). Οι αλγόριθμοι αυτοί ανακαλούν μονάχα τη σελίδα για την οποία αποτυχαίνει μια λογική αναφορά. Οι αλγόριθμοι αντικατάστασης που παρουσιάζονται στη συνέχεια και χρησιμοποιούνται από τους μηχανισμούς κρυφής μνήμης ανήκουν σε αυτή την κατηγορία.

Οι αλγόριθμοι αντικατάστασης έχουν σκοπό την ελαχιστοποίηση των λογικών αναφορών που αποτυγχάνουν. Στους αλγόριθμους που παρουσιάζονται, το πλήθος των λογικών αναφορών που αποτυγχάνουν κυμαίνεται μεταξύ του πλήθους των λογικών αναφορών που αποτυγχάνουν όταν χρησιμοποιείται ο αλγόριθμος *OPT* και του πλήθους των λογικών αναφορών που αποτυγχάνουν όταν χρησιμοποιείται ο αλγόριθμος *RANDOM*. Ο αλγόριθμος OPT αντικαθιστά τη σελίδα στην οποία γίνεται λογική αναφορά αργότερα στο μέλλον και μπορεί να χρησιμοποιηθεί για να υπολογισθεί το ελάχιστο πλήθος λογικών αναφορών που αποτυγχάνουν. Επειδή ο αλγόριθμος αυτός χρησιμοποιεί μελλοντική πληροφορία για τη λειτουργία του, δεν είναι υλοποιήσιμος. Ο αλγόριθμος RANDOM αντικαθιστά μια σελίδα που επιλέγεται τυχαία και δε χρησιμοποιεί καμμία πληροφορία για τις προηγούμενες λογικές αναφορές. Ο αλγόριθμος αυτός παρουσιάζει το μέγιστο αποδεκτό πλήθος λογικών αναφορών που αποτυγχάνουν για το σύνολο των πιθανών αλγορίθμων αντικατάστασης.

Οι αλγόριθμοι που παρουσιάζονται στη συνέχεια χρησιμοποιούν πληροφορία από προηγούμενες λογικές αναφορές προκειμένου να επιλέξουν για αντικατάσταση τη σελίδα για την οποία η πιθανότητα να ξαναγίνει άμεσα λογική αναφορά ‘‘είναι ελάχιστη’’. Οι αλγόριθμοι αυτοί θεωρούν ότι αν υπάρχει κάποια τοπικότητα στις λογικές αναφορές, τότε η τοπικότητα που συναντάται στο άμεσο παρελθόν αντιστοιχεί στην τοπικότητα αναφορών που αναμένεται να σημειωθεί κατά τις μελλοντικές λογικές αναφορές.

Οι αλγόριθμοι που παρουσιάζονται στην τρέχουσα εργασία χρησιμοποιούν κάποια από τα παρακάτω κριτήρια για την επιλογή της σελίδας που αντικαθιστούν:

- πότε έγινε η πρώτη λογική αναφορά στη σελίδα
- πότε έγινε η τελευταία λογική αναφορά στη σελίδα
- όλες τις λογικές αναφορές στη σελίδα
- την τελευταία λογική αναφορά στη σελίδα

Η επιλογή ενός μονάχα από τα παραπάνω κριτήρια δε μπορεί να οδηγήσει στην επιλογή κάποιας βέλτιστης πολιτικής αντικατάστασης.

Ο αλγόριθμος FIFO (first-in, first-out) επιλέγει για αντικατάσταση τη σελίδα που τοποθετήθηκε παλαιότερα στον ενταμιευτή. Έτσι, ο αλγόριθμος αυτός δε λαμβάνει καθόλου υπόψη του τη συχνότητα με την οποία συμβαίνουν λογικές αναφορές σε κάποια σελίδα και είναι κατάλληλος μονάχα για εφαρμογές όπου συμβαίνουν σειριακές προσπελάσεις.

Ο αλγόριθμος LFU (least frequently used) επιλέγει για αντικατάσταση τη σελίδα που παρουσιάζει μικρότερη συχνότητα λογικών αναφορών. Σε κάθε σελίδα αντιστοιχεί ένας μετρητής λογικών αναφορών που αυξάνεται κάθε φορά που συμβαίνει λογική αναφορά στη σελίδα αυτή. Όταν οι λογικές αναφορές σε μια σελίδα είναι πολύ συχνές σε κάποιο μικρό διάστημα, ο μετρητής παίρνει μεγάλες τιμές και ακόμη κι αν δε συμβεί άλλη λογική αναφορά στη σελίδα αυτή αργότερα, η σελίδα παραμένει στον ενταμιευτή.

Ο αλγόριθμος LRU (least recently used) επιλέγει για αντικατάσταση τη σελίδα για την οποία δεν έχει συμβεί λογική αναφορά για το μεγαλύτερο χρονικό διάστημα. Όταν χρησιμοποιείται ο μηχανισμός FIX-UNFIX, δηλαδή οι διάφορες εφαρμογές μπορούν να καθορίζουν χρονικά διαστήματα κατά τα οποία είναι γνωστό ότι γίνονται λογικές αναφορές, τότε μπορεί να χρησιμοποιηθεί μια παραλλαγή του αλγορίθμου LRU η οποία αντικαθιστά τη σελίδα που είναι γνωστό ότι δε γίνονται λογικές αναφορές σ' αυτή για το μεγαλύτερο χρονικό διάστημα. Ο αλγόριθμος LRU είναι ο πιο διαδεδομένος αλγόριθμος αντικατάστασης σε συστήματα διαχείρισης βάσεων δεδομένων γιατί προσφέρει ικανοποιητική απόδοση για πληθώρα εφαρμογών. Ο αλγόριθμος αυτός συνήθως χρησιμοποιεί μια δομή διπλά συνδεδεμένης λίστας από σελίδες, όπου η επικεφαλίδα της λίστας αντιστοιχεί στη σελίδα που αντικαθιστάται πρώτη.

Ο αλγόριθμος CLOCK είναι ένας αλγόριθμος που τείνει να προσομοιώσει τον αλγόριθμο LRU χρησιμοποιώντας απλούστερες δομές. Κάθε σελίδα του ενταμιευτή περιέχει ένα πεδίο που ονομάζεται πεδίο χρήσης (use-bit) και παίρνει τιμές 0 ή 1 ανάλογα με το αν έχει συμβεί λογική αναφορά κατά τη διάρκεια ελέγχου όλων των σελίδων του ενταμιευτή. Ένας δείκτης καθορίζει την τρέχουσα σελίδα προς εξέταση. Μια εξεταζόμενη σελίδα αντικαθιστάται αν έχει use-bit=0, ενώ αν use-bit=1 το πεδίο χρήσης μηδενίζεται και ο δείκτης προχωράει στην επόμενη σελίδα του ενταμιευτή. Η σελίδα που αντικαθιστάται είναι πάντα η πρώτη σελίδα για την οποία το πεδίο χρήσης είναι 0. Ο αλγόριθμος αυτός ονομάζεται και SECOND-CHANCE δείχνοντας ότι μια σελίδα για την οποία μηδενίζεται το πεδίο χρήσης αντικαθίσταται μονάχα αν την επόμενη φορά που ο δείκτης φτάσει σ' αυτή (αφού περάσει από όλες τις άλλες σελίδες), το πεδίο χρήσης είναι 0.

Η κατηγορία αλγορίθμων LRD (least reference density) [**?**] χρησιμοποιεί ως κριτήριο τη

μέση πυκνότητα αναφορών σε μια σελίδα. Η σελίδα που αντικαθιστάται έχει ελάχιστη τιμή για τη συνάρτηση πυκνότητας αναφορών (RD):

$$RD(i) \;=\; RC(i)/(GRC - FC(i)) \quad \text{όπου} \quad GRC - FC(i) \;\geq\; 1.$$

Οι παράμετροι για τη συνάρτηση ορίζονται ως εξής:

$RC(i)$**:** μετρητής αναφορών (reference counter) για τη σελίδα i. Ο μετρητής αυτός περιέχει το πλήθος των λογικών αναφορών στη σελίδα i, αφότου η σελίδα i τοποθετήθηκε στον ενταμιευτή.

GRC**:** ολικός μετρητής αναφορών (global reference counter). Περιέχει το συνολικό πλήθος αναφορών που έχουν συμβεί στις σελίδες του ενταμιευτή. Κάθε λογική αναφορά προκαλεί αύξηση του μετρητή αυτού.

$FC(i)$**:** μετρητής αναφοράς (fetch counter). Η τιμή του μετρητή είναι ίση με την τιμή του GRC τη στιγμή που η σελίδα i τοποθετήθηκε στον ενταμιευτή.

Η απλούστερη μορφή αλγορίθμων LRD που παρουσιάστηκε μπορεί με κατάλληλες παραμέτρους να οδηγήσει σε πληθώρα αλγορίθμων αντικατάστασης που μπορεί να ταιριάζουν ικανοποιητικά σε κάποιο σύστημα διαχείρισης βάσεων δεδομένων. Η μορφή αυτή που ήδη χρησιμοποιείται ως μια εναλλακτική λύση για τους αλγόριθμους αντικατάστασης στους διάφορους μηχανισμούς κρυφής μνήμης θα μπορούσε για παράδειγμα να χρησιμοποιεί ένα μηχανισμό για μεταβολή του $RC(i)$ σε τακτά χρονικά διαστήματα (δηλαδή κάθε φορά που πραγματοποιείται ένα συγκεκριμένο πλήθος λογικών αναφορών). Έτσι, θα μπορούσε η μεταβλητή $RC(i)$ να μειώνεται κατά μια σταθερά C κάθε φορά που πραγματοποιούνται N λογικές αναφορές στην κρυφή μνήμη. Για τη σωστή επιλογή κατάλληλων τιμών για τα C και N πρέπει να γίνουν εκτενείς μετρήσεις και να δοκιμαστούν πολλές τιμές, πράγμα που βρίσκεται πέρα από τα πλαίσια της τρέχουσας εργασίας.

Οι γνωστοί αλγόριθμοι MFU (most frequently used) και MRU (most recently used) δε χρησιμοποιούνται μια και η απόδοσή τους για συστήματα διαχείρισης βάσεων δεδομένων είναι χειρότερη από την απόδοση των υπόλοιπων αλγορίθμων εκτός από ελάχιστες περιπτώσεις εφαρμογών.

6.2.3 Επιμέρους μηχανισμοί κρυφής μνήμης

Στη συνέχεια παρουσιάζουμε τον τρόπο προσπέλασης, τους αλγορίθμους αντικατάστασης και τα μεγέθη των πινάκων κατακερματισμού για τους επιμέρους μηχανισμούς κρυφής μνήμης για τις διάφορες δομές αποθήκευσης.

Σε κάθε δομή αποθήκευσης αντιστοιχεί ξεχωριστός μηχανισμός κρυφής μνήμης, για τον οποίο οι διάφορες παράμετροι για μέγεθος αλλά και αλγόριθμοι αντικατάστασης μπορούν να ρυθμιστούν ανεξάρτητα από τους υπόλοιπους μηχανισμούς.

Κατάλογος συστήματος

Οι σελίδες του καταλόγου κατακερματισμού για την κρυφή μνήμη του *καταλόγου συστήματος* φιλοξενούν *τμήματα εγγραφών* με τη μορφή που παρουσιάστηκαν στην παράγραφο 4.2.

Το μέγεθος του καταλόγου κατακερματισμού είναι $N_1 = 128$ και κάθε θέση του καταλόγου μπορεί να περιέχει έως 8 τμήματα εγγραφών. Το πλήθος των τμημάτων όμως που μπορούν να βρίσκονται στην κρυφή μνήμη του καταλόγου συστήματος δεν υπερβαίνει ποτέ τα 256 τμήματα και έτσι το συνολικό μέγεθος της κρυφής μνήμης είναι 1 MByte. Καθώς τα τμήματα εγγραφών περιέχουν δείκτες στη μνήμη για τις οντότητες που βρίσκονται στη μνήμη, ο μηχανισμός κρυφής μνήμης παρακολουθεί συνεχώς το μέγεθος της μνήμης που καταλαμβάνουν οι διάφορες οντότητες και το περιορίζει στα 4 MBytes. Έτσι, ο συνολικός χώρος που μπορεί να καταλαμβάνει η κρυφή μνήμη για τον κατάλογο συστήματος είναι 5 MBytes.

Ο αλγόριθμος αντικατάστασης που χρησιμοποιείται είναι μια παραλλαγή του αλγόριθμου LRU, μέσω της οποίας μπορούν να αντικατασταθούν μονάχα σελίδες που τα περιεχόμενά τους δεν έχουν μεταβληθεί. Ο αλγόριθμος επιλέγει για αντικατάσταση κάποια σελίδα που βρίσκεται στη θέση του καταλόγου κατακερματισμού στην οποία τοποθετείται η σελίδα που προκαλεί τη λογική αναφορά που αποτυγχάνει όταν η θέση αυτή περιέχει 8 σελίδες. Σε κάθε άλλη περίπτωση επιλέγεται η σελίδα που καθορίζεται από τον αλγόριθμο LRU.

Η κρυφή μνήμη μπορεί να περιέχει δεσμευμένες σελίδες (reserved pages) οι οποίες δεν αντικαθίστανται ποτέ. Τέτοιες είναι οι σελίδες που περιέχουν πληροφορία για τις εγγενείς οντότητες, αλλά ο μηχανισμός αυτός μπορεί να μεταβληθεί ώστε να περιλαμβάνει σελίδες που περιέχουν πληροφορία για μετακλάσεις (μια και οι μετακλάσεις ουσιαστικά περιέχουν τη δομή της βάσης δεδομένων) ή μεγάλες οντότητες (καθώς το κόστος προσπέλασης για μεγάλες οντότητες στο δίσκο, είναι πολύ υψηλό, αφού οι οντότητες αυτές με τις επεκτάσεις τους μπορεί να καταλαμβάνουν πολλές σελίδες δίσκου).

Η συνάρτηση κατακερματισμού χρησιμοποιεί τα $log_2 N$ λιγότερα σημαντικά δυαδικά ψηφία του πεδίου *δείκτης τμήματος* (που περιγράφεται στην παράγραφο 4.2.2) για τον καθορισμό της θέσης στον πίνακα κατακερματισμού όπου τοποθετείται κάποιο τμήμα εγγραφών.

Τμήματα οντοτήτων

Κάθε τύπος που αντιστοιχεί σε δομές αναπαράστασης για τους τύπους οντοτήτων στο εννοιολογικό μοντέλο της TELOS αποθηκεύεται σε χωριστά *τμήματα οντοτήτων*. Έτσι, για κάθε τύπο οντοτήτων αντιστοιχεί και ένας διαφορετικός μηχανισμός κρυφής μνήμης. Επίσης ξεχωριστός μηχανισμός κρυφής μνήμης χρησιμοποιείται για τα τμήματα οντοτήτων που περιέχουν *δομές επέκτασης* για τις διάφορες οντότητες.

Το μέγεθος των καταλόγων κατακερματισμού για κάθε ένα από αυτούς τους μηχανισμούς κρυφής μνήμης είναι $N_2 = 32$ και οι θέσεις του κάθε καταλόγου μπορούν να περιέχουν μέχρι 4 διαφορετικά τμήματα οντοτήτων. Και στην περίπτωση αυτή το συνολικό πλήθος των τμημάτων οντοτήτων που αποθηκεύονται σε κάθε κρυφή μνήμη μπορεί να καθοριστεί μέσω κατάλληλων παραμέτρων. Το συνολικό μέγεθος για το χώρο μνήμης που μπορούν να χρησιμοποιήσουν οι μηχανισμοί κρυφής μνήμης για τα τμήματα οντοτήτων είναι 1MByte.

Επειδή η κρυφή μνήμη για τα τμήματα οντοτήτων αποτελεί ενδιάμεση ιεραρχία μνήμης ανάμεσα στο δίσκο και την κρυφή μνήμη για τον κατάλογο συστήματος (ο οποίος περιέχει τα αντίγραφα οντοτήτων που χρησιμοποιούν οι διάφορες εφαρμογές) δεν είναι απαραίτητο να έχει πολύ μεγάλο μέγεθος. Επειδή οι προσπελάσεις στις επιμέρους κρυφές μνήμες για τα τμήματα οντοτήτων δεν είναι τόσο συχνές όσο στην κρυφή μνήμη του καταλόγου συστήματος, ο αλγόριθμος αντικατάστασης που χρησιμοποιείται είναι ο LRD όπως περιγράφηκε προηγουμένως.

Η συνάρτηση κατακερματισμού για κάθε μηχανισμό κρυφής μνήμης χρησιμοποιεί $log_2 N_2$ δυαδικά ψηφία της διεύθυνσης στο δίσκο του εκάστοτε τμήματος οντοτήτων, καθώς η προσπέλαση στα τμήματα οντοτήτων γίνεται μέσω της φυσικής τους διεύθυνσης στο δίσκο. Τα δυαδικά ψηφία της διεύθυνσης που χρησιμοποιούνται από τη συνάρτηση κατακερματισμού βρίσκονται στις θέσεις $log_2 l$ ως και $log_2 l + log_2 N_2$ της διεύθυνσης του συγκεκριμένου τμήματος, όπου l είναι το μέγεθος σε bytes για τα τμήματα εγγραφής.

Κατάλογος λογικών ονομάτων

Στον κατάλογο λογικών ονομάτων χρησιμοποιούνται 3 διαφορετικοί μηχανισμοί κρυφής μνήμης που αντιστοιχούν στις δομές *υποκαταλόγου αναγνωριστικών συστήματος*, *υποκαταλόγου λογικών ονομάτων* και *υποκαταλόγου βοηθητικών εγγραφών*.

Ο αλγόριθμος αντικατάστασης που χρησιμοποιείται για τους μηχανισμούς αυτούς είναι ο LRU, όπως και στην κρυφή μνήμη για τον κατάλογο συστήματος. Συνήθως ο κατάλογος λογικών ονομάτων χρησιμοποιείται όταν ζητείται το λογικό όνομα για κάποια οντότητα της οποίας τα περιεχόμενα ενδιαφέρουν κάποια εφαρμογή ή αλλάζονται από μια δοσοληψία, ή στην περίπτωση που ζητείται προσπέλαση σε κάποια οντότητα, οπότε ο κατάλογος

λογικών ονομάτων χρησιμοποιείται για την εύρεση του αναγνωριστικού συστήματος που καθορίζει την οντότητα προκειμένου να πραγματοποιηθεί η προσπέλαση σ'αυτήν. Έτσι, για κάθε πρόσβαση στον κατάλογο λογικών ονομάτων προηγούνται ή ακολουθούν αντίστοιχες προσβάσεις στον κατάλογο συστήματος. Χρησιμοποιώντας λοιπόν τον ίδιο αλγόριθμο αντικατάστασης βοηθάμε στην ταυτόχρονη παρουσία των εγγραφών του καταλόγου συστήματος και του καταλόγου λογικών οντοτήτων που αφορούν μια συγκεκριμένη οντότητα στις επιμέρους κρυφές μνήμες.

Επειδή οι προσβάσεις στον κατάλογο λογικών ονομάτων δεν είναι τόσο συχνές όσο οι προσβάσεις στον κατάλογο συστήματος, τα μεγέθη για τους επιμέρους μηχανισμούς κρυφής μνήμης για τον κατάλογο λογικών ονομάτων είναι μικρότερα από αυτά του μηχανισμού κρυφής μνήμης για τον κατάλογο συστήματος. Έτσι, το μέγεθος για κάθε κατάλογο κατακερματισμού είναι $N_3 = 64$ και κάθε θέση των καταλόγων κατακερματισμού μπορεί να περιέχει ως 4 τμήματα εγγραφών. Το συνολικό επιτρεπτό μέγεθος είναι 1MByte για τους μηχανισμούς κρυφής μνήμης για τους δυο υποκαταλόγους και 0.5 MBytes για το μηχανισμό κρυφής μνήμης του καταλόγου *βοηθητικών εγγραφών*. Το μέγεθος του μηχανισμού κρυφής μνήμης για τον κατάλογο βοηθητικών εγγραφών μπορεί να αλλάξει σε πολύ μικρότερες τιμές γιατί όπως έχει φανεί από διάφορες εφαρμογές ο κατάλογος αυτός έχει συνήθως πολύ μικρό μέγεθος. Για παράδειγμα σε μια πραγματική εφαρμογή με 450.000 οντότητες ο κατάλογος αυτός περιέχει μονάχα 40 σελίδες ενώ η χωρητικότητα της αντίστοιχης κρυφής μνήμης είναι 128 σελίδες.

Η συνάρτηση κατακερματισμού για τους μηχανισμούς κρυφής μνήμης για τον κατάλογο λογικών ονομάτων ορίζεται με ακριβώς αντίστοιχο τρόπο όπως και για το μηχανισμό κρυφής μνήμης του καταλόγου συστήματος. Τα κλειδιά αναζήτησης για τους τρεις μηχανισμούς είναι ο δείκτης τμήματος (παράγραφος 5.2.1), η τιμή g(K) για τον κατάλογο γραμμικού κατακερματισμού (παράγραφος 5.4.1) και η διεύθυνση βοηθητικής ομάδας εγγραφών (παράγραφος 5.4.5).

6.2.4 Μηχανισμός προανάκλησης

Ο μηχανισμός προανάκλησης που χρησιμοποιείται από τους μηχανισμούς κρυφής μνήμης είναι πολύ απλός και εφαρμόζεται μονάχα κατά την εκκίνηση των διάφορων εφαρμογών. Όταν συμβαίνουν λογικές αναφορές που αποτυγχάνουν δε γίνεται προανάκληση, αλλά τοποθετείται στην εκάστοτε κρυφή μνήμη η σελίδα που προκάλεσε την αποτυχία της λογικής αναφοράς.

Κατά την εκκίνηση του συστήματος τοποθετούνται στην κρυφή μνήμη του *καταλόγου συστήματος* τα τμήματα εγγραφών που αντιστοιχούν σε εγγενείς οντότητες του συστήματος,

αλλά και οι εγγενείς οντότητες. Στην κρυφή μνήμη για τα *τμήματα οντοτήτων* τοποθετούνται τα τμήματα οντοτήτων που αποτελούν την αρχή για κάθε συνδεδεμένη λίστα τμημάτων οντοτήτων. Τα τμήματα αυτά περιέχουν τις οντότητες που δημιουργήθηκαν ή μεταβλήθηκαν πρόσφατα. Στην κρυφή μνήμη του *υποκαταλόγου αναγνωριστικών συστήματος* για τον κατάλογο λογικών ονομάτων, τοποθετούνται τα τμήματα που αφορούν τα λογικά ονόματα των εγγενών οντοτήτων του συστήματος, ενώ δε γίνεται καμία προανάκληση για τον *υποκατάλογο λογικών ονομάτων* και τον *κατάλογο βοηθητικών εγγραφών*. Η προανάκληση για τις εγγενείς οντότητες συμβαίνει γιατί οι οντότητες αυτές και τα λογικά τους ονόματα χρησιμοποιούνται πολύ συχνά από τις διάφορες εφαρμογές.

Ο μηχανισμός προανάκλησης λειτουργεί για σταθερό πλήθος σελίδων δίσκου (συνολικά 13 σελίδες δίσκου: μια για τον κατάλογο συστήματος, 6 για τα τμήματα οντοτήτων και 1 για τον υποκατάλογο αναγνωριστικών συστήματος, καθώς και την πρώτη σελίδα με την επικεφαλίδα για το καθένα από τα 5 αρχεία αποθήκευσης), ανεξάρτητα από το μέγεθος της βάσης δεδομένων. Έτσι, η αρχικοποίηση του συστήματος χρειάζεται ελάχιστο χρόνο για την πραγματοποίησή της καθώς κάθε δομική πληροφορία για τη βάση περιέχεται στις δομές αποθήκευσης χωρίς να απαιτείται καθόλου χρόνος για δημιουργία δομών δεικτοδότησης στη μνήμη.

Η μελέτη του μηχανισμού προανάκλησης είναι ένας χώρος που επιδέχεται αρκετή δουλειά, καθώς ο μηχανισμός αυτός μπορεί να χρησιμοποιηθεί αποτελεσματικά προκειμένου να βελτιωθεί η απόδοση του συστήματος για πολύ μεγάλες βάσεις δεδομένων.

Κεφάλαιο 7

Αξιολόγηση επιδόσεων του συστήματος

7.1 Εισαγωγή

Οι επιδόσεις ενός συστήματος βάσεων δεδομένων είναι ένας από τους σημαντικότερους παράγοντες για την αξιολόγηση του συστήματος και την επιλογή του για κάποια εφαρμογή. Για το σκοπό αυτό έχουν σχεδιαστεί αρκετά *προγράμματα αξιολόγησης επιδόσεων* (benchmarks) για τα παραδοσιακά συστήματα βάσεων δεδομένων, τα οποία χρησιμοποιούν το σχεσιακό μοντέλο για την αναπαράσταση δεδομένων που αποθηκεύουν. Τα προγράμματα αξιολόγησης επιδόσεων περιλαμβάνουν συχνές πράξεις που αναμένεται να πραγματοποιεί κάθε εξεταζόμενο σύστημα. Για την αξιολόγηση του συστήματος εκτελούνται τα προγράμματα αυτά και μετράται ο χρόνος που απαιτείται προκειμένου το σύστημα να πραγματοποιήσει τις συγκεκριμένες πράξεις. Καθώς το μοντέλο αναπαράστασης δεδομένων για τα σχεσιακά συστήματα βάσεων δεδομένων είναι κοινό, το κυριότερο κριτήριο αξιολόγησης τέτοιων συστημάτων, είναι οι χρονικές απαιτήσεις τους για την εκπλήρωση συγκεκριμένων πράξεων, κάτι που μπορεί να υπολογιστεί με τη χρήση των προγραμμάτων μετρήσεως επιδόσεων.

Η αξιολόγηση επιδόσεων για τα παραδοσιακά συστήματα αφορά κυρίως στην εκτίμηση της χρονικής απόδοσης των συστημάτων αυτών κατά τη λειτουργία τους όταν συμβαίνουν πολλές δοσοληψίες ταυτόχρονα από πολλούς χρήστες. Επίσης ένα άλλο μέτρο για την αξιολόγηση των επιδόσεων αφορά στην ικανότητα του μηχανισμού βελτιστοποίησης ερωτήσεων στη βάση (query optimizer) να βελτιστοποιεί πολύπλοκες ερωτήσεις ώστε αυτές να εκτελούνται γρηγορότερα.

Τα οντοκεντρικά συστήματα διαχείρισης βάσεων δεδομένων χρησιμοποιούνται κυρίως σε τεχνικές εφαρμογές (engineering applications) όπως σχεδίαση μέσω υπολογιστή (CAD) ή εφαρμογές αναχρησιμοποίησης λογισμικού (όπως στην περίπτωση του συστήματος για

την TELOS που παρουσιάζεται στην τρέχουσα εργασία). Οι πράξεις στα συστήματα αυτά διαφέρουν από τις πράξεις στα παραδοσιακά συστήματα διαχείρισης βάσεων δεδομένων. Ένα παράδειγμα τυπικής πράξης για το σύστημα της TELOS είναι η αναζήτηση όλων των συναρτήσεων που βρίσκονται στο γράφο κλήσης συναρτήσεων για μια συγκεκριμένη συνάρτηση ή η αναζήτηση όλων των μεταβλητών που χρησιμοποιούνται από κάποιες συναρτήσεις.

Στο κεφάλαιο αυτό αρχικά παρουσιάζουμε το πρόγραμμα μετρήσεως επιδόσεων *OO1* που περιγράφεται στην εργασία [**?**] και αποτελεί το πιο διαδεδομένο τέτοιο πρόγραμμα για αξιολόγηση οντοκεντρικών συστημάτων διαχείρισης βάσεων δεδομένων. Στη συνέχεια αναφέρουμε τις μετρήσεις που πήραμε για το σύστημα διαχείρισης οντοτήτων στη γλώσσα TELOS και συγκρίνουμε τα αποτελέσματα με τα αποτελέσματα για άλλα οντοκεντρικά συστήματα διαχείρισης βάσεων δεδομένων που συναντάμε στη βιβλιογραφία.

7.2 Περιγραφή του προγράμματος αξιολόγησης επιδόσεων

Ο πιο ακριβής τρόπος για την αξιολόγηση επιδόσεων για κάποιο σύστημα διαχείρισης βάσεων δεδομένων που χρησιμοποιείται σε μια τεχνική εφαρμογή θα ήταν να μετρηθεί η χρονική απόδοση του συστήματος για τις διάφορες πράξεις που εκτελούνται στην εφαρμογή αυτή, δεδομένου ότι η αναπαράσταση για τα δεδομένα της εφαρμογής γίνεται με τον αποδοτικότερο τρόπο για το μηχανισμό αναπαράστασης του συγκεκριμένου συστήματος. Ο τρόπος αυτός όμως μπορεί να δώσει μονάχα απόλυτα αποτελέσματα για το κάθε σύστημα βάσεων δεδομένων και δεν προσφέρεται για σύγκριση επιδόσεων για τα διάφορα επιμέρους συστήματα.

Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί μια συγκεκριμένη εφαρμογή αναφοράς που αντιπροσωπεύει όλες τις επιμέρους τεχνικές εφαρμογές. Χρησιμοποιώντας λοιπόν την εφαρμογή αυτή σε διάφορα συστήματα θα μπορούσε να γίνει μια αντικειμενική σύγκριση των συστημάτων αυτών. Επειδή ο σχεδιασμός μιας τόσο γενικής εφαρμογής είναι μια πολύ δύσκολη αν όχι ακατόρθωτη διαδικασία οι σχεδιαστές του προγράμματος *OO1* στράφηκαν σε άλλη κατεύθυνση.

Έτσι, το πρόγραμμα *OO1* χρησιμοποιείται για την αξιολόγηση των επιδόσεων των διαφόρων συστημάτων για λειτουργίες που αναμένεται να συμβαίνουν πολύ συχνά στις τεχνικές εφαρμογές. Οι λειτουργίες αυτές αφορούν ένα μικρό σύνολο διαφορετικών πράξεων σε οντότητες και έτσι οι μετρήσεις μπορούν να πραγματοποιηθούν σε πολλά διαφορετικά συστήματα διαχείρισης βάσεων δεδομένων. Οι μετρήσεις αφορούν πράξεις *εισαγωγής* οντοτήτων στη βάση, *αναζήτησης* οντοτήτων και *διάσχισης συνδέσμων* μεταξύ οντοτήτων της βάσης

δεδομένων.

7.2.1 Περιεχόμενα της βάσης δεδομένων

Η βάση δεδομένων που χρησιμοποιείται για την εκτέλεση του προγράμματος μετρήσεως επιδόσεων είναι ανεξάρτητη από το μοντέλο αναπαράστασης δεδομένων που χρησιμοποιούν τα πιθανά εξεταζόμενα συστήματα διαχείρισης βάσεων δεδομένων.

Οι τύποι δεδομένων για τα δεδομένα της βάσης είναι ο τύπος *τμήμα* και ο τύπος *σύνδεσμος* που περιγράφονται ως εξής:

τμήμα : RECORD[*id* : INT, *type* : STRING, *x*, *y* : INT, *build-date* : DATE]

σύνδεσμος : RECORD[*from* : Part-id, *to* : Part-id, *type* : STRING, *length* : INT]

Η βάση δεδομένων περιέχει *τμήματα* με μοναδικά αναγνωριστικά στο διάστημα $[1, \ldots, 20000]$. Κάθε οντότητα τύπου *τμήμα* συνδέεται ακριβώς με 3 άλλες οντότητες του τύπου *τμήμα* μέσω οντοτήτων τύπου *σύνδεσμος*. Έτσι, η βάση περιέχει συνολικά 80000 οντότητες (20000 οντότητες του τύπου *τμήμα* και 60000 οντότητες του τύπου *σύνδεσμος*). Οι οντότητες που αποτελούν τις οντότητες προορισμού για τις οντότητες *σύνδεσμος* επιλέγονται τυχαία από το σύνολο των οντοτήτων τύπου *τμήμα*. Τα πεδία *x*, *y* και *length* παίρνουν ακέραιες τιμές στο διάστημα $[0, \ldots, 99999]$ που επιλέγονται τυχαία ενώ το πεδίο *type* παίρνει τιμές στο διάστημα [''*part-type0*'',. . .,''*part-type9*'']. Το πεδίο *build-date* παίρνει τιμές τυχαία κατανεμημένες σε κάποιο διάστημα 10 ετών.

Τα πεδία *from* και *to* για τις οντότητες τύπου *σύνδεσμος* δημιουργούνται με τέτοιο τρόπο ώστε να δημιουργείται τοπικότητα αναφορών στη βάση δεδομένων. Για την ακρίβεια, 90 % των συνδέσεων περιλαμβάνουν οντότητες τύπου *τμήμα* που αποτελούν μια κοντινή περιοχή της βάσης δεδομένων και το υπόλοιπο 10 % των συνδέσεων μπορεί να περιλαμβάνει οποιεσδήποτε οντότητες τύπου *τμήμα* της βάσης. Η κοντινή περιοχή για κάποια οντότητα τύπου *τμήμα* ορίζεται ως το 1 % των οντοτήτων τύπου *τμήμα* που βρίσκονται πιο κοντά στη συγκεκριμένη οντότητα. Η τοπικότητα ορίζεται βάση των αριθμητικών τιμών (ή μοναδικών αναγνωριστικών οντοτήτων) για τα πεδία *id* των οντοτήτων τύπου *τμήμα*. Αν θεωρήσουμε ότι τα αναγνωριστικά οντοτήτων βρίσκονται στο διάστημα $[1, \ldots, 20000]$, οι οντότητες τύπου *τμήμα* που αποτελούν την κοντινή περιοχή μιας οντότητας τύπου *τμήμα* με αναγνωριστικό 15000 είναι οι οντότητες με αναγνωριστικό στην περιοχή $[14900, \ldots, 15100]$. Η επιλογή για ποσοστό 90 % τοπικών αναφορών είναι κάπως αυθαίρετη, αλλά αν η τοπικότητα κυμαίνεται σε ποσοστά 1 % έως και 99 % τα αποτελέσματα των επιδόσεων του συστήματος διαχείρισης βάσεων δεδομένων υπό εξέταση δεν επηρεάζονται από αυτήν.

Η βάση δεδομένων που δημιουργείται είναι αντιπροσωπευτική του τμήματος κάποιας μεγαλύτερης βάσης δεδομένων που χρησιμοποιεί μια συγκεκριμένη εφαρμογή ή χρήστης. Φυσικά τα συστήματα διαχείρισης βάσεων δεδομένων πρέπει να μπορούν να λειτουργούν και για πολύ μεγαλύτερες βάσεις δεδομένων. Για το λόγο αυτό η βάση δεδομένων που περιγράψαμε θεωρείται ως η *μικρή βάση δεδομένων* που χρησιμοποιείται στα διάφορα πειράματα μετρήσεων. Τα πειράματα αυτά πραγματοποιούνται και για άλλη μια βάση, στην οποία το πλήθος των οντοτήτων είναι δεκαπλάσιο από το πλήθος των οντοτήτων στη μικρή βάση. Η βάση αυτή αναφέρεται ώς *μεγάλη βάση δεδομένων*. Μερικά συστήματα παρουσιάζουν παρόμοιες επιδόσεις για τη μικρή και τη μεγάλη βάση. Τα συστήματα όμως που χρησιμοποιούν μηχανισμούς κρυφής μνήμης παρουσιάζουν καλύτερες επιδόσεις για τη μικρή βάση.

Όλα τα πειράματα πραγματοποιούνται για απομακρυσμένη (remote) και τοπική (local) βάση δεδομένων. Στην πρώτη περίπτωση η βάση δεδομένων αποθηκεύεται σε κάποιον από τους σταθμούς εξυπηρέτησης ενός δικτύου υπολογιστών και τα πειράματα πραγματοποιούνται σε κάποιο από τους σταθμούς εργασίας του δικτύου. Στην περίπτωση της τοπικής βάσης, η βάση δεδομένων βρίσκεται αποθηκευμένη στο δίσκο του σταθμού εργασίας όπου πραγματοποιούνται τα πειράματα μετρήσεων.

7.2.2 Περιγραφή των πειραμάτων

Το πρόγραμμα μετρήσεως επιδόσεων υπολογίζει το χρόνο απόκρισης ενός συστήματος (response time) και εκτελείται από ένα μονάχα χρήστη. Οι μετρήσεις αυτές αποτελούν το αντίστοιχο τεχνικών εφαρμογών όπου κάποιος μηχανικός παίρνει ένα τμήμα μιας μεγάλης βάσης δεδομένων, το οποίο χρησιμοποιεί αποκλειστικά ο ίδιος. Συνολικά ελέγχονται 3 λειτουργίες. Κάθε λειτουργία εκτελείται 10 φορές και ο χρόνος απόκρισης του συστήματος καταγράφεται ώστε να διαπιστωθεί αν οι μετρήσεις βρίσκονται σε αντιστοιχία μεταξύ τους. Επίσης η επανάληψη του ίδιου πειράματος επιτρέπει στους μηχανισμούς κρυφής μνήμης,σε συστήματα που διαθέτουν τέτοιους μηχανισμούς, να λειτουργήσουν βελτιώνοντας τις επιδόσεις του συστήματος υπό εξέταση.

Οι λειτουργίες για τις οποίες γίνονται τα πειράματα είναι οι εξής:

(1) Αναζήτηση: Κατά τη λειτουργία αυτή προσπελαύνονται και μεταφέρονται στη μνήμη οι οντότητες τύπου *τμήμα* για 1000 τυχαία επιλεγμένα αναγνωριστικά οντοτήτων τύπου *τμήμα*. Για κάθε οντότητα καλείται μια κενή διαδικασία με παραμέτρους τα γνωρίσματα *x*, *y* και *type* της οντότητας.

(2) Διάσχιση συνδέσεων: Κατά τη λειτουργία αυτή αναζητούνται όλες οι οντότητες τύπου

τμήμα με τις οποίες συνδέεται (μέσω οντοτήτων τύπου *σύνδεσμος*) μια οντότητα τύπου *τμήμα* που επιλέγεται τυχαία. Η αναζήτηση αυτή εφαρμόζεται διαδοχικά σε κάθε οντότητα τύπου *τμήμα* που προσπελαύνεται, ακολουθώντας έτσι ένα μονοπάτι συνδέσεων μεταξύ τμημάτων μήκους 7. Κατά τη λειτουργία αυτή λοιπόν προσπελαύνονται συνολικά $\sum_{i=0}^{7} 3^i = 3280$ οντότητες τύπου *τμήμα*. Για κάθε τέτοια οντότητα καλείται μια κενή διαδικασία, όπως στο πείραμα 1.

Η λειτουργία αυτή πραγματοποιείται και προς τις δύο κατευθύνσεις ακολουθώντας τα πεδία *from* και *to* της οντότητας τύπου *σύνδεσμος* για οντότητες τύπου *τμήμα* που συνδέονται μαζί της. Έτσι, κατά τη λειτουργία αυτή παίρνουμε δυο μετρήσεις, για διάσχιση συνδέσμων κατά την κανονική και την αντίστροφη φορά.

(3) Εισαγωγή δεδομένων στη βάση: Εισαγωγή 100 οντοτήτων τύπου *τμήμα* και 3 οντοτήτων τύπου *σύνδεσμος* για κάθε μια από αυτές τις οντότητες. Η λειτουργία αυτή εισάγει συνολικά 1000 οντότητες τύπου *τμήμα* και 3000 οντότητες τύπου *σύνδεσμος* στη βάση δεδομένων καθώς εκτελείται 10 φορές. Στην περίπτωση αυτή κάθε επανάληψη αποτελεί ξεχωριστή δοσοληψία, και για κάθε δοσοληψία υπολογίζεται ο συνολικός χρόνος ώστε να γίνουν όλες οι απαραίτητες ενημερώσεις για τα δεδομένα της βάσης στο δίσκο.

Οι λειτουργίες που μόλις αναφέραμε έχουν επιλεγεί ώστε να είναι αντιπροσωπευτικές των λειτουργιών που πραγματοποιούνται σε βάσεις δεδομένων που χρησιμοποιούνται σε τεχνικές εφαρμογές. Για παράδειγμα, η λειτουργία διάσχισης συνδέσεων είναι το αντίστοιχο της διάσχισης κάποιου γράφου κλήσεων συναρτήσεων σε μια τυπική εφαρμογή της TELOS, ενώ η λειτουργία αναζήτησης είναι το αντίστοιχο της αναζήτησης όλων των συναρτήσεων που έχουν κάποια κοινή ιδιότητα.

Κατά την πρώτη επανάληψη κάθε μιας από τις λειτουργίες οι ενταμιευτές δεν περιέχουν καθόλου δεδομένα της βάσης και έτσι κάθε λογική αναφορά σε δεδομένα της βάσης οδηγεί σε ανάγνωση της κατάλληλης σελίδας από το δίσκο. Οι μετρήσεις για την πρώτη επανάληψη κάθε λειτουργίας αναφέρονται ως μετρήσεις με κενή κρυφή μνήμη (cold start results). Στις υπόλοιπες επαναλήψεις των λειτουργιών, οι ενταμιευτές στους μηχανισμούς κρυφής μνήμης έχουν αρχίσει να περιέχουν σελίδες δίσκου και έτσι τα αποτελέσματα των μετρήσεων αυτών αναφέρονται ως αποτελέσματα μετρήσεων με κρυφή μνήμη (warm startup results). Σε κάθε επανάληψη προσπελαύνονται διαφορετικές οντότητες. Επειδή όμως η μικρή βάση δεδομένων σε πολλά συστήματα έχει μέγεθος μικρότερο από το μέγεθος της κρυφής μνήμης, είναι δυνατόν να βρίσκεται σχεδόν όλη η βάση δεδομένων στη μνήμη του υπολογιστή κατά την τελευταία μέτρηση. Έτσι, καθώς αυξάνεται ο αριθμός των επαναλήψεων οι μετρήσεις γίνονται καλύτερες. Η καλύτερη τιμή των μετρήσεων αυτών καταγράφεται ως μέτρηση με κρυφή μνήμη.

Τα αποτελέσματα για την αντίστροφη διάσχιση συνδέσεων είναι δυνατό να παρουσιάζουν μεγάλες διακυμάνσεις. Οι διακυμάνσεις αυτές οφείλονται στο γεγονός ότι κάθε οντότητα τύπου *τμήμα* συνδέεται ακριβώς με 3 οντότητες τύπου *τμήμα* μέσω του πεδίου *to* κάποιας οντότητας τύπου *σύνδεσμος*, όχι όμως και μέσω του πεδίου *from*. Έτσι, κάθε οντότητα τύπου *τμήμα* έχει τυχαίο αριθμό συνδέσεων αντίστροφης φοράς μέσω οντοτήτων τύπου *σύνδεσμος* με άλλες οντότητες τύπου *τμήμα*, πράγμα που προκαλεί αυτή τη διακύμανση στις μετρήσεις. Τα αποτελέσματα για την λειτουργία διάσχισης συνδέσεων κατά την αντίστροφη φορά κανονικοποιούνται ως προς το πλήθος των οντοτήτων τύπου *τμήμα* που λαμβάνουν μέρος στη διάσχιση. Αν λοιπόν μια τέτοια λειτουργία προσπελαύνει K οντότητες τύπου *τμήμα* σε χρόνο t, η κανονικοποιημένη τιμή για το χρόνο διάσχισης που αναφέρεται είναι η τιμή $\frac{t \times 3280}{K}$. Η τιμή αυτή αναφέρεται στο χρόνο που θα απαιτούσε η διάσχιση αν αντί για K, προσπελαύνονταν 3280 οντότητες τύπου *τμήμα*.

7.3 Μετρήσεις επιδόσεων για το σύστημα της TELOS

Τα πειράματα πραγματοποιούνται σε ένα σταθμό εργασίας Sun SPARCstation ELC(4/25) με 8 Mbytes μνήμη και τοπικό δίσκο με διαθέσιμα 60 Mbytes που χρησιμοποιούνται ως swap space. Ο δίσκος αυτός περιέχει τη βάση δεδομένων για τις μετρήσεις για την τοπική βάση. Ο σταθμός εξυπηρέτησης που περιέχει την απομακρυσμένη βάση δεδομένων είναι ένα Sun SS-10. Ο δίσκος αυτός κατά τη διάρκεια των πειραμάτων είχε πληρότητα της τάξεως 99 %. Το λειτουργικό σύστημα που χρησιμοποιείται είναι το SunOS-4.1.2.

Κατά τη διάρκεια των πειραμάτων, φροντίσαμε ώστε τις περισσότερες φορές ο σταθμός εργασίας να απασχολείται από ένα μονάχα χρήστη, αλλά δεν μπορέσαμε να εξασφαλίσουμε αποκλειστική χρήση του σταθμού αυτού από τη διεργασία που πραγματοποιούσε τα πειράματα. Το ίδιο ισχύει και για το σταθμό εξυπηρέτησης όπου βρισκόταν αποθηκευμένη η απομακρυσμένη βάση. Έτσι, τα πειράματα πραγματοποιήθηκαν αρκετές φορές, ώστε να διαπιστωθεί αν υπήρχε απόκλιση στις μετρήσεις λόγω διαφορετικού φόρτου στους σταθμούς εργασίας. Οι τιμές των μετρήσεων παρουσιάζουν μια απόκλιση της τάξεως του 10 % και για το λόγο αυτό οι τιμές που αναφέρουμε είναι οι τιμές που καταγράφηκαν από τα προγράμματα που πραγματοποιούσαν τις μετρήσεις και προσεγγίζουν περισσότερο το μέσο όρο όλων των μετρήσεων που πραγματοποιήσαμε.

Κατά τις μετρήσεις αυτές θεωρήσαμε σκόπιμο να χρησιμοποιήσουμε δυο διαφορετικές βάσεις δεδομένων με διαφορετικές τιμές για την τοπικότητα των δεδομένων που αναφέρεται στην προηγούμενη παράγραφο. Η πρώτη βάση δεδομένων περιέχει συνδέσεις μεταξύ οντοτήτων τύπου *τμήμα* που δημιουργούνται με τυχαίο τρόπο χωρίς τοπικότητα, ενώ η δεύτερη βάση

χρησιμοποιεί τοπικότητα 90 % στις συνδέσεις μεταξύ οντοτήτων τύπου *τμήμα* όπως απαιτεί η περιγραφή του προγράμματος μετρήσεως επιδόσεων. Οι δύο αυτές βάσεις χρησιμοποιούνται στα πειράματα προκειμένου να διαπιστωθεί η συμπεριφορά του συστήματος ανάλογα με το αν υπάρχει τοπικότητα στις συνδέσεις ή όχι.

Για κάθε μια από τις βάσεις που χρησιμοποιούνται, εκτός από τις μετρήσεις χωρίς ή με κρυφή μνήμη, αναφέρεται και άλλη μια μέτρηση ως *μέσος όρος μετρήσεων με κρυφή μνήμη.* Η μέτρηση αυτή αν και δεν ζητείται, πιστεύουμε ότι αποτελεί ένα καλό μέτρο για το πόσο γρήγορα αρχίζει να αποδίδει αρκετά ο μηχανισμός κρυφής μνήμης. Έτσι, όταν η τιμή αυτή βρίσκεται πολύ κοντά στην τιμή για την καλύτερη μέτρηση με χρήση κρυφής μνήμης, η βέλτιστη τιμή για το μηχανισμό κρυφής μνήμης επιτυγχάνεται σε μικρό αριθμό επαναλήψεων του πειράματος.

7.3.1 Μετρήσεις για τη μικρή βάση δεδομένων

Μετρήσεις για τη μικρή βάση δεδομένων χωρίς τοπικότητα

Ο πίνακας **??** παρουσιάζει τα αποτελέσματα για τη μικρή βάση δεδομένων χωρίς τοπικότητα συνδέσεων μεταξύ οντοτήτων, η οποία βρίσκεται αποθηκευμένη σε διαφορετικό σταθμό εργασίας από αυτόν που πραγματοποιούνται οι μετρήσεις.

Μετρήσεις για μικρή απομακρυσμένη βάση			
Είδος μέτρησης	Αρχική	Με κρυφή μνήμη	
		Βέλτιστη	Μέση τιμή
Αναζήτηση	6.777	0.648	0.880
Κανονική Διάσχιση	18.613	2.244	4.613
Αντίστροφη Διάσχιση	96.015	12.391	20.633
Αντίστροφη Διάσχιση (2)	23.789	2.768	3.403
Εισαγωγή δεδομένων	10.400	7.540	7.540
Συνολικός Χρόνος	35.790	10.432	13.033

Πίνακας 7.1: Αποτελέσματα των πειραμάτων για μικρή απομακρυσμένη βάση

Το πεδίο *Συνολικός χρόνος* σε όλους τους πίνακες αναφέρεται μονάχα στα αθροίσματα των τιμών για τα πεδία *Αναζήτηση*, *Κανονική Διάσχιση* και *Εισαγωγή δεδομένων*

Όπως φαίνεται από τα αποτελέσματα του πίνακα, ο χρόνος που απαιτείται για τις λειτουργίες της αναζήτησης και της διάσχισης συνδέσεων κατά την κανονική φορά είναι αρκετά μικρός, και στις περιπτώσεις όπου έχει ήδη αρχικοποιηθεί η κρυφή μνήμη, το σύστημα μπορεί να πραγματοποιεί περισσότερες από 1000 προσπελάσεις σε οντότητες ανά

δευτερόλεπτο. Η τιμή αυτή είναι πολύ αξιόλογη, καθώς αν οι προσπελάσεις που δημιουργούν λογικές αναφορές που αποτυγχάνουν αποτελούν το 10 % των συνολικών λογικών αναφορών σε 1000 λογικές αναφορές, τότε ο ελάχιστος χρόνος για να μεταφερθούν οι αντίστοιχες σελίδες από το δίσκο στη μνήμη υπερβαίνει το 1 δευτερόλεπτο. Οι σύγχρονοι δίσκοι έχουν χρόνο προσπέλασης της τάξης των 10-15 msecs, οπότε 100 προσπελάσεις στο δίσκο απαιτούν τουλάχιστον 1-1.5 δευτερόλεπτα για την πραγματοποίησή τους. Επίσης για να επιτευχθούν οι τιμές που παρουσιάζονται στον πίνακα **??** πρέπει το ποσοστό επιτυχίας αναζητήσεων στη κρυφή μνήμη να είναι τουλάχιστον 90 %, πράγμα που αποτελεί μια πολύ καλή τιμή για την απόδοση των μηχανισμών κρυφής μνήμης.

Δυστυχώς όμως οι μετρήσεις για τη διάσχιση κατά την αντίστροφη κατεύθυνση δεν δείχνουν να είναι τόσο καλές όσο για τις προηγούμενες δύο περιπτώσεις, παρ'όλο που η δομή των οντοτήτων στο σύστημα αποθήκευσης επιτρέπει την διάσχιση συνδέσεων και προς τις δυο κατευθύνσεις εξίσου αποδοτικά. Η διαφορά των μετρήσεων οφείλεται στο γεγονός ότι κάθε επανάληψη της λειτουργίας αντίστροφης διάσχισης προσπελαύνει διαφορετικό πλήθος οντοτήτων. Το πλήθος αυτό κυμαίνεται συνήθως μεταξύ 10 και 500 οντοτήτων. Έτσι, αν σε κάποια επανάληψη της λειτουργίας αυτής γίνονται 10 λογικές αναφορές σε οντότητες και οι μισές από τις αναφορές αποτυγχάνουν προκαλώντας το διάβασμα των κατάλληλων σελίδων από το δίσκο, είναι φυσικό η μέτρηση για το χρόνο να επηρεάζεται σημαντικά από το χρόνο για τις προσπελάσεις στο δίσκο. Επίσης ο χρόνος που αναφέρεται στη λειτουργία της αντίστροφης διάσχισης, προκύπτει μετά την κανονικοποίηση του πραγματικού χρόνου που μετρήθηκε. Έτσι, οι αναφερόμενες τιμές στο πεδίο αυτό είναι συνήθως πολύ υψηλές και δεν ανταποκρίνονται στις τιμές που αναμέναμε, οι οποίες θα ήταν παραπλήσιες με τις τιμές για τη δάσχιση κατά την κανονική φορά.

Για το λόγο αυτό πραγματοποιήσαμε άλλο ένα πείραμα αντίστροφης διάσχισης για συνδέσεις μεταξύ τμημάτων. Για το πείραμα αυτό χρησιμοποιήθηκε μια ελαφρώς διαφορετική βάση δεδομένων. Στη βάση αυτή, κάθε οντότητα τύπου *τμήμα* αποτελεί το πεδίο *το* για ακριβώς τρεις οντότητες τύπου *σύνδεσμος*. Δηλαδή στη βάση αυτή απλά αντιστρέψαμε τη φορά των συνδέσμων. Το πείραμα διάσχισης συνδέσεων κατά την αντίστροφη φορά για τη βάση αυτή έδωσε μετρήσεις παραπλήσιες με αυτές του πειράματος διάσχισης κατά την κανονική κατεύθυνση, όπως άλλωστε περιμέναμε. Οι μετρήσεις για το πείραμα αυτό φαίνονται στη γραμμή *Αντίστροφη Διάσχιση (2)* του πίνακα **??**. Μπορούμε να παρατηρήσουμε ότι οι μετρήσεις αυτές παρουσιάζουν μια απόκλιση της τάξεως του 20 % από τις μετρήσεις για την κανονική διάσχιση.

Η διαφορά αυτή όμως δεν πρέπει να μας προβληματίζει για τους παρακάτω λόγους. Ο φόρτος του συστήματος κατά τα δύο διαφορετικά πειράματα που έδωσαν αυτές τις μετρήσεις

ήταν διαφορετικός, αφού δεν μπορούσαμε να εξασφαλίσουμε αποκλειστικότητα στη χρήση των υπολογιστών για τη διεργασία που πραγματοποιούσε τις μετρήσεις. Επίσης, ο κώδικας του προγράμματος μετρήσεως επιδόσεων για τη λειτουργία της αντίστροφης διάσχισης είναι πιο πολύπλοκος, αφού πρέπει να πραγματοποιεί περισσότερους ελέγχους, από τον κώδικα για την διάσχιση κατά την κανονική φορά. Ακόμη, η φυσική τοποθέτηση των διαφόρων σελίδων στο δίσκο διαφέρει για τις δύο βάσεις δεδομένων, κάτι που επηρεάζει το χρόνο αναζήτησης (seek time) για τις βάσεις αυτές. Τέλος, πρέπει να αναφέρουμε ότι και η διαφορετική τοπικότητα που δημιουργείται στις συνδέσεις ανάμεσα σε οντότητες στις δύο βάσεις είναι ένας παράγοντας που μπορεί να επηρεάσει τις μετρούμενες τιμές από τα πειράματα διάσχισης. Λαμβάνοντας λοιπόν υπόψη τους παράγοντες αυτούς, θεωρούμε ότι οι δυο μετρήσεις βρίσκονται σε συμφωνία μεταξύ τους.

Οι μόνες μετρήσεις που απέχουν σημαντικά από τις υπόλοιπες, είναι οι μετρήσεις για το χρόνο εισαγωγής δεδομένων στη βάση. Οι τιμές για αυτές τις μετρήσεις επηρεάζονται από το γεγονός ότι ο μηχανισμός ελέγχου για τη σημασιολογική συνέπεια των δεδομένων είναι σχετικά πολύπλοκος, καθώς η γλώσσα TELOS υποστηρίζει ποικίλους μηχανισμούς αναπαράστασης. Επίσης η εύρεση νέων σελίδων δίσκου από το λειτουργικό σύστημα όπου γράφονται τα νέα δεδομένα της βάσης είναι πιο αργός, επειδή οι δίσκοι είχαν μεγάλο βαθμό πληρότητας και παρουσίαζαν μεγάλο κατακερματισμό όσον αφορά τη φυσική θέση των διαφόρων σελίδων.

Ο πίνακας **??** παρουσιάζει τα αποτελέσματα για τη μικρή βάση δεδομένων χωρίς τοπικότητα συνδέσεων μεταξύ οντοτήτων, η οποία βρίσκεται αποθηκευμένη στον τοπικό δίσκο του σταθμού εργασίας όπου πραγματοποιούνται οι μετρήσεις.

Μετρήσεις για μικρή τοπική βάση			
Είδος μέτρησης	Αρχική	Με κρυφή μνήμη	
		Βέλτιστη	Μέση τιμή
Αναζήτηση	6.426	0.653	0.759
Κανονική Διάσχιση	17.679	2.281	3.131
Αντίστροφη Διάσχιση	65.186	0.849	22.311
Αντίστροφη Διάσχιση (2)	19.201	3.109	4.346
Εισαγωγή δεδομένων	8.100	6.470	6.470
Συνολικός Χρόνος	32.205	9.404	10.360

Πίνακας 7.2: Αποτελέσματα των πειραμάτων για μικρή τοπική βάση

Οι μετρήσεις στον πίνακα αυτό είναι παρόμοιες με τις μετρήσεις στον πίνακα **??**, αλλά

παρατηρούνται σχετικά διαφορετικές τιμές για το χρόνο εισαγωγής δεδομένων στη βάση. Ο τοπικός δίσκος έχει αρκετό ελεύθερο χώρο και έτσι η εισαγωγή νέων σελίδων δίσκου στη βάση πραγματοποιείται γρηγορότερα από ότι στην περίπτωση της απομακρυσμένης βάσης.

Επίσης παρατηρούμε ότι οι βέλτιστοι χρόνοι που δίνουν οι μηχανισμοί κρυφής μνήμης είναι ελαφρώς χειρότεροι, πράγμα που μπορεί να οφείλεται στην απόδοση του τοπικού δίσκου η οποία είναι μικρότερη από την απόδοση του δίσκου που περιέχει την απομακρυσμένη βάση.

Η μέτρηση για το βέλτιστο χρόνο κατά την αντίστροφη διάσχιση αξίζει προσοχής για την πολύ μικρή της τιμή. Η τιμή αυτή οφείλεται στο γεγονός ότι κατά την επανάληψη που την προκάλεσε προσπελαύνεται μονάχα μια οντότητα η οποία βρίσκεται ήδη στην κρυφή μνήμη, καθώς ο πραγματικός χρόνος που απαιτήθηκε για τη συγκεκριμένη επανάληψη είναι μόλις 0.25 msecs.

Μετρήσεις για τη μικρή βάση δεδομένων με τοπικότητα

Στον πίνακα **??** παρουσιάζονται οι μετρήσεις για τη μικρή βάση δεδομένων που περιέχει τοπικότητα στις συνδέσεις μεταξύ των οντοτήτων τύπου *τμήμα* και βρίσκεται αποθηκευμένη στο δίσκο του σταθμού εξυπηρέτησης.

Μετρήσεις για μικρή απομακρυσμένη βάση			
Είδος μέτρησης	Αρχική	Με κρυφή μνήμη	
		Βέλτιστη	Μέση τιμή
Αναζήτηση	6.767	0.644	0.748
Κανονική Διάσχιση	10.561	1.120	2.010
Αντίστροφη Διάσχιση	12.382	1.567	4.028
Αντίστροφη Διάσχιση (2)	14.914	1.075	2.483
Εισαγωγή δεδομένων	7.900	5.510	5.510
Συνολικός Χρόνος	25.228	7.274	8.268

Πίνακας 7.3: Αποτελέσματα των πειραμάτων για μικρή απομακρυσμένη βάση με τοπικότητα στις συνδέσεις

Μπορούμε να παρατηρήσουμε ότι ο χρόνος για την αρχική αναζήτηση, είναι παραπλήσιος με το χρόνο αναζήτησης για την βάση χωρίς τοπικότητα, αφού η τοποθέτηση των οντοτήτων τύπου *τμήμα* είναι ίδια και για τις δυο βάσεις.

Σημαντική είναι η διαφορά τους για τους αρχικούς χρόνους διάσχισης σε σχέση με τη μικρή βάση χωρίς τοπικότητα στις συνδέσεις μεταξύ οντοτήτων. Οι χαμηλές τιμές οφείλονται στο γεγονός ότι οι περισσότερες προσπελάσεις στο δίσκο κατά τις οποίες με-

ταφέρονται ολόκληρες σελίδες δίσκου στην κρυφή μνήμη, μεταφέρουν σελίδες στις οποίες πραγματοποιούνται αργότερα και άλλες λογικές αναφορές.

Επίσης οι τιμές για την μέτρηση αντίστροφης διάσχισης συνδέσμων είναι πολύ κοντά στις τιμές για τη διάσχιση όπως και οι τιμές για την αντίστροφη διάσχιση στη βάση με τους αντίστροφους συνδέσμους. Η σύμπτωση των τιμών οφείλεται στην τοπικότητα που υπάρχει στη βάση μια και τώρα συμβαίνουν λιγότερες λογικές αναφορές που αποτυγχάνουν για την αντίστροφη διάσχιση από ότι στις περιπτώσεις που παρουσιάζονται στους πίνακες **??** και **??**.

Στον πίνακα **??** παρουσιάζονται οι μετρήσεις για την ίδια βάση, η οποία όμως

Μετρήσεις για μικρή τοπική βάση			
Είδος μέτρησης	Αρχική	Με κρυφή μνήμη	
		Βέλτιστη	Μέση τιμή
Αναζήτηση	5.435	0.643	0.739
Κανονική Διάσχιση	6.134	0.734	1.559
Αντίστροφη Διάσχιση	13.890	1.095	2.686
Αντίστροφη Διάσχιση (2)	14.054	1.069	1.602
Εισαγωγή δεδομένων	7.200	4.480	4.480
Συνολικός Χρόνος	18.769	5.857	6.288

Πίνακας 7.4: Αποτελέσματα των πειραμάτων για μικρή τοπική βάση με τοπικότητα στις συνδέσεις

βρίσκεται αποθηκευμένη στον τοπικό δίσκο του σταθμού εργασίας όπου πραγματοποιούνται οι μετρήσεις.

Οι μετρήσεις του πίνακα βρίσκονται πολύ κοντά στις μετρήσεις του πίνακα **??** με εξαίρεση τις μετρήσεις για εισαγωγή δεδομένων. Αντίστοιχες διαφορές παρατηρήσαμε και για τους πίνακες **??** και **??** αντίστοιχα, καθώς ο τοπικός και ο απομακρυσμένος δίσκος έχουν διαφορετικές τιμές πληρότητας.

7.3.2 Μετρήσεις για τη μεγάλη βάση δεδομένων

Για τη μεγάλη βάση δεδομένων αναφέρουμε μονάχα μετρήσεις για απομακρυσμένη βάση δεδομένων, μια και ο τοπικός δίσκος του σταθμού εργασίας όπου πραγματοποιήθηκαν οι μετρήσεις δεν ήταν αρκετά μεγάλος για να την αποθηκεύσει. Οι μετρήσεις πραγματοποιούνται μονάχα για τη βάση όπου υπάρχει τοπικότητα για τις συνδέσεις μεταξύ οντοτήτων.

Οι μετρήσεις για τη βάση αυτή παρουσιάζονται στο σχήμα **??**. Στην περίπτωση αυτής

της βάσης οι λογικές αναφορές αποτυγχάνουν τις περισσότερες φορές προκαλώντας συχνές προσπελάσεις στο δίσκο για τις αντίστοιχες σελίδες.

Μετρήσεις για μεγάλη απομακρυσμένη βάση			
Είδος μέτρησης	Αρχική	Με κρυφή μνήμη	
		Βέλτιστη	Μέση τιμή
Αναζήτηση	46.061	26.433	29.322
Κανονική Διάσχιση	72.420	49.294	61.119
Αντίστροφη Διάσχιση	65.773	44.112	55.811
Εισαγωγή δεδομένων	17.100	11.040	11.040
Συνολικός Χρόνος	135.581	86.767	101.481

Πίνακας 7.5: Αποτελέσματα των πειραμάτων για μεγάλη απομακρυσμένη βάση με τοπικότητα στις συνδέσεις

Έτσι, οι τιμές των αρχικών μετρήσεων είναι πολύ αυξημένες σε σχέση με τις τιμές για τις μικρές βάσεις, ενώ οι μηχανισμοί κρυφής μνήμης δεν μπορούν να λειτουργήσουν αποτελεσματικά, καθώς το τμήμα της βάσης δεδομένων που χρησιμοποιείται σε κάθε πείραμα είναι πολύ μεγαλύτερο από τη χωρητικότητά τους.

Η λειτουργία της διάσχισης συνδέσμων κατά την αντίστροφη κατεύθυνση πραγματοποιείται αρκετά γρηγορότερα από τη λειτουργία κανονικής διάσχισης. Αυτό οφείλεται στο γεγονός ότι η λειτουργία αυτή χρησιμοποιεί μικρότερο τμήμα της βάσης, πράγμα που επιτρέπει στους μηχανισμούς κρυφής μνήμης να λειτουργήσουν αποδοτικότερα.

Οι τιμές για τη λειτουργία εισαγωγής δεδομένων είναι σχετικά κοντά στις τιμές για τις μικρές βάσεις δεδομένων. Το φαινόμενο αυτό εξηγείται από το γεγονός ότι η λειτουργία αυτή προκαλεί σχετικά λίγες προσπελάσεις στο δίσκο καθώς χρησιμοποιεί πολύ μικρό τμήμα της βάσης για την πραγματοποίησή της.

Η διαφορά των τιμών σε σχέση με τις μετρήσεις για τις μικρές βάσεις δεδομένων δεν πρέπει να μας προβληματίζει γιατί η τοπικότητα λογικών αναφορών είναι πολύ μικρή. Επίσης όπως θα δούμε παρακάτω οι μετρήσεις για άλλα συστήματα βάσεων δεδομένων όσον αφορά τη μεγάλη βάση δεδομένων κυμαίνονται στα ίδια επίπεδα. Πρέπει να παρατηρήσουμε επίσης ότι οι τιμές για τις μετρήσεις δεν έχουν δεκαπλασιαστεί με το δεκαπλασιασμό του μεγέθους της βάσης δεδομένων.

7.4 Σύγκριση των μετρήσεων με τα αποτελέσματα για άλλα συστήματα

Στο τμήμα αυτό παρουσιάζουμε τα αποτελέσματα των μετρήσεων για τις διάφορες λειτουργίες του προγράμματος αξιολόγησης επιδόσεων *OO1* που αναφέρονται στην εργασία [**?**] και αφορούν διάφορα συστήματα διαχείρισης βάσεων δεδομένων.

Οι μετρήσεις αυτές πραγματοποιήθηκαν χρησιμοποιώντας ένα Sun3/260 για σταθμό εργασίας και ένα Sun3/280 για σταθμό εξυπηρέτησης όπου βρισκόταν η απομακρυσμένη βάση δεδομένων. Η έκδοση του λειτουργικού συστήματος είναι η Sun O/S 4.0.3 και οι δίσκοι που χρησιμοποιήθηκαν είχαν μέγεθος 892 Mbytes και χρόνο αναζήτησης (seek time) 15 msecs. Έτσι, οι συγκρίσεις με τα αποτελέσματα για το σύστημα της TELOS δεν μπορούν να είναι απόλυτες, αφού χρησιμοποιούνται διαφορετικοί συνδυασμοί hardware, αλλά μπορούν απλά να είναι ενδεικτικές.

7.4.1 Μεγάλη απομακρυσμένη βάση

Οι συγκριτικές μετρήσεις για μεγάλη απομακρυσμένη βάση κατά την αρχική επανάληψη των λειτουργιών και την επανάληψη κατά την οποία οι μηχανισμοί κρυφής μνήμης παρουσιάζουν την καλύτερη απόδοση παρουσιάζονται στους πίνακες **??** και **??** αντίστοιχα.

Στους πίνακες παρουσιάζονται τρία συστήματα με τα ονόματα OODB, RDBMS και INDEX. Το σύστημα OODB αποτελεί ‘‘βήτα’’ έκδοση ενός οντοκεντρικού συστήματος διαχείρισης βάσεων δεδομένων. Το σύστημα RDBMS είναι ένα τυπικό εμπορικό σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων και το σύστημα INDEX είναι ένα σύστημα B^+ δέντρων αναζήτησης. Το σύστημα αυτό παρέχει αποδοτικές μεθόδους αναζήτησης και χρησιμοποιείται ως σύστημα αναφοράς, καθώς προσφέρει γρηγορότερη προσπέλαση και αναζήτηση από τα άλλα συστήματα. Αξίζει να αναφέρουμε ότι το σύστημα αυτό δεν παρέχει βασικές λειτουργίες συστημάτων διαχείρισης βάσεων δεδομένων όπως δοσοληψίες ή ερωτηματικές γλώσσες.

Στους πίνακες **??** και **??** μπορούμε να παρατηρήσουμε τις σημαντικά βελτιωμένες επιδόσεις του συστήματος για την TELOS για λειτουργίες διάσχισης και εισαγωγής δεδομένων αναφορικά με το σχεσιακό σύστημα. Οι σημαντικές διαφορές εξασφαλίζουν ότι ανεξάρτητα από τη διαφορά στα συστήματα όπου πραγματοποιήθηκαν οι μετρήσεις, το σύστημα TELOS έχει πολύ καλύτερες επιδόσεις από ένα τυπικό σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων.

Επίσης πρέπει να παρατηρήσουμε τις σημαντικά καλύτερες επιδόσεις του συστήματος για την TELOS για αντίστροφη διάσχιση συνδέσεων σε σχέση με το σύστημα INDEX, πράγμα που οφείλεται στο γεγονός ότι στο σύστημα TELOS η διάσχιση συνδέσμων πραγματοποιείται

Αρχικές μετρήσεις για μεγάλη απομακρυσμένη βάση για διάφορα συστήματα βάσεων δεδομένων				
Είδος μέτρησης	*TELOS*	OODB	RDBMS	INDEX
Αναζήτηση	46	49	49	47
Κανονική Διάσχιση	72	64	135	56
Αντίστροφη Διάσχιση	65	84	212	100
Εισαγωγή δεδομένων	17	10	24	20
Συνολικός Χρόνος	124	127	208	123

Πίνακας 7.6: Σύγκριση των αρχικών αποτελεσμάτων για διαφορετικά συστήματα βάσεων δεδομένων

Μετρήσεις για μεγάλη απομακρυσμένη βάση με χρήση κρυφής μνήμης σε διαφορετικά συστήματα				
Είδος μέτρησης	*TELOS*	OODB	RDBMS	INDEX
Αναζήτηση	26	43	24	18
Κανονική Διάσχιση	49	59	107	41
Εισαγωγή δεδομένων	11	7	24	18
Συνολικός Χρόνος	86	110	155	77

Πίνακας 7.7: Σύγκριση των αποτελεσμάτων για διαφορετικά συστήματα βάσεων δεδομένων, όταν έχουν γίνει κάποιες επαναλήψεις των πειραμάτων

εξίσου αποδοτικά και προς τις δύο κατευθύνσεις.

Τέλος, παρατηρώντας τις μετρήσεις για το σύστημα OODB και το σύστημα για την TELOS μπορούμε να διαπιστώσουμε την αδυναμία του συστήματος για την TELOS για πολύ γρήγορη επεξεργασία δοσοληψιών, όπως εξάλλου έχουμε παρατηρήσει και στις προηγούμενες μετρήσεις. Για τις άλλες μετρήσεις αν και τα αριθμητικά δεδομένα διαφέρουν, θεωρούμε ότι τα δύο συστήματα είναι ισοδύναμα, λόγω των διαφορετικών συστημάτων όπου πραγματοποιούνται οι μετρήσεις.

7.4.2 Μικρή βάση δεδομένων

Για την περίπτωση της μικρής βάσης δεδομένων, εκτός από τα συστήματα INDEX και RDBMS, παρουσιάζονται και τα αποτελέσματα για 4 εμπορικά οντοκεντρικά συστήματα διαχείρισης βάσεων δεδομένων. Τα συστήματα αυτά είναι τα: Objectivity/DB, Object Design Objectstore, Ontologic ONTOS και VERSANT^{TM}. Οι μετρήσεις για τα συστήματα αυτά πραγματοποιήθηκαν σε ελαφρώς διαφορετικά συστήματα από ότι οι μετρήσεις για τα

συστήματα INDEX και RDBMS και βρίσκονται επίσης στην εργασία [**?**]. Τα οντοκεντρικά συστήματα χαρακτηρίζονται στους πίνακες ως συστήματα OODB1 έως και OODB4 χωρίς να καθορίζεται η αντιστοιχία συστήματος βάσεων δεδομένων και μετρήσεων. Και στην περίπτωση αυτή θεωρούμε απλά ως ενδεικτικές τις μετρήσεις και δεν τις συγκρίνουμε αυστηρά, μια και έχουν πραγματοποιηθεί σε διαφορετικά συστήματα υπολογιστών από αυτό που εμείς χρησιμοποιήσαμε.

Στον πίνακα **??** παρουσιάζονται τα αποτελέσματα για την αρχική επανάληψη των λειτουργιών για μικρή απομακρυσμένη και μικρή τοπική βάση.

Αρχικές μετρήσεις για μικρή απομακρυσμένη βάση σε διάφορα συστήματα διαχείρισης βάσεων δεδομένων							
Είδος μέτρησης	*TELOS*	OODB1	OODB2	OODB3	OODB4	RDBMS	INDEX
Αναζήτηση	6.7	18	13	22	20	29	7.6
Κανονική Διάσχιση	10.5	27	13	18	17	94	17
Αντίστροφη Διάσχιση	12.4	34	13	16	22	95	23
Εισαγωγή δεδομένων	7.9	14	6.7	3.7	3.6	20	8.2
Συνολικός Χρόνος	25.2	59	33	44	41	143	18

Αρχικές μετρήσεις για μικρή τοπική βάση σε διάφορα συστήματα διαχείρισης βάσεων δεδομένων							
Είδος μέτρησης	*TELOS*	OODB1	OODB2	OODB3	OODB4	RDBMS	INDEX
Αναζήτηση	5.4	12	10	10	13	27	5.4
Κανονική Διάσχιση	6.1	18	10	8.3	9.8	90	13
Εισαγωγή δεδομένων	7.2	9	5.3	1.9	1.5	22	7.4
Συνολικός Χρόνος	18.7	39	25	20	24	139	26

Πίνακας 7.8: Σύγκριση των αρχικών αποτελεσμάτων για μικρή βάση

Η σύγκριση των αποτελεσμάτων για το σύστημα της TELOS και τα συστήματα RDBMS και INDEX είναι αντίστοιχη με τα αποτελέσματα για μεγάλη βάση όπου το σύστημα για την TELOS υπερτερεί σημαντικά του συστήματος RDBMS σε όλες τις λειτουργίες και παρουσιάζει σημαντικά καλύτερο χρόνο για την αντίστροφη διάσχιση από το σύστημα INDEX.

Συγκρίνοντας τώρα το σύστημα για την TELOS με τα οντοκεντρικά συστήματα, μπορούμε να πούμε ότι παρουσιάζει συγκρίσιμα αποτελέσματα με αυτά όσον αφορά τις λειτουργίες αναζήτησης και διάσχισης και προς τις δυο κατευθύνσεις, αλλά υστερεί όσον αφορά το χρόνο εισαγωγής δεδομένων στη βάση, όπως εξάλλου περιμέναμε.

Ο συνολικός χρόνος[1] για το σύστημα της TELOS είναι και στους δύο πίνακες μικρότερος από τους αντίστοιχους χρόνους για όλα τα οντοκεντρικά συστήματα, αλλά καθώς οι μετρήσεις δεν είναι άμεσα συγκρίσιμες δεν θα μπορούσαμε να ισχυριστούμε ότι το σύστημα για την TELOS παρουσιάζει καλύτερες επιδόσεις.

Τέλος, στον πίνακα **??** παρουσιάζονται οι μετρήσεις για τη μικρή τοπική βάση, αφού έχουν γίνει οι επαναλήψεις των λειτουργιών και έχουν λειτουργήσει αποδοτικά οι μηχανισμοί κρυφής μνήμης.

Μετρήσεις για μικρή τοπική βάση με χρήση κρυφής μνήμης σε διάφορα συστήματα διαχείρισης βάσεων δεδομένων							
Είδος μέτρησης	*TELOS*	OODB1	OODB2	OODB3	OODB4	RDBMS	INDEX
Αναζήτηση	0.6	0.1	0.03	1.1	1.0	19	2.4
Κανονική Διάσχιση	0.7	0.7	0.1	1.2	1.2	84	8.4
Εισαγωγή δεδομένων	4.5	3.7	3.1	1.0	2.9	20	7.5
Συνολικός Χρόνος	5.8	4.5	3.2	3.3	5.1	123	18

Πίνακας 7.9: Σύγκριση των αποτελεσμάτων για μικρή τοπική βάση με χρήση κρυφής μνήμης

Όπως φαίνεται και σε αυτόν τον πίνακα, το σύστημα για την TELOS παρουσιάζει σημαντικά καλύτερες επιδόσεις από το σύστημα RDBMS και παρόμοιες επιδόσεις με τα οντοκεντρικά συστήματα εκτός από τη λειτουργία εισαγωγής δεδομένων όπου τα άλλα οντοκεντρικά συστήματα παρουσιάζουν καλύτερους χρόνους. Αξίζει να παρατηρήσουμε τις μετρήσεις για αναζήτηση και διάσχιση για το σύστημα OODB2, οι οποίες είναι εντυπωσιακές σε σχέση με τις αντίστοιχες μετρήσεις για όλα τα άλλα συστήματα. Το σύστημα αυτό επιτυγχάνει πολύ χαμηλούς χρόνους προσπέλασης για οντότητες που βρίσκονται στη μνήμη χρησιμοποιώντας το hardware του μηχανισμού διαχείρισης εικονικής μνήμης του υπολογιστή όπως αναφέρεται στο [**?**]. Έτσι, κάθε προσπέλαση σε σελίδες που βρίσκονται στη κρυφή μνήμη του συστήματος αυτού πραγματοποιείται με μια μονάχα εντολή.

7.4.3 Απαιτούμενος αποθηκευτικός χώρος

Στον πίνακα **??** παρουσιάζουμε το μέγεθος της μικρής βάσης δεδομένων για όλα τα συστήματα που χρησιμοποιήθηκαν στο πρόγραμμα μέτρησης επιδόσεων.

Το σύστημα για την TELOS απαιτεί 18.4 Mbytes στο δίσκο, αποτελώντας έτσι το σύστημα που καταναλώνει περισσότερο χώρο, ενώ όλα τα υπόλοιπα συστήματα απαιτούν από 3.3 έως

[1] ως συνολικό χρόνο μετράμε μονάχα το άθροισμα των χρόνων για αναζήτηση, κανονική διάσχιση και εισαγωγή δεδομένων

7 Mbytes.

Απαιτούμενος χώρος για τη βάση δεδομένων (σε Mbytes)							
	TELOS	OODB1	OODB2	OODB3	OODB4	RDBMS	INDEX
Συνολικός Χώρος	18.4	5.6	3.4	7.0	3.7	4.5	3.3

Πίνακας 7.10: Απαιτούμενος χώρος για αποθήκευση της βάσης

Η διαφορά για το σύστημα της TELOS οφείλεται στο γεγονός ότι τα αρχεία αποθήκευσης του καταλόγου λογικών ονομάτων και των οντοτήτων καταναλώνουν σημαντικό χώρο. Κάθε οντότητα στη βάση έχει ένα λογικό όνομα με μήκος μέχρι 96 χαρακτήρες, και ο χώρος για λογικά ονόματα κρατείται και για γνωρίσματα που δεν έχουν λογικά ονόματα. Έτσι, το μέγεθος του αρχείου αποθήκευσης του καταλόγου λογικών ονομάτων για την μικρή βάση είναι 8.6 Mbytes. Αν αλλαχτεί το μέγεθος των λογικών ονομάτων θα επακολουθήσει αντίστοιχη μείωση του αρχείου αυτού. Επίσης, μπορούμε να αποφύγουμε την χρήση αποθηκευτικού χώρου για λογικά ονόματα σε γνωρίσματα, όπως προτείνουμε στο επόμενο κεφάλαιο, και να μειώσουμε έτσι το μέγεθος αυτού του αρχείου κατά 40 %. Στην περίπτωση αυτή αναμένεται αντίστοιχη μείωση και στα αρχεία που περιέχουν τον κατάλογο κατακερματισμού για τα λογικά ονόματα.

Οι δομές αναπαράστασης για τις οντότητες είναι αρκετά μεγαλύτερες από τις δομές που απαιτούνται στην εφαρμογή για την οποία πραγματοποιούνται οι μετρήσεις. Έτσι, το συνολικό μέγεθος του αρχείου που περιέχει τις οντότητες είναι 7.3 Mbytes. Αν χρησιμοποιούσαμε κατάλληλα μεγέθη για τις δομές αναπαράστασης των οντοτήτων για τη συγκεκριμένη εφαρμογή το συνολικό μέγεθος του αρχείου όπου αποθηκεύονται οι οντότητες θα μπορούσε να μειωθεί κατά 60 %. Έτσι, το συνολικό μέγεθος της βάσης θα κυμαινόταν μεταξύ 6.5 και 7.4 Mbytes πλησιάζοντας αρκετά τα μεγέθη για τα άλλα συστήματα. Τα μεγέθη για τις δομές αναπαράστασης οντοτήτων έχουν επιλεγεί ώστε να βελτιώνουν τις επιδόσεις του συστήματος για τυπικές εφαρμογές που αφορούν αναπαράσταση οντοτήτων λογισμικού και για το λόγο αυτό δεν τα μεταβάλλαμε κατά την διάρκεια των μετρήσεων επιδόσεων.

Κεφάλαιο 8

Συμπεράσματα και Μελλοντική Εργασία

8.1 Συμπεράσματα

Στα πλαίσια της εργασίας αυτής σχεδιάστηκαν και υλοποιήθηκαν οι δομές δεδομένων που χρησιμοποιούνται για την αναπαράσταση και αποθήκευση των δεδομένων για μια βάση περιγραφής λογισμικού που δημιουργείται χρησιμοποιώντας τη γλώσσα αναπαράστασης TELOS. Επίσης, σχεδιάστηκαν και υλοποιήθηκαν κατάλληλοι μηχανισμοί που επιτρέπουν την διαχείριση των δομών αυτών από διάφορες εφαρμογές.

Κατά τη σχεδίαση των μηχανισμών και των δομών δεδομένων μελετήθηκαν οι μηχανισμοί αποθήκευσης και διαχείρισης οντοτήτων σε διάφορα οντοκεντρικά συστήματα βάσεων δεδομένων. Στις περισσότερες περιπτώσεις, τα συστήματα αυτά χρησιμοποιούσαν σχεσιακά συστήματα αποθήκευσης, τα οποία λειτουργούν ανεξάρτητα από το μοντέλο αναπαράστασης δεδομένων των οντοκεντρικών συστημάτων βάσεων δεδομένων. Στο σύστημα για την TELOS δημιουργήσαμε δομές αποθήκευσης οι οποίες ανταποκρίνονται στους διάφορους τύπους οντοτήτων που υποστηρίζει η γλώσσα TELOS. Έτσι, το σύστημα που δημιουργείται είναι πολύ αποδοτικό κατά τη μεταφορά δεδομένων μεταξύ μνήμης και δίσκου, αφού χρησιμοποιούνται κοινές δομές για την οργάνωση των δεδομένων στη μνήμη και το δίσκο. Τα δεδομένα αποθηκεύονται σε δυαδικά αρχεία του λειτουργικού συστήματος UNIX, επιτρέποντας στο σύστημα για την TELOS να λειτουργεί σε ποικίλα υπολογιστικά συστήματα χωρίς να χρειάζεται ιδιαίτερες μετατροπές.

Για την δημιουργία και οργάνωση των μηχανισμών αναζήτησης που χρησιμοποιούνται χρειάστηκε να μελετηθούν διάφοροι μηχανισμοί αναζήτησης. Ο μηχανισμός γραμμικού κατακερματισμού που επιλέχθηκε για τον κατάλογο λογικών ονομάτων αποτελεί ένα μη-

χανισμό με υψηλές επιδόσεις που δεν επηρεάζεται από το μέγεθος της βάσης περιγραφής λογισμικού.

Η αρχική σχεδίαση του συστήματος περιείχε δυο διαφορετικά B-δένδρα αναζήτησης για τους υποκαταλόγους του καταλόγου λογικών ονομάτων. Τα δέντρα αυτά διατηρούνταν μονάχα στη μνήμη του υπολογιστή και δημιουργούνταν κατά την εκκίνηση του συστήματος. Έτσι η εκκίνηση απαιτούσε αρκετά δευτρόλεπτα για βάσεις που περιείχαν λίγες χιλιάδες οντότητες. Επίσης, οι διάφορες διεργασίες απαιτούσαν πολύ χώρο στη μνήμη για την εκτέλεσή τους μια και δεν υπήρχαν μηχανισμοί ενταμιευτών. Η τρέχουσα υλοποίηση επιτρέπει την άμεση λειτουργία του συστήματος ακόμα και για βάσεις που περιέχουν εκατοντάδες χιλιάδες οντότητες. Επίσης η διαγραφή ή εισαγωγή λογικών ονομάτων στους δύο υποκαταλόγους πραγματοποιείται σε σταθερό χρόνο που δεν εξαρτάται από το μέγεθος της βάσης.

Οι δομές αποθήκευσης που βρίσκονται στη μνήμη του υπολογιστή τοποθετούνται σε ενταμιευτές που διαθέτουν οι μηχανισμοί κρυφής μνήμης. Η δομική πληροφορία για τις διάφορες οντότητες περιλαμβάνεται στις δομές που τις αναπαριστούν και έτσι δεν χρειάζεται διατήρηση στη μνήμη πολύπλοκων δομών δεικτοδότησης που περιγράφουν το εννοιολογικό σχήμα της βάσης. Ακόμη, οι μηχανισμοί κρυφής μνήμης είναι παραμετρικοί, επιτρέποντας τη χρησιμοποίηση διαφορετικών αλγορίθμων αντικατάστασης, αλλά και τον ορισμό των μεγεθών για τους ενταμιευτές που χρησιμοποιούνται από κάθε μηχανισμό.

Το σύστημα διαχείρισης οντοτήτων χρησιμοποιεί το πρωτόκολλο κλειδώματος δύο φάσεων επιτρέποντας έτσι σε πολλές δοσοληψίες να λειτουργούν ταυτόχρονα. Ο μηχανισμός αυτός είναι σχετικά απλός και δεν επιβαρύνει τις επιδόσεις του συστήματος κατά τη λειτουργία του.

8.2 Βελτιώσεις

Στην παράγραφο αυτή προτείνουμε διάφορες βελτιώσεις για το σύστημα διαχείρισης οντοτήτων, οι οποίες μπορούν να υλοποιηθούν σχετικά εύκολα και μπορούν να βοηθήσουν στη βελτίωση των χρονικών επιδόσεων του συστήματος.

Η πρώτη πρόταση αφορά τον κατάλογο συστήματος. Στην τρέχουσα υλοποίηση, η ανάθεση αναγνωριστικών συστήματος για τις διάφορες οντότητες που δημιουργούνται, πραγματοποιείται σειριακά. Δηλαδή, κάθε φορά που δημιουργείται μια νέα οντότητα, της παραχωρείται το πρώτο αναγνωριστικό συστήματος που βρίσκεται στη λίστα ελεύθερων εγγραφών του καταλόγου συστήματος. Έτσι, κάθε τμήμα του καταλόγου συστήματος περιέχει εγγραφές που αντιστοιχούν σε οντότητες όλων των τύπων οντοτήτων. Εναλλακτικά,

θα μπορούσαμε να δεσμεύσουμε κάθε τμήμα του καταλόγου συστήματος ώστε να περιέχει εγγραφές που αντιστοιχούν σε ένα τύπο οντοτήτων μονάχα. Η πολιτική αυτή δημιουργεί μια ομαδοποίηση των εγγραφών του καταλόγου συστήματος για τους τύπους δεδομένων και βελτιώνει τις επιδόσεις του συστήματος για ερωτήσεις διάσχισης που αποτελούν τις πιο συχνές ερωτήσεις στη βάση παράστασης λογισμικού.

Η επόμενη πρόταση αφορά τον κατάλογο λογικών ονομάτων. Πολλές φορές κατά την εισαγωγή γνωρισμάτων στη βάση δεδομένων, τα γνωρίσματα δεν έχουν λογικά ονόματα. Στις περιπτώσεις αυτές, το σύστημα δημιουργεί λογικά ονόματα για τα γνωρίσματα και τα αποθηκεύει στις δομές του καταλόγου λογικών ονομάτων. Τα γνωρίσματα αυτά συνήθως έχουν απλή δομή και αναφέρονται μονάχα σε δυο οντότητες, την οντότητα στην οποία ορίζονται και την οντότητα την οποία προσδιορίζουν. Τα γνωρίσματα αυτά χρησιμοποιούνται μονάχα από το σύστημα και δεν ενδιαφέρουν ποτέ τους χρήστες, οπότε δεν είναι απαραίτητη η αποθήκευση των λογικών τους ονομάτων στον κατάλογο λογικών ονομάτων. Έτσι, μπορεί να μειωθεί σημαντικά το μέγεθος του υποκαταλόγου λογικών ονομάτων, αλλά όχι και το μέγεθος του υποκαταλόγου αναγνωριστικών συστήματος για τον κατάλογο λογικών ονομάτων, που περιέχει εγγραφές για κάθε αναγνωριστικό συστήματος που χρησιμοποιείται. Μπορεί όμως να διαχωριστεί ο κατάλογος συστήματος σε δύο διαφορετικούς υποκαταλόγους, έναν για τα γνωρίσματα που δεν έχουν λογικό όνομα και έναν για τις υπόλοιπες οντότητες. Στην περίπτωση αυτή ο υποκατάλογος αναγνωριστικών συστήματος θα περιέχει εγγραφές μονάχα για τις υπόλοιπες οντότητες. Επίσης, μπορεί να δημιουργηθεί ένας ειδικός τύπος οντότητας αποθήκευσης που θα χρησιμοποιείται αποκλειστικά για γνωρίσματα χωρίς λογικό όνομα. Ο τύπος αυτός θα έχει μέγεθος πολύ μικρότερο από τις κανονικές δομές αποθήκευσης για γνωρίσματα απαιτώντας πολύ λιγότερο αποθηκευτικό χώρο στο δίσκο. Έχουμε παρατηρήσει ότι σε τυπικές εφαρμογές της TELOS περισσότερο από 40 % των οντοτήτων που αποθηκεύονται είναι γνωρίσματα χωρίς λογικό όνομα. Εφαρμόζοντας λοιπόν τις αλλαγές που μόλις αναφέραμε αναμένουμε σημαντική βελτίωση των επιδόσεων του συστήματος, αλλά και μείωση του μεγέθους της βάσης δεδομένων που δημιουργείται στο δίσκο.

Η τελευταία από τις προτεινόμενες βελτιώσεις αφορά τις μεγάλες οντότητες αποθήκευσης. Κάθε προσπέλαση σε τέτοιες οντότητες οδηγεί σε ανάκληση ολόκληρης της δομής αναπαράστασης για την οντότητα αυτή. Η δομή αναπαράστασης σε τέτοιες περιπτώσεις περιλαμβάνει τη δομή αναπαράστασης που αντιστοιχεί στον τύπο της οντότητας και το σύνολο των δομών επεκτάσεων που χρησιμοποιούνται για την επιπλέον πληροφορία. Αρκετές φορές όμως, όλη η πληροφορία που χρειάζεται από κάποια εφαρμογή βρίσκεται στην δομή αναπαράστασης για τον τύπο της οντότητας. Θα μπορούσε λοιπόν να υλοποιηθεί ένας μηχανισμός ο οποίος σε κάθε λογική αναφορά σε μεγάλες οντότητες θα μεταφέρει στη μνήμη

μονάχα τη δομή αναπαράστασης για τον τύπο της οντότητας, ενώ οι δομές επεκτάσεων θα μεταφέρονται στη μνήμη μονάχα όταν απαιτείται από την εφαρμογή. Με το μηχανισμό αυτό ελαττώνεται το πλήθος των προσπελάσεων στο δίσκο για μεγάλες οντότητες, σε βάσεις που περιέχουν μεγάλο πλήθος οντοτήτων, βελτιώνοντας το χρόνο απόκρισης του συστήματος σε ερωτήσεις που αφορούν τέτοιες οντότητες.

8.3 Μελλοντική εργασία

Στην παράγραφο αυτή προτείνουμε κάποια θέματα για μελλοντική εργασία που αναφέρονται στο σύστημα διαχείρισης οντοτήτων και μπορούν να οδηγήσουν σε ένα πλήρες σύστημα με μεγάλη λειτουργικότητα. Τα θέματα αυτά είναι τα εξής:

- Δημιουργία ενός μηχανισμού καταγραφής μεταβολών στη βάση

  Ο μηχανισμός αυτός μπορεί να παρέχει τη δυνατότητα ανάκλησης προηγούμενων καταστάσεων της βάσης. Επίσης μπορεί να χρησιμοποιηθεί για επανάκτηση της βάσης δεδομένων μετά από πιθανές βλάβες στο σύστημα.

- Δημιουργία καλύτερου μηχανισμού ταυτόχρονης προσπέλασης στη βάση από πολλές δοσοληψίες.

  Ο μηχανισμός αυτός μπορεί να συνδυαστεί με αρχιτεκτονική σταθμού εξυπηρέτησης - πελατών για το σύστημα διαχείρισης οντοτήτων. Ο σταθμός εξυπηρέτησης σε μια τέτοια περίπτωση θα είναι η μόνη διεργασία, η οποία θα έχει πρόσβαση στα δεδομένα της βάσης στο δίσκο και η οποία θα συντονίζει τις διάφορες δοσοληψίες. Οι διεργασίες πελατών θα μπορούν να επικοινωνούν με τη διεργασία εξυπηρέτησης μέσω κλήσεως συναρτήσεων (remote procedure calls) και να λαμβάνουν τα δεδομένα της βάσης από τους μηχανισμούς κρυφής μνήμης της διεργασίας εξυπηρέτησης.

- Ομαδοποίηση δεδομένων της βάσης (object clustering)

  Η χρήση ενός μηχανισμού ομαδοποίησης για τα δεδομένα της βάσης μπορεί να βελτιώσει σημαντικά τις επιδόσεις του συστήματος.

- Μηχανισμός προανάκλησης δεδομένων της βάσης (prefetch mechanism)

  Ο μηχανισμός αυτός πρέπει να λαμβάνει υπόψη τη συμπεριφορά των διαφόρων εφαρμογών όσον αφορά τις προσπελάσεις σε δεδομένα που αυτές πραγματοποιούν. Ο μηχανισμός αυτός μπορεί να καθοδηγείται από στατιστικά δεδομένα για το φόρτο εργασίας του συστήματος ή να παίρνει συμβουλές από τους χρήστες και μπορεί να

συνδυαστεί με επιλογή διαφορετικών αλγορίθμων αντικατάστασης για τους επιμέρους μηχανισμούς κρυφής μνήμης, όταν πραγματοποιούνται διαφορετικές πράξεις στη βάση.

8.4 Επίλογος

Η TELOS είναι μια γλώσσα αναπαράστασης γνώσης που χρησιμοποιεί ποικίλους μηχανισμούς δόμησης από το χώρο των σημασιολογικών δικτύων. Στην παρούσα εργασία δημιουργήσαμε κατάλληλους μηχανισμούς για την αναπαράσταση και διαχείριση των δεδομένων που αποθηκεύονται σε μια βάση παράστασης λογισμικού που περιγράφεται στη γλώσσα TELOS.

Ο χώρος που καταλαμβάνει η βάση αναπαράστασης λογισμικού είναι πολύ μεγαλύτερος από το χώρο που καταλαμβάνουν βάσεις που δημιουργούνται από διαφορετικά συστήματα διαχείρισης βάσεων δεδομένων. Η φαινομενική σπατάλη χώρου όμως βοηθάει στη βελτίωση των επιδόσεων του συστήματος, καθώς οι ερωτήσεις διάσχισης μπορούν να πραγματοποιηθούν εξίσου αποδοτικά και προς τις δυο κατευθύνσεις των συνδέσεων μεταξύ οντοτήτων της βάσης. Επίσης η εκκίνηση του συστήματος είναι άμεση, ανεξάρτητα από το μέγεθος της βάσης, γιατί δεν χρειάζεται να δημιουργηθεί καμμία δομή αναζήτησης, αφού όλες οι απαραίτητες δομές βρίσκονται στο δίσκο.

Οι μηχανισμοί διαχείρισης δεδομένων της βάσης παρουσιάζουν σταθερή απόδοση που δεν εξαρτάται από το μέγεθος της βάσης δεδομένων. Οι έλεγχοι για τις επιδόσεις του συστήματος έδειξαν ότι το σύστημα μπορεί να συναγωνιστεί εμπορικά συστήματα διαχείρισης βάσεων δεδομένων ξεπερνώντας σημαντικά τις επιδόσεις των συστημάτων που χρησιμοποιούν το σχεσιακό μοντέλο για αναπαράσταση δεδομένων.

Παράρτημα Α

Η συνάρτηση μετασχηματισμού $g(K)$ για τα λογικά ονόματα

Στο παράρτημα αυτό παρουσιάζουμε τη συνάρτηση μετασχηματισμού για τα λογικά ονόματα.

Η συνάρτηση αυτή μετασχηματίζει όλους τους χαρακτήρες του λογικού ονόματος όταν το λογικό όνομα έχει μήκος μικρότερο από 13 χαρακτήρες.

Διαφορετικά, μετασχηματίζει δειγματοληπτικά 11 από τους χαρακτήρες του λογικού ονόματος. Η δειγματοληψία εξαρτάται από το μήκος του λογικού ονόματος και από το αν αυτό αποτελείται από άρτιο ή περιττό αριθμό χαρακτήρων.

```
# define PRIME1    37
# define PRIME2    1048583

int g(char *k)
{
  register int h = 0;  /* η τιμή g(k) */
  register int l = 0;  /* μέγεθος του λογικού ονόματος */
  int step = 0;        /* αριθμός των βημάτων κατά τις
                        * επαναλήψεις των βρόγχων */

  /* υπολογισμός του πλήθους των γραμμάτων στο λογικό όνομα */
  while (k[l++] != '\0');

  /* αν το λογικό όνομα έχει περισσότερους από 13 χαρακτήρες,
   * τότε δεν χρησιμοποιούνται όλοι για τον υπολογισμό του g(k) */
  if ( l > 13 ){
    step = (l / 11) +1;

    /* αν το μέγεθος είναι περιττός αριθμός */
    if ( l & 1 ){
      for ( int i = 0; i < (l/step) ; i++ ){
        /* μετασχηματισμός κάποιου γράμματος από την αρχή του k*/
        h = h * PRIME1 + (k[i*step] - ' ');
        /* μετασχηματισμός κάποιου γράμματος από το τέλος του k*/
        h = h * PRIME1 + (k[l - (i*step) -2] - ' ');
      }
    }

    /* αν το μέγεθος είναι άρτιος αριθμός */
    else {
      for ( int i = 0; i < (l/step) ; i++ ){
        /* μετασχηματισμός κάποιου γράμματος από την αρχή του k*/
        h = h * PRIME1 + (k[i*step] - ' ');
        /* μετασχηματισμός κάποιου γράμματος από το τέλος του k*/
        h = h * PRIME1 + (k[l - (i*step) -1] - ' ');
      }
    }
  }

  /* αν το μήκος του λογικιού ονόματος είναι μικρότερο απο 13 χαρακτήρες
   * τότε χρησιμοποιούνται όλοι για τον υπολογισμό του g(k) */
  else {
    l = 0;
    while (k[l] != '\0') h = h * PRIME1 + (k[l++] - ' ');
    for ( ; l < 11; l++ )
      h = h * PRIME1;
  }
  /* πάντα επιστρέφουμε θετική τιμή για το g(k) */
  return (( h > 0 ) ? h: -h);
}
```

Παράρτημα Β

Αναλυτικές μετρήσεις για τη μικρή βάση δεδομένων

Στο παράρτημα αυτό παρουσιάζονται πλήρεις μετρήσεις για την μικρή βάση δεδομένων για τις λειτουργίες αναζήτησης και διάσχισης συνδέσμων. Η βάση δεδομένων περιλαμβάνει συνολικά 80.000 οντότητες από τις οποίες 20.000 αποτελούν οντότητες τύπου *τμήμα* και 60.000 αποτελούν οντότητες τύπου *σύνδεσμος*.

Οι τιμές που αναφέρονται αφορούν μία μονάχα πραγματοποίηση των λειτουργιών. Η πρώτη μέτρηση αφορά την αρχική επανάληψη. Η γραμμή με την ονομασία *Μέσος όρος* παρουσιάζει τον μέσο όρο για τις υπόλοιπες 9 επαναλήψεις της λειτουργίας. Η τιμή αυτή είναι ενδεικτική της ταχύτητας με την οποία επιτυγχάνεται η καλύτερη μέτρηση. Έτσι, όσο πιο κοντά βρίσκεται η τιμή αυτή στην βέλτιστη τιμή που παρουσιάζεται για τις επαναλήψεις, τόσο γρηγορότερα επιτυγχάνεται η βέλτιστη τιμή.

Για τη λειτουργία της διάσχισης συνδέσμων κατά την αντίστροφη φορά παρουσιάζονται δύο διαφορετικές μετρήσεις. Η δεύτερη μέτρηση αφορά διαφορετική οργάνωση της βάσης δεδομένων. Στην περίπτωση αυτή η φορά των συνδέσμων έχει αντιστραφεί. Η μέτρηση αυτή πραγματοποιείται για να δείξει ότι το σύστημα επιτυγχάνει εξίσου αποδοτική διάσχιση συνδέσμων και προς τις δύο κατευθύνσεις.

Β.1 Τοπική βάση δεδομένων

Στον πίνακα **??** παρουσιάζονται οι μετρήσεις για τοπική βάση δεδο μένων. Έτσι η βάση δεδομένων βρίσκεται στον τοπικό δίσκο ενός σταθμού εργασίας Sun SPARCstation ELC (4/25) και οι μετρήσεις πραγματοποιούνται στον ίδιο σταθμό εργασίας.

Χρόνοι (σε δευτερόλεπτα) για την τοπική βάση δεδομένων				
	Λειτουργίες			
Μέτρηση	Αναζήτηση	Διάσχιση συνδέσμων		
		Κανονική	Αντίστροφη	Αντίστροφη (2)
1η	6.767	10.561	12.382	14.914
2η	0.891	3.842	6.058	4.723
3η	0.823	2.713	4.706	3.877
4η	0.784	2.517	3.519	3.757
5η	0.771	2.104	5.419	2.697
6η	0.741	1.358	3.844	1.813
7η	0.724	1.834	4.618	1.612
8η	0.681	1.328	3.556	1.546
9η	0.674	1.278	2.960	1.244
10η	0.644	1.120	1.576	1.075
Μέσος όρος	0.748	2.010	4.028	1.075

Πίνακας Β.1: Μετρήσεις για την τοπική βάση δεδομένων

Β.2 Απομακρυσμένη βάση δεδομένων

Στον πίνακα **??** παρουσιάζονται οι μετρήσεις για την απομακρυσμέ νη βάση δεδομένων. Οι μετρήσεις πραγματοποιούνται σε ένα σταθμό εργασίας Sun SPARCstation ELC (4/25), ενώ τα δεδομένα της βάσης βρίσκονται στον τοπικό δίσκο ενός σταθμού εργασίας Sun SS-10.

Χρόνοι (σε δευτερόλεπτα) για την απομακρυσμένη βάση δεδομένων				
	Λειτουργίες			
Μέτρηση	Αναζήτηση	Διάσχιση συνδέσμων		
		Κανονική	Αντίστροφη	Αντίστροφη (2)
1η	6.767	10.561	12.382	14.914
2η	0.891	3.842	6.058	4.723
3η	0.823	2.713	4.706	3.877
4η	0.784	2.517	3.519	3.757
5η	0.771	2.104	5.419	2.697
6η	0.741	1.358	3.844	1.813
7η	0.724	1.834	4.618	1.612
8η	0.681	1.328	3.556	1.546
9η	0.674	1.278	2.960	1.244
10η	0.644	1.120	1.576	1.075
Μέσος όρος	0.748	2.010	4.028	2.483

Πίνακας Β.2: Μετρήσεις για την απομακρυσμένη βάση δεδομένων

Βιβλιογραφία

[1] M. J. Bach. *The Design of the UNIX Operating System*, chapter System Calls for the File System, pages 91--140. Prentice Hall, 1986.

[2] R. A. Baeza-Yates and Per-Ake Larson. Analysis of B^+-trees with Partial Expansions. Technical Report CS-87-04, University of Waterloo, Computer Science Department, February 1987.

[3] R. Bayer and E.M. McCreight. Organization and Maintenance of large ordered indices. *Acta Informatica*, 1(3), 1972.

[4] E. Bertino and L. Martino. Object-Oriented Database Management Systems: Concepts and Issues. *IEEE Computer*, 24(3), April 1991.

[5] P. Butterworth, A. Otis, and J. Stein. The GemStone Object Database Management System. *Communications of the ACM*, 34(10), 1991.

[6] M. J. Carey, D. J. DeWitt, J. E. Richardson, and E. J. Shekita. Storage Management for Objects in EXODUS. In *Object-Oriented Concepts, Databases, and Apllications*, pages 341--369. ACM Press, 1989.

[7] R. G. G. Cattell and J. Skeen. Object Operations Benchmark. *ACM Transactions on Database Systems*, 17(1), January 1992.

[8] H-T Chou, D. J. Dewitt, R. H. Katz, and A. C. Klug. Design and Implementation of the Wisconsin Storage System. *Software-Practice and Experience*, 15(10), October 1985.

[9] Jiang Hsing Chu and Gary D. Knott. An Analysis of B-trees and their Variants. Technical Report CS-TR-238, University of Maryland, November 1986.

[10] D. Comer. The Ubiquitous B-Tree. *ACM Computing Surveys*, 11(2), 1979.

[11] C. Dadouris, M.Doerr, S.Kizlaridou, D.Mauroidis, E.Pataki, M.Theodorakis, M.Theodoridou, and G.Yeorgiannakis. Implementation of the SIB System. Technical Report ITHACA.FORTH.92.E2.#7, CSI-FORTH, Jan 24 1992.

[12] C. J. Date. *Relational Database : selected writings*, chapter 7, pages 178--185. Reading Mass: Addison-Wesley Pub. Co., 1986.

[13] M. Doerr, P. Klimathianakis, and M. Theodoridou. SIB Data Entry Language User's Manual. Technical report, Institute of Computer Science, FORTH, December 1992.

[14] W. Effelsberg and N. Goodman. Principles of Database Buffer Management. *ACM Transactions on Database Systems*, 9(4), December 1984.

[15] R. J. Enbody and H. C. Du. Dynamic Hashing Schemes. *ACM Computing Surveys*, 20(2), June 1988.

[16] O. Deux et al. The Story of O_2. *IEEE Trans. on Knowledge and Data Engineering*, 2(1), March 1990.

[17] O. Deux et al. The O_2 System. *Communications of the ACM*, 34(10), October 1991.

[18] D. H. Fishman, J. Annevelink, E. Chow, T. Connors, J. W. Davis, W. Hasan, C. G. Hoch, W. Kent, S. Leichner P. Lyngbaek, B. Mahbod, M. A. Neimat, T. Risch, M. C. Shan, and W. K. Wilkinson. Iris: An Object-Oriented Database Management System. *ACM Transactions on Office Information Systems*, 5(1), January 1987.

[19] A. R. Hurson, S.H. Pakzad, and J.J Cheng. Object-Oriented Database Management Systems: Evolution and Performance Issues. *IEEE Computer*, 26(2), February 1993.

[20] *An Introduction to ObjectStore, Release 1.0.* Object Design Inc., Burlington, Massachusets, 1989.

[21] W. Kim. Object-Oriented Databases: Definition and Research Directions. *IEEE Transactions on Knowledge and Data enginering*, 2(3), September 1991.

[22] W. Kim, Jorge, F. Garza, Nathaniel Ballou, and Darrell Woelk. Architecture of the ORION Next-Generation Database System. *IEEE Trans. on Knowledge and Data Engineering*, 2(1), March 1990.

[23] M. Koubarakis, J. Mylopoulos, M. Stanley, and A. Borgida. Telos: Features and Formalization. Technical Report FORTH/CSI/TR/1989/018, CSI-FORTH, 1989.

[24] C. Lamb, G. Landis, and D. Weinreb. The ObjectStore Database System. *Communications of the ACM*, 34(10), October 1991.

[25] Per-Ake Larson. Dynamic Hash Tables. *Communications of the ACM*, 31(4), April 1988.

[26] W. Litwin. Virtual Hashing: a dynamically changing hashing. In *Proceedings of the 4th VLDB Conference*, pages 517 -- 523, 1978.

[27] W. Litwin. Linear Hashing: a new tool for file and table addressing. In *Proceedings of the 6th VLDB Conference*, pages 212 -- 223, 1980.

[28] D. B. Lomet. A Simple Bounded Disorder File Organization with Good Performance. Technical Report TR-86-13, Wang Institute of Graduate Studies, 1986.

[29] J. E. B. Moss. Design of the Mneme Persistent Object Store. *ACM Transactions on Information Systems*, 8(2), April 1990.

[30] K. Ntadouris. Programmatic Query Interface and Query Processing for the TELOS Language. Master's thesis, University Of Crete, 1993.

[31] *Ontos Release 2.0 Data Sheet*. Ontologic Inc., Burlington, Massachusets, 1990.

[32] M. V. Ramachrisna and Per Ake Larson. File Organization Using Composite Perfect Hashing.

[33] E. Shekita and M. Zwilling. Cricket: A Mapped, Persistent Object Store. Technical report, Computer Sciences Department, University of Wisconsin, August 1990.

[34] A. Silberschatz and J. L. Peterson. *Operating System Concepts (Alternate Edition)*, chapter Virtual Memory, pages 277--283. Addison Wesley Publishing Company, 1988.

[35] A. J. Smith. Sequentiallity and Prefething in Database Systems. *ACM Transactions on Database Systems*, 3(3), 1978.

[36] A. J. Smith. Cache Memories. *ACM Computing Surveys*, 14(3), March 1982.

[37] M. Stonebraker. The Design of the POSTGRES Storage System. In *Proceedings of the 13th VLDB Conference, Brighton*, 1987.

[38] M. Stonebraker and G. Kemnitz. The POSTGRES Next-Generation Database Management System. *Communications of the ACM*, 34(10), October 1991.

[39] M. Stonebraker, L. A. Rowe, and M. Hirohama. The Implementation of POSTGRES. *IEEE Transactions on Knowledge and Data Engineering*, 2(1), January 1990.

[40] Bjarne Stroustroup. *The C++ Programming Language*. Addison-Wesley, Reading, Massachusets, 1991.

[41] *Versant Product Profile*. Versant Object Technology, Menlo Park, California, 1990.

[42] K. Wilkinson, P. Lyngbaek, and W. Hasan. The Iris Architecture and Implementation. *IEEE Transactions on Knowledge and Data enginering*, 2(1), March 1990.