ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Ευρετηρίων για Σύνθετα Δεδομένα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΧΡΙΣΤΙΝΑΣ Β. ΚΑΣΚΟΥΡΑ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2006

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Ευρετηρίων για Σύνθετα Δεδομένα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΧΡΙΣΤΙΝΑΣ Β. ΚΑΣΚΟΥΡΑ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 12η Ιουλίου 2006.

.....	.....	.....
Τιμολέων Σελλής	Ιωάννης Βασιλείου	Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επικ. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2006

.....

ΧΡΙΣΤΙΝΑ Β. ΚΑΣΚΟΥΡΑ

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

Περίληψη

Ο στόχος της διπλωματικής αυτής εργασίας είναι η ανάπτυξη ενός ευρετηρίου το οποίο θα είναι αποδοτικό για χρήση σε σύνθετα δεδομένα και συγκεκριμένα για τιμές-σύνολα, δηλαδή για δοσοληψίες, η κάθε μία από τις οποίες αποτελείται από ένα σύνολο (set) από ίδιου τύπου δεδομένα. Το ευρετήριο που αναπτύσσουμε μας ενδιαφέρει να μπορεί να απαντάει σε συγκεκριμένα ερωτήματα, τα οποία είναι subset queries, equality queries και superset queries. Έτσι, για την ανάπτυξη του ευρετηρίου μας χρησιμοποιούμε το πιο αποδοτικό από τα ήδη υπάρχοντα ευρετήρια, το inverted file, το οποίο συνδυάζουμε με το γνωστό Β-Δέντρο, δημιουργώντας έτσι το ordered inverted file. Ο στόχος που επιθυμούμε να πετύχουμε με την ανάπτυξη του ordered inverted file είναι να κάνουμε πιο αποδοτική την αποτίμηση των ερωτημάτων, αποκτώντας μέσω του Β-Δέντρου πρόσβαση και σε άλλα σημεία των λιστών του inverted file εκτός από την αρχή τους. Αναπτύσσεται επίσης κώδικας σε C++ ο οποίος υλοποιεί την κατασκευή του ordered inverted file καθώς και την αποτίμηση ερωτημάτων με χρήση αυτού, και ο οποίος χρησιμοποιείται για τη διενέργεια πειραμάτων που συγκρίνουν την απόδοση του ordered inverted file με αυτή του απλού inverted file. Η υλοποίησή μας αποθηκεύει τα Β-Δέντρα στο σκληρό δίσκο ενώ για το inverted file τμήμα του ευρετηρίου προσφέρει την επιλογή να αποθηκευτεί είτε στο δίσκο είτε στην κύρια μνήμη. Τα πειράματα που έγιναν με χρήση του κώδικα αυτού δείχνουν ότι σε γενικές γραμμές το ordered inverted file είναι πιο αποδοτικό από το inverted file, ειδικά για την αποτίμηση ερωτημάτων equality και superset. Για την αποτίμηση subset queries κατά την οποία η απόδοση του ordered inverted file δε φάνηκε να υπερτερεί σημαντικά αυτής του απλού inverted file προτείνονται επιπλέον μέθοδοι βελτίωσης, οι οποίες όμως δε συμπεριλαμβάνονται στην υλοποίηση.

Λέξεις Κλειδιά: δομές δεικτοδότησης, ευρετήριο, ανεστραμμένο αρχείο, διατεταγμένο ανεστραμμένο αρχείο, ερωτήματα υποσυνόλου, ερωτήματα ισότητας, ερωτήματα υπερσυνόλου

Abstract

The goal of this thesis is to develop an index which will be efficient to use with set-valued attributes. With the term set-valued attributes we mean transactions, each one of which comprises of a set of data of the same type. The index which will be developed must be able to answer efficiently certain types of queries, and more specifically subset, equality and superset queries. To develop our index we combine the already existing inverted file, which is the most efficient index for set-valued attributes, with the B-Tree, therefore creating the ordered inverted file. This way, we manage to answer the aforementioned queries more efficiently than by using the simple inverted file, since the B-Tree grants us access to more entry points in the inverted file's lists than just the beginning of each list. We also implement the construction and query evaluation of the ordered inverted file in C++, in order to be able to run experiments that will compare the efficiency of the ordered inverted file to that of the simple inverted file. Our implementation stores the B-Trees in the hard disk, while giving the user a choice of whether to store the inverted file portion of the index in the hard disk or in the main memory. The experiments we ran using that code, showed that the ordered inverted file is generally more efficient than the simple inverted file, especially in the evaluation of equality and superset queries. For the evaluation of subset queries, where the ordered inverted file is not significantly more efficient than the simple inverted file, we also propose new methods to enhance its efficiency, whose implementation however is not part of this thesis.

Keywords: indexing structures, index, inverted file, ordered inverted file, subset queries, equality queries, superset queries

Πίνακας περιεχομένων

1

Εισαγωγή

Στο πρώτο αυτό κεφάλαιο εξετάζουμε το αντικείμενο με το οποίο ασχολείται η παρούσα διπλωματική, αναλύοντας κάποιες βασικές έννοιες που θα χρησιμοποιηθούν ευρέως στα επόμενα κεφάλαια της διπλωματικής. Επίσης, παραθέτουμε ένα σύντομο οδηγό των κεφαλαίων που ακολουθούν και των περιεχομένων τους, με στόχο τη διευκόλυνση του αναγνώστη.

1.1 Αντικείμενο της διπλωματικής

Ο στόχος της διπλωματικής αυτής εργασίας είναι η ανάπτυξη ενός ευρετηρίου το οποίο θα είναι αποδοτικό για χρήση σε σύνθετα δεδομένα και συγκεκριμένα για τιμές-σύνολα. Με τον όρο τιμές-σύνολα (set-valued attributes) εννοούμε δοσοληψίες, η κάθε μία από τις οποίες αποτελείται από ένα σύνολο (set) από ίδιου τύπου δεδομένα. Το γεγονός ότι μιλάμε για σύνολα δεδομένων σημαίνει αφ' ενός ότι τα δεδομένα που απαρτίζουν μία δοσοληψία είναι αταξινόμητα και αφ' ετέρου ότι το κάθε δεδομένο - αντικείμενο μπορεί να εμφανιστεί το πολύ μία φορά σε κάθε δοσοληψία. Έτσι, η κάθε δοσοληψία t η οποία περιέχεται μέσα στη βάση μας (έστω D), έχει τη μορφή t = [id, s], όπου το id είναι ένα (μοναδικό) αναγνωριστικό και το s είναι το σύνολο των αντικειμένων που περιέχονται στη δοσοληψία και τα οποία προέρχονται όλα από ένα λεξικό (vocabulary), έστω V. Ένα παράδειγμα τέτοιων δοσοληψιών φαίνεται στον πίνακα 1.1.

ID	Items
1	a, c, d, f
2	a, g, f
3	a, b, c, d
4	a, c, e, f
5	e, f, g
6	b, c, e, f

Πίνακας 1.1: Παράδειγμα δεδομένων με τιμές – σύνολα

Τέτοιου είδους δοσοληψίες μπορούν να βρουν πολλές εφαρμογές στις σύγχρονες βάσεις δεδομένων. Έτσι, με ένα τέτοιο σχήμα μπορεί να παρασταθεί ένα σύνολο από συναλλαγές σε κάποιο κατάστημα λιανικής πώλησης (π.χ. supermarket), όπου το κάθε αντικείμενο συμβολίζει κάποιο προϊόν, ή ένα σύνολο από χημικές ενώσεις όπου το κάθε αντικείμενο συμβολίζει κάποιο χημικό στοιχείο που περιέχεται στην ένωση.

Το ευρετήριο το οποίο θα αναπτυχθεί θα θεωρείται αποδοτικό εφόσον μπορεί να απαντά γρήγορα στα τρία συνηθέστερα είδη ερωτημάτων που γίνονται στον τύπο αυτό δεδομένων. Τα ερωτήματα αυτά είναι subset queries, equality queries και superset queries. Για το κάθε ερώτημα ο χρήστης ορίζει εκτός από τον τύπο του ερωτήματος και ένα σύνολο από αντικείμενα ερωτήματος (query set) έστω qs το οποίο προέρχεται από το ίδιο λεξικό με τα αντικείμενα που περιέχονται στη βάση. Τα subset queries (ερωτήματα υποσυνόλου) είναι ερωτήματα στα οποία ψάχνουμε να βρούμε όλες τις εγγραφές οι οποίες περιέχουν το σύνολο των αντικειμένων που ορίζεται από το ερώτημα, δηλαδή τις $\{t \mid t \in D \wedge qs \subseteq t.s\}$. Στα equality queries (ερωτήματα ισότητας) ψάχνουμε να βρούμε τις εγγραφές που περιέχουν ακριβώς το δοθέν σύνολο αντικειμένων χωρίς να περιέχουν επιπλέον δεδομένα, δηλαδή τις $\{t \mid t \in D \wedge qs \equiv t.s\}$. Τέλος, με τα superset queries (ερωτήματα υπερσυνόλου) ψάχνουμε να βρούμε τις εγγραφές που περιέχουν μόνο αντικείμενα που ανήκουν στο δοθέν σύνολο, αλλά όχι κατ' ανάγκην όλα, δηλαδή τις $\{t \mid t \in D \wedge qs \supseteq t.s\}$.

Έτσι, στο παράδειγμα της βάσης χημικών ενώσεων που αναφέρθηκε παραπάνω, ένα subset query αντιστοιχεί σε μία ερώτηση του τύπου «βρες όλες τις χημικές ενώσεις που περιέχουν τα στοιχεία άζωτο και οξυγόνο», ένα equality query αντιστοιχεί σε μία ερώτηση του τύπου «βρες όλες τις χημικές ενώσεις που αποτελούνται μόνο από άζωτο και οξυγόνο», ενώ ένα superset query αντιστοιχεί σε μία ερώτηση του τύπου «βρες όλες τις χημικές ενώσεις που μπορώ να κατασκευάσω αν διαθέτω μόνο άζωτο και οξυγόνο».

Το ευρετήριο που θα αναπτυχθεί θα πρέπει λοιπόν να μπορεί να απαντάει στα τρία αυτά βασικά είδη ερωτημάτων, ενώ για να δείξουμε ότι είναι πράγματι αποδοτικό θα το συγκρίνουμε με άλλα ευρετήρια που έχουν ήδη αναπτυχθεί για τον τύπο δεδομένων που μας ενδιαφέρει.

1.2 Οργάνωση του τόμου

Ο τόμος αυτός απαρτίζεται συνολικά από επτά κεφάλαια.

Το πρώτο, είναι το παρόν κεφάλαιο, το οποίο αποτελεί μία σύντομη εισαγωγή στο αντικείμενο της εργασίας και στη διάρθρωση του παρόντος τόμου.

Το δεύτερο κεφάλαιο, ασχολείται με την περιγραφή του θέματος της διπλωματικής. Έτσι, περιλαμβάνεται σε αυτό μία ανασκόπηση των ευρετηρίων που έχουν ήδη αναπτυχθεί και εξεταστεί στη διεθνή βιβλιογραφία για τον τύπο δεδομένων με τον οποίο ασχολούμαστε. Παράλληλα, γίνεται μία σύντομη περιγραφή της δομής του B-Δέντρου, η οποία επίσης χρησιμοποιείται στα πλαίσια αυτής της εργασίας, ενώ τέλος διατυπώνεται ο στόχος που η διπλωματική αυτή προσπαθεί να πετύχει με την ανάπτυξη του νέου ευρετηρίου με το οποίο ασχολούμαστε, σε σχέση με τα ήδη υπάρχοντα ευρετήρια.

Στο τρίτο κεφάλαιο, παρουσιάζεται μία αναλυτική περιγραφή της κατασκευής και χρήσης του ευρετηρίου μας για την απάντηση στα ερωτήματα που μας ενδιαφέρουν και τα οποία περιγράψαμε στην προηγούμενη ενότητα. Εκεί φαίνονται επίσης οι διαφορές που αυτό παρουσιάζει σε σχέση με τα ήδη υπάρχοντα ευρετήρια στα οποία στηρίζεται, και ο τρόπος με τον οποίο επιχειρούμε να ξεπεράσουμε τα μειονεκτήματα αυτών.

Στο τέταρτο κεφάλαιο, αναλύεται ο κώδικας με τον οποίο υλοποιούμε το ευρετήριό μας. Έτσι, περιγράφουμε τις βασικές λειτουργίες του κώδικα αυτού, με ταυτόχρονη παράθεση των σημαντικότερων τμημάτων του.

Στο πέμπτο κεφάλαιο ασχολούμαστε με την πειραματική σύγκριση του ευρετηρίου μας με το ήδη υπάρχον inverted file. Παρουσιάζουμε λοιπόν τα δεδομένα πάνω στα οποία έγινε η διενέργεια των πειραμάτων, καθώς και τα αποτελέσματα των πειραμάτων αυτών και τα συμπεράσματα που μπορούμε να εξάγουμε από αυτά για την απόδοση και λειτουργία του ευρετηρίου μας.

Στο έκτο κεφάλαιο, κάνουμε μία σύνοψη των όσων παρουσιάζονται στο προηγούμενα κεφάλαια, και δίνουμε κάποιες κατευθύνσεις σχετικά με περαιτέρω έρευνα που θα είχε ενδιαφέρον να γίνει πάνω στο ευρετήριό μας.

Τέλος, στο έβδομο κεφάλαιο παρουσιάζεται η βιβλιογραφία στην οποία στηρίζεται αυτή η εργασία.

2

Περιγραφή Θέματος

Ευρετήρια για τον τύπο των δεδομένων που μας ενδιαφέρει έχουν αναπτυχθεί και μελετηθεί και στο παρελθόν, τα περισσότερα δε προέρχονται από το χώρο του Information Retrieval και του Data Mining. Στο κεφάλαιο αυτό θα γίνει μία ανασκόπηση των κυριότερων ευρετηρίων που χρησιμοποιούνται για τα δεδομένα αυτά με αναφορές σε σχετική βιβλιογραφία, αναφέροντας επίσης τα μειονεκτήματα αυτών που το ευρετήριο που αναπτύσσεται στα πλαίσια αυτής της εργασίας σκοπεύει να ξεπεράσει.

2.1 Σχετικές εργασίες

Μεγάλο τμήμα της εργασίας αυτής στηρίζεται στην ήδη υπάρχουσα δομή ευρετηρίου του inverted file, την κατασκευή και λειτουργία του οποίου παρουσιάζουμε παρακάτω. Εξετάζουμε ακόμη το signature file το οποίο μπορεί επίσης να χρησιμοποιηθεί για τον τύπο δεδομένων που μας ενδιαφέρουν, ενώ κάνουμε επίσης και μία σύντομη αναφορά στο BTree το οποίο, αν και από μόνο του δε βρίσκει εφαρμογές σε σύνθετα δεδομένα, χρησιμοποιείται επίσης σε αυτή τη διπλωματική σε συνδυασμό με το inverted file.

2.1.1 Το Inverted File

Τα inverted files (ανεστραμμένα αρχεία) είναι αρκετά κοντά στην άποψη που έχει η πλειοψηφία των ανθρώπων για τα ευρετήρια, όπως παραδείγματος χάριν το ευρετήριο ενός βιβλίου. Ένα inverted file αποτελείται από δύο τμήματα. Το πρώτο τμήμα, ή αλλιώς λεξικό (vocabulary), είναι το σύνολο όλων των αντικειμένων που μπορούμε να συναντήσουμε στα δεδομένα μας, διατεταγμένα συνήθως αλφαβητικά ή με κάποιον άλλο τρόπο αν πρόκειται για δεδομένα για τα οποία η αλφαβητική διάταξη δεν έχει νόημα. Το δεύτερο τμήμα είναι η

ανεστραμμένη λίστα (inverted list), η οποία είναι μία λίστα για καθένα από τα αντικείμενα στο λεξικό, η οποία περιέχει δείκτες προς τις εμφανίσεις του αντικειμένου αυτού.

Στο χώρο του information retrieval από τον οποίο προέρχεται η δομή αυτή, όπου χρησιμοποιείται σαν ευρετήριο πάνω στις λέξεις ενός συνόλου από κείμενα, έχει νόημα και η έννοια του granularity, αφού ο κάθε δείκτης στην inverted list μπορεί να δείχνει σε ένα συγκεκριμένο κείμενο, σε ένα block το οποίο περιέχει περισσότερα από ένα κείμενα ή τμήμα ενός κειμένου, ή και σε μία συγκεκριμένη λέξη ενός κειμένου. Αυτό βέβαια δεν έχει νόημα στην περίπτωση με την οποία ασχολούμαστε εδώ, όπου η κάθε λίστα περιέχει δείκτες σε δοσοληψίες, ή για απλότητα τα ids των δοσοληψιών διατεταγμένα κατά αύξουσα σειρά. Εκτός από το id κάθε δοσοληψίας αποθηκεύεται συνήθως στις inverted lists και το πλήθος των αντικειμένων που περιέχει η δοσοληψία αυτή (cardinality), το οποίο χρησιμοποιείται για την αποδοτική αποτίμηση των equality και superset queries.

Εδώ πρέπει επίσης να σημειώσουμε ότι καθώς το λεξικό μπορεί να έχει πολύ μεγάλο μέγεθος ανάλογα με τη βάση πάνω στην οποία έχει κατασκευαστεί, χρησιμοποιείται συχνά ένα επιπλέον ευρετήριο πάνω σε αυτό, το οποίο είναι κάποιο από τα κλασσικά ευρετήρια των βάσεων δεδομένων (BTrees, B+Trees, Bitmaps, Hash, κ.λ.π.) και βοηθάει στη γρηγορότερη εύρεση του στοιχείου του λεξικού το οποίο ψάχνουμε [HM99], [WMB99], [ZMR98].

2.1.1.1 Κατασκευή

Η κατασκευή του inverted file, όπως προκύπτει και από τα παραπάνω, είναι αρκετά απλή. Αρκεί να κατασκευάσουμε το λεξικό διασχίζοντας τη βάση για να καταγράψουμε τα αντικείμενα που συναντώνται σε αυτήν και στη συνέχεια, διασχίζοντας τις δοσοληψίες με τη σειρά, να προσθέσουμε την κάθε δοσοληψία στις ανεστραμμένες λίστες των αντικειμένων που αυτή περιέχει. Έτσι, το inverted file που προκύπτει για τις δοσοληψίες που παραθέσαμε στον πίνακα 1.1 του προηγούμενου κεφαλαίου φαίνεται στον πίνακα 2.1

Λεξικό	Inverted list
a	1(4), 2(3), 3(4), 4(4)
b	3(4), 6(4)
c	1(4), 3(4), 4(4), 6(4)
d	1(4), 3(4)
e	4(4), 5(3), 6(4)
f	1(4), 2(3), 4(4), 5(3), 6(4)
g	2(3), 5(3)

Πίνακας 2.1.: Inverted File για τις δοσοληψίες του πίνακα 1.1

Αφού ολοκληρωθεί η κατασκευή του inverted file μπορούμε να το αποθηκεύσουμε στο δίσκο, σε δύο διαφορετικά αρχεία, ένα για το λεξικό και ένα για τις ανεστραμμένες λίστες. Στις ήδη υπάρχουσες εργασίες συνηθίζεται η υπόθεση ότι το λεξικό χωράει στην κύρια μνήμη [ZMS92], και έτσι, κατά τη διάρκεια τις αποτίμησης των ερωτημάτων μπορούμε να

κρατήσουμε το λεξικό στην κύρια μνήμη ανατρέχοντας στο δίσκο μόνο όποτε χρειάζεται να προσπελάσουμε κάποια λίστα, πράγμα το οποίο μειώνει τις προσπελάσεις στο δίσκο που απαιτούνται για την αποτίμηση ενός ερωτήματος και άρα και το χρόνο αποτίμησης. Στην περίπτωση που το λεξικό δε χωράει ολόκληρο στην κύρια μνήμη μπορούμε να κρατήσουμε στην κύρια μνήμη μόνο τα τμήματα του λεξικού που περιέχουν τα πιο συχνά αντικείμενα, ανατρέχοντας στο δίσκο όταν χρειαζόμαστε τα υπόλοιπα

Στη βιβλιογραφία έχουν προταθεί επίσης μέθοδοι για την περίπτωση που οι ανεστραμμένες λίστες είναι πολύ μεγάλες και έτσι το inverted file δεν χωράει ολόκληρο στην κύρια μνήμη κατά την κατασκευή του και με αποτέλεσμα να μην μπορεί να κατασκευαστεί σε ένα βήμα όπως περιγράφηκε προηγουμένως [BA99]. Στην περίπτωση αυτή, κάθε φορά που εξαντλείται η κύρια μνήμη γράφουμε το inverted file που κατασκευάσαμε μέχρι στιγμής στο δίσκο και συνεχίζουμε να διατρέχουμε τις δοσοληψίες κατασκευάζοντας όμως ένα καινούργιο inverted file. Έτσι, καταλήγουμε στο τέλος με περισσότερα από ένα inverted files τα οποία μπορούμε να συγχωνεύσουμε με κάποιον από τις πολλούς αλγορίθμους για συγχώνευση λιστών που υπάρχουν.

2.1.1.2 Συμπίεση

Καθώς το πλήθος των δοσοληψιών που υπάρχουν μέσα στη βάση μπορεί να είναι πολύ μεγάλο, είναι πιο αποδοτικό αντί να αποθηκεύουμε στην κάθε ανεστραμμένη λίστα το id μιας δοσοληψίας να αποθηκεύσουμε τη διαφορά του από το προηγούμενο id που υπάρχει μέσα στην λίστα αυτή (θεωρούμε ότι τα ids σε κάθε λίστα είναι ταξινομημένα κατά αύξουσα σειρά). Αυτό συμβαίνει γιατί αν και θεωρητικά η μέγιστη διαφορά ανάμεσα σε δοσοληψίες θα είναι ίση με τη μεγαλύτερη δοσοληψία, στατιστικά έχει παρατηρηθεί ότι αυτό συμβαίνει σπάνια, και οι διαφορές που χρειάζεται να αποθηκεύσουμε είναι αρκετά μικρότερες από τα ids των αντίστοιχων δοσοληψιών. Έτσι, το inverted file που παρουσιάζεται στον πίνακα 2.1. θα γίνει όπως φαίνεται στον πίνακα 2.2.

Λεξικό	Inverted list
a	1(4), 1(3), 1(4), 1(4)
b	3(4), 3(4)
c	1(4), 2(4), 1(4), 2(4)
d	1(4), 2(4)
e	4(4), 1(3), 1(4)
f	1(4), 1(3), 2(4), 1(3), 1(4)
g	2(3), 3(3)

Πίνακας 2.2.: Inverted File για τις δοσοληψίες του πίνακα 1.1 με αποθήκευση της διαφοράς ανάμεσα στις δοσοληψίες αντί για τα ids των δοσοληψιών

Από τη στιγμή που αναπαριστούμε τις δοσοληψίες στις ανεστραμμένες λίστες σα διαφορές ανάμεσα στα ids, έχουν προταθεί διάφορες μέθοδοι οι οποίες εκμεταλλεύονται την ιδιότητα

που έχουν οι διαφορές αυτές να είναι σχετικά μικρές για να μειώσουν τον αποθηκευτικό χώρο που χρειάζονται οι λίστες. Οι μέθοδοι αυτοί χωρίζονται σε γενικές (global) και τοπικές (local). Οι γενικές μέθοδοι κωδικοποιούν όλες τις λίστες με βάση το ίδιο μοντέλο, ενώ οι τοπικές μέθοδοι χρησιμοποιούν διαφορετικό μοντέλο για την κωδικοποίηση της κάθε λίστας, ανάλογα με διάφορες παραμέτρους όπως για παράδειγμα η συχνότητα εμφάνισης του κάθε όρου και τα ιδιαίτερα στατιστικά χαρακτηριστικά της κάθε λίστας. Οι πιο γνωστές μέθοδοι κωδικοποίησης είναι η unary κωδικοποίηση, οι Elias-γ και Elias-δ, καθώς και η κωδικοποίηση Golomb.

Με τη unary (εναδική) μέθοδο κωδικοποίησης, ο κάθε αριθμός x αναπαρίσταται με x-1 μηδενικά ακολουθούμενα από έναν άσσο. Με τη μέθοδο Elias-γ ο κάθε αριθμός x αναπαρίσταται με το $\lfloor \log_2 x \rfloor + 1$ σε εναδική αναπαράσταση (δηλαδή $\lfloor \log_2 x \rfloor$ μηδενικά ακολουθούμενα από έναν άσσο), ακολουθούμενο από το $x - 2^{\lfloor \log_2 x \rfloor}$ σε δυαδική αναπαράσταση (δηλαδή τη δυαδική αναπαράσταση του x χωρίς όμως το περισσότερο σημαντικό bit). Με τη μέθοδο Elias-δ ο κάθε αριθμός x αναπαρίσταται με το $\lfloor \log_2 x \rfloor + 1$ σε κωδικοποίηση Elias-γ, ακολουθούμενο πάλι από το $x - 2^{\lfloor \log_2 x \rfloor}$ σε δυαδική αναπαράσταση.

Στον πίνακα 2.3. φαίνονται οι αριθμοί από το 1 έως το 10 κωδικοποιημένοι με τις τρεις παραπάνω μεθόδους. Το πλεονέκτημα που προσφέρουν αυτές οι μέθοδοι κωδικοποίησης, όπως μπορεί να φανεί και στον πίνακα, είναι ότι για κάθε ζευγάρι κωδικοποιημένων αριθμών x_1 και x_2 μπορούμε να βρούμε έναν ακέραιο n τέτοιο ώστε τα n πρώτα ψηφία του x_1 να μην ταυτίζονται με αυτά του x_2 γεγονός το οποίο μας δίνει μία δυνατότητα διάκρισης των αριθμών. Με τον τρόπο αυτό δεν είναι πλέον απαραίτητο να χρησιμοποιούμε ένα σταθερό αριθμό από bits για την αποθήκευση του κάθε αριθμού ούτως ώστε να μπορούμε να τους ξεχωρίσουμε, και έτσι οι μικροί αριθμοί, που όπως είπαμε προηγουμένως είναι συχνότεροι, αποθηκεύονται με λιγότερα bits συντελώντας έτσι σε σημαντική εξοικονόμηση αποθηκευτικού χώρου, ιδιαίτερα στις ανεστραμμένες λίστες των πιο συχνών αντικειμένων της βάσης.

X	Κωδικοποίηση		
	Unary	Elias-γ	Elias-δ
1	1	1	1
2	01	010	0100
3	001	011	0101
4	0001	00100	0100
5	00001	00101	01101
6	000001	00110	01110
7	0000001	00111	01111
8	00000001	0001000	00100000
9	000000001	0001001	00100001
10	0000000001	0001010	00100010

Πίνακας 2.3.: Κωδικοποίηση αριθμών

Η κωδικοποίηση Elias-δ αν και απαιτεί περισσότερα bits από την Elias-γ για να κωδικοποιήσει τους αριθμούς που φαίνονται στον πίνακα 2.3. είναι πιο αποδοτική από αυτήν για τιμές του x μεγαλύτερες από 15. Και οι δύο αυτές μέθοδοι κωδικοποίησης αποτελούν υποπεριπτώσεις μίας γενικότερης μεθόδου που παρουσιάζεται στη συνέχεια. Έστω V ένα (πιθανώς άπειρο) διάνυσμα θετικών ακεραίων v_i το άθροισμα των συνιστωσών του οποίου είναι μεγαλύτερο ή ίσο από το πλήθος των δοσοληψιών της βάσης μας. Ο αριθμός x μπορεί να κωδικοποιηθεί σε σχέση με το V αν βρούμε έναν ακέραιο k τέτοιο ώστε $\sum_{j=1}^{k-1} v_j < x \leq \sum_{j=1}^{k} v_j$ και τον κωδικοποιήσουμε ακολουθούμενο από τη διαφορά $d = x - \sum_{j=1}^{k-1} v_j - 1$. Έτσι, η μέθοδος Elias-γ αποτελεί υποπερίπτωση αυτής της γενικότερης μεθόδου με V=(1, 2, 4, 8, 16, …).

Προφανώς, η συμπίεση του inverted file σημαίνει ότι κατά την αποτίμηση των ερωτημάτων θα πρέπει επιπλέον να αποκωδικοποιήσουμε τα τμήματα του ευρετηρίου που χρειαζόμαστε, πράγμα το οποίο εκ πρώτης όψεως αυξάνει το χρόνο που απαιτεί η αποτίμηση. Παρ' όλα αυτά, η συμπίεση που επιτυγχάνεται με τις περισσότερες μεθόδους κωδικοποίησης είναι τέτοια ώστε ο χρόνος που χρειάζεται για την αποκωδικοποίηση να είναι μικρός συγκριτικά με το χρόνο που εξοικονομείται κατά τη μεταφορά των δεδομένων από το δίσκο στην κύρια μνήμη, μια που ο χώρος που καταλαμβάνουν αυτά μετά τη συμπίεση είναι σημαντικά μικρότερος.

2.1.1.3 Επεξεργασία ερωτημάτων

Καθώς η δομή αυτή αναπτύχθηκε και χρησιμοποιήθηκε αρχικά στο χώρο του Information Retrieval θεωρήθηκε σημαντικό να μπορεί να χρησιμοποιηθεί μεταξύ άλλων για την αποτίμηση boolean queries, δηλαδή ερωτημάτων των οποίων οι όροι (ή οι αρνήσεις τους) συνδέονται με τις τελεστές AND και OR. Τα ερωτήματα των οποίων οι όροι συνδέονται αποκλειστικά με τελεστές AND ονομάζονται <u>συζευκτικά (conjunctive)</u>, ενώ αυτά των οποίων οι όροι συνδέονται αποκλειστικά με τελεστές OR ονομάζονται <u>διαζευκτικά (disjunctive)</u>. Για να απαντήσουμε σε ένα γενικό boolean ερώτημα αρκεί να προσπελάσουμε στο λεξικό τις όρους που εμφανίζονται στο ερώτημα, να συγκεντρώσουμε τις αντίστοιχες λίστες και να κάνουμε τις κατάλληλες πράξεις ανάμεσά τους. Οι πράξεις αυτές είναι τομή λιστών για τον τελεστή AND, ένωση για τον τελεστή OR και αφαίρεση για αρνήσεις (NOT). Εδώ πρέπει να σημειωθεί ότι στην περίπτωση που έχουμε ένα συζευκτικό ερώτημα με πολλούς όρους μπορεί να αποδειχθεί πιο αποδοτικό να αποτιμήσουμε τμήμα αυτών μέχρις ότου μείνει αρκετά μικρός αριθμός υποψήφιων δοσοληψιών και στη συνέχεια να ψάξουμε κατευθείαν

αυτές τις δοσοληψίες για τους υπόλοιπους όρους, ιδιαίτερα στην περίπτωση που το λεξικό έχει μεγάλο μέγεθος ενώ οι δοσοληψίες όχι.

Ειδικότερα για τα ερωτήματα που μας απασχολούν σε αυτή την εργασία, τα subset queries αποτελούν υποσύνολο των boolean ερωτημάτων που αναφέρθηκαν παραπάνω και μπορούμε να απαντήσουμε σε αυτά όπως θα απαντούσαμε σε ένα οποιοδήποτε συζευκτικό ερώτημα. Η τομή των λιστών που απαιτείται για την αποτίμηση του ερωτήματος γίνεται συνήθως με τον αλγόριθμο MergeJoin ο οποίος θα αναπτυχθεί περαιτέρω σε επόμενα κεφάλαια της εργασίας. Για να απαντήσουμε σε ένα equality query εργαζόμαστε με τον ίδιο ακριβώς τρόπο, με τη διαφορά ότι ελέγχουμε το πλήθος των εγγραφών της κάθε δοσοληψίας, το οποίο έχουμε κρατήσει στις ανεστραμμένες λίστες μαζί με το id της δοσοληψίας όπως αναφέρθηκε προηγουμένως, και απαιτούμε να είναι ίσο με το πλήθος των όρων του ερωτήματος. Τέλος, για να απαντήσουμε σε ένα superset query, αφού μαζέψουμε τις λίστες όλων των όρων που εμφανίζονται στο ερώτημα μετράμε για κάθε δοσοληψία σε πόσες από τις λίστες αυτές εμφανίζεται. Αν κάποια δοσοληψία εμφανίζεται σε λιγότερες λίστες από το πλήθος των στοιχείων που περιέχει, αυτό σημαίνει ότι περιέχει και άλλα στοιχεία εκτός από τις όρους του ερωτήματος και έτσι την απορρίπτουμε. Η μέτρηση αυτή γίνεται με το να διασχίζουμε την καθεμία από τις λίστες των αντικειμένων του query set με τη σειρά και να ψάχνουμε για τις δοσοληψίες που συναντάμε στις επόμενες λίστες. Ο λόγος που δεν ψάχνουμε και στις προηγούμενες λίστες είναι ότι αν η δοσοληψία που εξετάζουμε βρίσκεται εκεί τότε την έχουμε ήδη βρει σε κάποιο από τα προηγούμενα βήματα.

2.1.1.4 Ανανέωση

Η ανανέωση του inverted file είναι εξίσου απλή με την κατασκευή του αφού κάθε φορά που εισάγουμε ή διαγράφουμε μία δοσοληψία αρκεί να επισκεφθούμε τις λίστες των αντικειμένων τις δοσοληψίας αυτής και να προσθέσουμε ή να διαγράψουμε αντίστοιχα το id τις δοσοληψίας. Στην περίπτωση που εισάγουμε μεγάλο πλήθος δοσοληψιών ταυτόχρονα μπορούμε να φτιάξουμε ένα καινούργιο inverted file για τις καινούργιες αυτές δοσοληψίες και στη συνέχεια να το συγχωνεύσουμε με το ήδη υπάρχον όπως αναφέρθηκε παραπάνω. Οποιαδήποτε άλλη μεταβολή της βάσης (π.χ. update) μπορεί να θεωρηθεί συνδυασμός εισαγωγών και διαγραφών.

2.1.2 To Signature File

Τα signature files (αρχεία υπογραφών) παίρνουν το όνομά τους από μία σειρά από bits η οποία προκύπτει με βάση τα περιεχόμενα της κάθε δοσοληψίας και προσδιορίζει έτσι το περιεχόμενό της και η οποία καλείται signature (υπογραφή). Η λειτουργία της δομής αυτής στηρίζεται στο ότι εφαρμόζοντας κατάλληλο hashing στα αντικείμενα της βάσης μπορούμε

να πάρουμε για το καθένα μία μοναδική υπογραφή (ή σχεδόν μοναδική αν έχουμε εξαιρετικά μεγάλο πλήθος διαφορετικών αντικειμένων) με βάση την οποία μπορούμε στη συνέχεια να κατασκευάσουμε την υπογραφή της κάθε δοσοληψίας αλλά και να αναζητήσουμε το αντικείμενο αυτό όταν χρειαστεί. Συγκεκριμένα, η υπογραφή της κάθε δοσοληψίας προκύπτει από το λογικό OR των υπογραφών των αντικειμένων τα οποία περιέχει. Έτσι, όταν ψάχνουμε για ένα αντικείμενο, βλέπουμε ποια bits στην υπογραφή του είναι ίσα με 1 και ψάχνουμε να βρούμε τις δοσοληψίες εκείνες που έχουν τα bits αυτά ίσα με 1 στη δική τους υπογραφή. Δυστυχώς, όπως θα φανεί και παρακάτω, σε κάθε περίπτωση πρέπει να εξετάζουμε στο τέλος τις δοσοληψίες που πήραμε σαν αποτέλεσμα για να βεβαιωθούμε ότι πράγματι υπάρχει εκεί το ζητούμενο αντικείμενο, αφού τα bits που αντιστοιχούν στην υπογραφή κάποιου αντικειμένου μπορούν να γίνουν ίσα με 1 από κατάλληλο συνδυασμό άλλων αντικειμένων, χωρίς το αντικείμενο το οποίο ψάχνουμε να είναι παρόν στη δοσοληψία. Έτσι, το ευρετήριο αυτό μας βοηθάει μάλλον να ανακαλύψουμε σε ποιες δοσοληψίες δε βρίσκεται ένα αντικείμενο παρά σε ποιες δοσοληψίες βρίσκεται.

2.1.2.1 Κατασκευή

Αρχικά επιλέγουμε την ή τις συναρτήσεις κατάτμησης (hash functions) με βάση τις οποίες θα δημιουργήσουμε τις υπογραφές των αντικειμένων. Συνήθως χρησιμοποιούνται περισσότερες από μία συναρτήσεις, καθεμία από τις οποίες θέτει κάποιο bit της υπογραφής ίσο με 1, με αποτέλεσμα με τη χρήση ν συναρτήσεων να έχουμε και ν άσσους στην υπογραφή του κάθε αντικειμένου. Προφανώς σε κάποιες περιπτώσεις μπορεί κάποια αντικείμενα να έχουν και λιγότερους από ν άσσους στην υπογραφή τους αν δύο διαφορετικές συναρτήσεις τύχει να θέσουν το ίδιο bit ίσο με 1. Στη συνέχεια δημιουργούμε τις υπογραφές των δοσοληψιών παίρνοντας το λογικό OR των υπογραφών των αντικειμένων που περιέχονται μέσα σε αυτές.

Αν για παράδειγμα οι υπογραφές των αντικειμένων της βάσης που παρουσιάζεται στον πίνακα 1.1, μετά από κάποιο hashing, είναι αυτές που φαίνονται στον πίνακα 2.4., τότε οι υπογραφές των δοσοληψιών της βάσης αυτής θα είναι αυτές του πίνακα 2.5.

Αντικείμενο	Υπογραφή
a	00010100
b	10001000
c	11000000
d	10000001
e	00100010
f	00011000
g	00000101

Πίνακας 2.4.: Υπογραφές αντικειμένων του Πίνακα 1.1.

ID	Υπογραφή
1	11011101
2	00011101
3	11011101
4	11111110
5	00111111
6	11111010

Πίνακας 2.5.: Υπογραφές δοσοληψιών του Πίνακα 1.1.

Στο παραπάνω παράδειγμα μπορούμε να δούμε καθαρά ότι οι υπογραφές των δοσοληψιών δεν είναι μοναδικές αφού, όπως βλέπουμε, οι δοσοληψίες 1 και 3 έχουν την ίδια ακριβώς υπογραφή χωρίς να περιέχουν τα ίδια αντικείμενα. Για να εντοπίσουμε ένα αντικείμενο, π.χ. το αντικείμενο a, πρέπει να ψάξουμε να βρούμε όλες τις δοσοληψίες το 4° και το 6° bit των υπογραφών των οποίων είναι ίσα με 1. Βλέπουμε όμως, ότι με τον τρόπο αυτό θα μας επιστραφεί και η δοσοληψία 5 χωρίς να περιέχει το a, αφού ο συνδυασμός των αντικειμένων f και g στη δοσοληψία αυτή είχε σαν αποτέλεσμα να τεθούν τα bits που μας ενδιαφέρουν ίσα με 1. Το γεγονός αυτό, που ονομάζεται false match είναι ο λόγος για τον οποίο πρέπει, σε αντίθεση με το inverted file, να εξετάζουμε τις δοσοληψίες που μας επιστρέφει το ευρετήριο για να βεβαιωθούμε ότι περιέχουν τα αντικείμενα που ψάχνουμε.

Οι υπογραφές αυτές, μπορούν να αποθηκευτούν μετά την κατασκευή τους σειριακά στο δίσκο, δημιουργώντας έτσι ένα αρχείο, το signature file. Εκτός από τη σειριακή αποθήκευση όλων των υπογραφών στο ίδιο αρχείο έχει προταθεί επίσης η χρήση ενός αρχείου για κάθε bit των υπογραφών, δηλαδή για κάθε τεμάχιο (slice) του signature file (ένα αρχείο που θα περιέχει τα πρώτα bits όλων των υπογραφών, ένα που θα περιέχει τα δεύτερα bits, κ.ο.κ). Η μέθοδος αυτή ονομάζεται bitslicing (τεμαχισμός) και μπορεί να κάνει πιο αποδοτική την αναζήτηση στο ευρετήριο, αφού μπορούμε κατά την αποτίμηση ενός ερωτήματος να προσπελάσουμε μόνο τα αρχεία που περιέχουν τα bits που μας ενδιαφέρουν το καθένα από τα οποία θα είναι n φορές πιο μικρό από το αρχικό αρχείο, όπου n το μήκος της υπογραφής.

Δυστυχώς, ακόμη και με τη μέθοδο του bitslicing, προκύπτουν πολύ μεγάλα αρχεία όταν έχουμε μεγάλες βάσεις, πράγμα που δυσχεραίνει τη μεταφορά τους στην κύρια μνήμη για την επεξεργασία των ερωτημάτων. Έτσι, έχει προταθεί επίσης η μέθοδος των blocked signature files όπου αντί να φτιάχνουμε μία υπογραφή για κάθε εγγραφή φτιάχνουμε μία υπογραφή για κάθε block από B εγγραφές, μειώνοντας έτσι το μέγεθος των αρχείων κατά B. Στην περίπτωση αυτή βέβαια, υπάρχουν ακόμη περισσότερες πιθανότητες δύο blocks να έχουν την ίδια υπογραφή χωρίς να περιέχουν τα ίδια στοιχεία, πράγμα το οποίο σημαίνει ότι η συγκεκριμένη μέθοδος απαιτεί μεγαλύτερο μήκος υπογραφών για να είναι αποδοτική. Για να εξαλείψουμε ακόμη περισσότερο την πιθανότητα για false matches μπορούμε να χρησιμοποιήσουμε διαφορετική απεικόνιση δοσοληψίας σε block για κάθε slice. Έτσι, μία

δοσοληψία μπορεί να βρίσκεται στο 3^{ο} block για το 1^{ο} slice, στο 6^{ο} block για το 2^{ο} slice, στο 14^{ο} block για το 3^{ο} slice κ.ο.κ.

Για την κατασκευή των signature files υπάρχουν διάφορες μέθοδοι για την επιλογή των βέλτιστων hash συναρτήσεων, όπως επίσης και του βέλτιστου μήκους των υπογραφών συναρτήσει των αναφορών στο δίσκο που θέλουμε να κάνουμε σε κάθε ερώτηση και του επιθυμητού ορίου των false matches σε κάθε ερώτημα. Γενικότερα τα false matches μειώνονται καθώς αυξάνεται το πλήθος των hash συναρτήσεων και το μήκος των υπογραφών, πράγμα το οποίο όμως έχει σαν αποτέλεσμα την αύξηση της υπολογιστικής δύναμης και του αποθηκευτικού χώρου που απαιτείται για την κατασκευή και διατήρηση του signature file. Έτσι, κάθε φόρα επιλέγονται οι παράμετροι του signature file ανάλογα με τις δυνατότητες του συστήματος στο οποίο θα κατασκευαστεί και τις απαιτήσεις της εφαρμογής την οποία θα εξυπηρετήσει.

2.1.2.2 Συμπίεση

Δυστυχώς, το signature file, αν και μπορεί να συμπιεστεί δεν παρουσιάζει αύξηση της απόδοσής του με τη συμπίεση όπως το inverted file. Αυτό συμβαίνει γιατί τα δεδομένα που αποθηκεύονται σε ένα signature file είναι πολύ πιο πυκνά από αυτά που αποθηκεύονται σε ένα inverted file με αποτέλεσμα η συμπίεση που μπορούμε να τους κάνουμε να είναι μικρότερη. Έτσι, ο χρόνος που εξοικονομείται κατά τη μεταφορά των συμπιεσμένων πια δεδομένων από το δίσκο στην κύρια μνήμη δεν είναι αρκετός για την αποσυμπίεση των δεδομένων, με αποτέλεσμα ένα συμπιεσμένο signature file να είναι τελικά λιγότερο αποδοτικό από ένα ασυμπίεστο.

2.1.2.3 Επεξεργασία ερωτημάτων

Η απάντηση σε κάποιο γενικό boolean ερώτημα γίνεται όπως είπαμε και προηγουμένως διατρέχοντας σειριακά το signature file για να βρούμε τις υποψήφιες δοσοληψίες και στη συνέχεια ελέγχοντάς τις για να βεβαιωθούμε ότι πράγματι πληρούν τις προδιαγραφές που τέθηκαν με το ερώτημα. Στην περίπτωση διαζευκτικού ερωτήματος απαιτούμε οι υποψήφιες δοσοληψίες να έχουν ίσα με ένα τουλάχιστον τα bits που αντιστοιχούν στην υπογραφή ενός από τα αντικείμενα, ενώ στην περίπτωση συζευκτικών ερωτημάτων πρέπει οι υποψήφιες δοσοληψίες να έχουν ίσα με ένα τα bits που αντιστοιχούν στις υπογραφές όλων των όρων του ερωτήματος. Τέλος, στην περίπτωση άρνησης, παίρνουμε τις δοσοληψίες που έχουν ίσα με 0 τα bits τα οποία στην υπογραφή του αντικειμένου είναι ίσα με 1 και αυτή είναι και η μοναδική περίπτωση που δεν μιλάμε για υποψήφιες δοσοληψίες αφού είμαστε σίγουροι ότι οι δοσοληψίες που πήραμε δεν περιέχουν το αντικείμενο.

Ειδικότερα για τα είδη των ερωτημάτων που εξετάζουμε σε αυτή την εργασία, τα subset queries αποτιμώνται σα συζευκτικά ερωτήματα με τον τρόπο που μόλις αναφέραμε. Για τα equality και superset queries απαιτούμε να μην είναι ίσο με 1 κανένα άλλο bit εκτός από αυτά που ορίζονται από τις υπογραφές των όρων του query, αλλά και σε αυτήν την περίπτωση θα πρέπει να ελέγξουμε τις δοσοληψίες που θα πάρουμε για να βεβαιωθούμε ότι ικανοποιούν το ερώτημα.

2.1.2.4 Ανανέωση

Η εισαγωγή καινούργιων δοσοληψιών στη βάση είναι αρκετά απλή αφού αρκεί να φτιάξουμε την υπογραφή της καινούργιας δοσοληψίας και να την προσθέσουμε στο τέλος του signature file. Δυστυχώς η διαγραφή δεν είναι τόσο απλή αφού μετά τη διαγραφή της υπογραφής της δοσοληψίας που θέλουμε να διαγράψουμε, θα πρέπει να ολισθήσουμε προς τα εμπρός (shift) τις υπογραφές των δοσοληψιών που την ακολουθούν για να μη μείνει κενός χώρος στο δίσκο, πράγμα το οποίο μπορεί να έχει σαν αποτέλεσμα να πρέπει να διαβαστεί ολόκληρο σχεδόν το signature file στην περίπτωση που διαγράφεται κάποια από τις πρώτες δοσοληψίες. Επίσης η ενημέρωση κάποιας δοσοληψίας μπορεί να γίνει με τον υπολογισμό της νέας της υπογραφής και την αντικατάσταση με αυτήν της παλιάς στο signature file.

2.1.2.5 Παραλλαγές – Προεκτάσεις

Ένα από τα μεγαλύτερα μειονεκτήματα του signature file όπως αυτό περιγράφηκε παραπάνω, είναι το γεγονός ότι η πρόσβαση σε αυτό μπορεί να γίνει μόνο σειριακά, πράγμα το οποίο έχει ως αποτέλεσμα αυξημένο χρόνο αποτίμησης ερωτημάτων. Για το λόγο αυτό έχουν προταθεί διάφορες προεκτάσεις του signature file οι οποίες αντιμετωπίζουν το πρόβλημα αυτό, όπως είναι το signature tree και το extendible signature hashing.

Οι προεκτάσεις αυτές ισοδυναμούν στην πραγματικότητα με το να χρησιμοποιήσουμε ένα δεύτερο ευρετήριο πάνω από το signature file κάνοντας έτσι εύκολη την πρόσβαση σε συγκεκριμένα μόνο τμήματα αυτού, πράγμα το οποίο, όπως έχουμε ήδη αναφέρει, γίνεται και με το λεξικό του inverted file. Έτσι, το signature tree βάζει τις εγγραφές του signature file στα φύλλα ενός δέντρου (κάθε φύλλο περιέχει n εγγραφές) και τοποθετεί σε κάθε εσωτερικό κόμβο του δέντρου δείκτες προς τους άλλους κόμβους ή τα φύλλα που βρίσκονται κάτω από αυτόν, σε καθέναν από τους οποίους αντιστοιχεί ένα κλειδί που δημιουργείται παίρνοντας το λογικό OR των εγγραφών του κόμβου στον οποίο δείχνει ο δείκτης. Αντίστοιχα, το extendible signature hashing χωρίζει τις εγγραφές του signature file σε buckets ανάλογα με το πρόθεμά τους. Οι εγγραφές που βρίσκονται στο ίδιο bucket έχουν κοινό πρόθεμα και έτσι η πρόσβαση στα buckets γίνεται μέσω ενός λεξικού στο οποίο αποθηκεύονται τα προθέματα, περίπου όπως συμβαίνει και με ένα κλασσικό hash ευρετήριο.

2.1.3 Το B-Tree

Το B-Tree (Β-Δέντρο) δε συνηθίζεται να χρησιμοποιείται για βάσεις του τύπου που μας απασχολεί, παρά μόνο ίσως συμπληρωματικά σε άλλα ευρετήρια για να κάνει τη χρήση τους ευκολότερη, όπως ειπώθηκε παραπάνω. Εξαίρεση αποτελεί η περίπτωση πλήρως σχεσιακών συστημάτων βάσεων δεδομένων, όπου τα δεδομένα που παρουσιάσαμε στον πίνακα 1.1. δε θα αποθηκεύονται πλέον ως δοσοληψίες με τιμές – σύνολα, αλλά με τον τρόπο που φαίνεται στον πίνακα 2.6.

ID	Τιμή
1	a
1	c
1	d
1	f
2	a
2	g
2	f
3	a
3	b
3	c
3	d
4	a
4	c
4	e
4	f
5	e
5	f
5	g
6	b
6	c
6	e
6	f

Πίνακας 2.6.: Αποθήκευση των δοσοληψιών του πίνακα 1.1. σε πλήρως σχεσιακή βάση

Έτσι, τα ερωτήματα που μας ενδιαφέρουν και περιγράψαμε στο κεφάλαιο 1, θα εκφράζονται πλέον σαν ενώσεις ή τομές ανάμεσα σε άλλα, απλούστερα ερωτήματα. Για παράδειγμα ένα subset query με query set το {a, b, c} μπορεί να εκφραστεί ως $\Pi_{ID}(\sigma_{\text{τιμή=a}} \cap \sigma_{\text{τιμή=b}} \cap \sigma_{\text{τιμή=c}})$. Εξετάζουμε λοιπόν το ευρετήριο αυτό σύντομα στο παρόν κεφάλαιο, καθώς στην εργασία αυτή θα χρησιμοποιηθεί σε συνδυασμό με το inverted file όπως ήδη αναφέρθηκε.

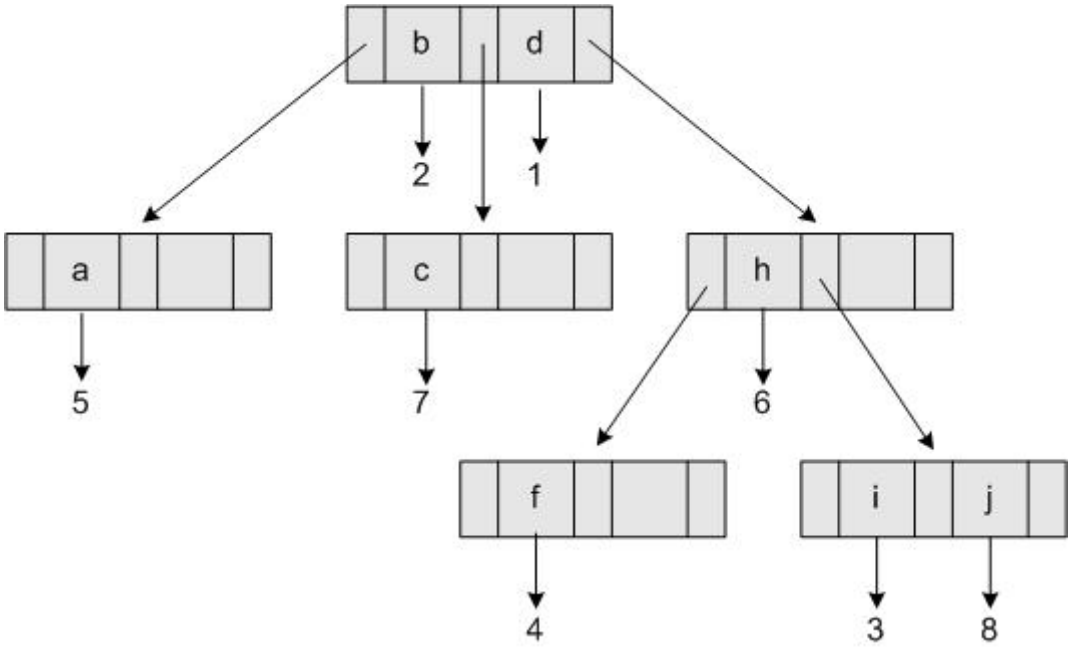
Το Β-Δέντρο είναι ένα από τα ευρύτερα χρησιμοποιούμενα ευρετήρια στις βάσεις δεδομένων καθώς βρίσκει εφαρμογή σε βάσεις με πολλούς διαφορετικούς τύπους αντικειμένων. Είναι στην ουσία ένα ισοσκελισμένο δέντρο οι κόμβοι του οποίου έχουν δείκτες προς τις εγγραφές της βάσης, σε καθέναν από τους οποίους αντιστοιχεί ένα κλειδί που είναι το πεδίο της εγγραφής πάνω στο οποίο κατασκευάζεται το ευρετήριο. Εκτός από δείκτες προς τις

διάφορες εγγραφές της βάσης, ο καθένας από τους εσωτερικούς κόμβους του δέντρου έχει επίσης δείκτες προς τα παιδιά του, έτσι ώστε ο δείκτης i να δείχνει σε έναν κόμβο τα κλειδιά του οποίου είναι μεγαλύτερα από το κλειδί i-1 του τρέχοντα κόμβου και μικρότερα από το κλειδί i. Ήδη από αυτό το σημείο γίνεται ίσως φανερό ότι ένα Β-Δέντρο βρίσκει εφαρμογή σε οποιονδήποτε τύπο δεδομένων υπό τον όρο ότι αυτά μπορούν να διαταχθούν με κάποιον τρόπο. Αυτός είναι και ο λόγος που δε χρησιμοποιείται σε σύνολα δεδομένων όπως είπαμε παραπάνω, αφού δεν έχει νόημα η διάταξη ενός συνόλου από σύνολα.

Στον πίνακα 2.7. που ακολουθεί φαίνεται μία ενδεικτική βάση η κάθε δοσοληψία της οποίας περιέχει εκτός από το id της και ένα δεύτερο στοιχείο που συμβολίζεται με κάποιο γράμμα του λατινικού αλφαβήτου (έχει νόημα η διάταξη), ενώ στο σχήμα 2.1. φαίνεται το Β-Δέντρο που κατασκευάζεται πάνω στα γράμματα που περιέχονται στις δοσοληψίες της βάσης αυτής, υποθέτοντας ότι στον κάθε κόμβο του δέντρου χωράνε το πολύ δύο κλειδιά.

ID	Τιμή
1	d
2	b
3	i
4	f
5	a
6	h
7	c
8	j

Πίνακας 2.7.: Παράδειγμα δοσοληψιών με τιμές από το λατινικό αλφάβητο



Σχήμα 2.1.: Β-Δέντρο πάνω στις τιμές των δοσοληψιών του πίνακα 2.6.

2.1.3.1 Κατασκευή

Για την κατασκευή ενός Β-Δέντρου αρκεί να διατρέξουμε τις δοσοληψίες της βάσης μας από την αρχή προς το τέλος και για καθεμία από αυτές να εισάγουμε στο δέντρο το αντίστοιχο κλειδί μαζί με το δείκτη προς τη δοσοληψία. Στην πραγματικότητα, επειδή όπως αναφέρθηκε και προηγουμένως το δέντρο πρέπει να είναι ισοσκελισμένο, δηλαδή το μεγαλύτερο μονοπάτι από τη ρίζα πρέπει να είναι το πολύ κατά έναν κόμβο μεγαλύτερο από το μικρότερο, οι αλγόριθμοι που χρησιμοποιούνται για την κατασκευή Β-Δέντρων έχουν ιδιαίτερη πρόβλεψη ώστε να μετασχηματίζουν αν χρειαστεί το δέντρο ούτως ώστε να παραμένει συνέχεια ισοσκελισμένο.

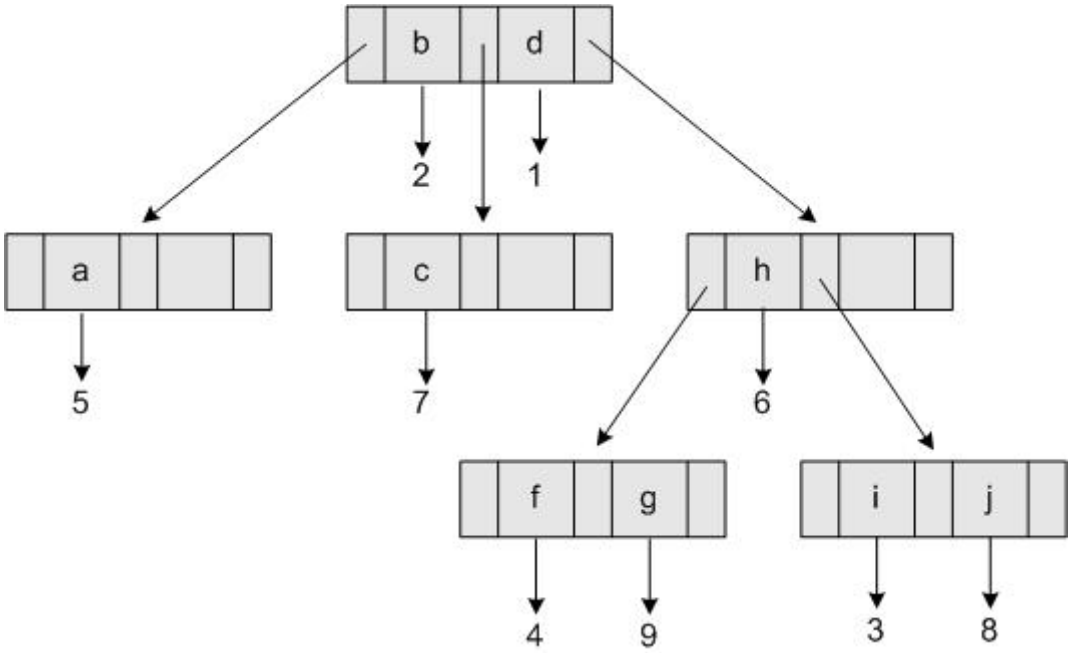
Η απαίτηση να είναι το δέντρο ισοσκελισμένο υπάρχει για να αποκλείσει την πιθανότητα να δημιουργηθούν στο δέντρο πολύ μεγάλα μονοπάτια, πράγμα που μπορεί να κάνει τη διάσχιση του δέντρου ιδιαίτερα χρονοβόρα σε ορισμένες περιπτώσεις, αφού όπως είναι γνωστό το κόστος της διάσχισης ενός μονοπατιού σε ένα δέντρο είναι ίσο με το μήκος του μονοπατιού αυτού. Για τον ίδιο λόγο υπάρχει ακόμη η απαίτηση ο κάθε κόμβος του δέντρου να είναι τουλάχιστον μισογεμάτος, δηλαδή αν το μέγιστο πλήθος κλειδιών που χωράνε σε έναν κόμβο είναι n, ανά πάσα στιγμή όλοι οι κόμβοι να έχουν τουλάχιστον $\lceil n/2 \rceil$ κλειδιά. Από τον περιορισμό αυτό εξαιρείται μόνο η ρίζα του δέντρου, η οποία όμως πρέπει να περιέχει τουλάχιστον δύο δείκτες, εκτός αν το δέντρο αποτελείται μόνο από έναν κόμβο.

2.1.3.2 Επεξεργασία ερωτημάτων

Τα ερωτήματα που κάνουμε συνήθως σε Β-Δέντρα συνίστανται στο να βρούμε τη δοσοληψία (ή τις δοσοληψίες) που αντιστοιχεί σε κάποιο συγκεκριμένο κλειδί από αυτά που υπάρχουν μέσα στο Β-Δέντρο, και αυτό είναι και το βασικό ερώτημα που θα μας απασχολήσει στα επόμενα τμήματα αυτής της εργασίας. Για να απαντήσουμε σε ένα τέτοιο ερώτημα αρκεί να ξεκινήσουμε από τη ρίζα του δέντρου και για κάθε κόμβο που συναντάμε να ελέγξουμε κατ' αρχήν αν το κλειδί που ψάχνουμε βρίσκεται μέσα στον κόμβο αυτό και αν όχι, να βρούμε ανάμεσα σε ποια δύο κλειδιά του τρέχοντος κόμβου θα έπρεπε να βρίσκεται το κλειδί που ψάχνουμε, ούτως ώστε να ακολουθήσουμε τον αντίστοιχο δείκτη που βρίσκεται ανάμεσα στα δύο αυτά κλειδιά προς έναν κόμβο-παιδί του τρέχοντα κόμβου, στον οποίο θα επαναλάβουμε την ίδια διαδικασία. Αν κάποια στιγμή φτάσουμε σε ένα φύλλο του δέντρου χωρίς να έχουμε βρει ακόμη το κλειδί που ψάχνουμε, αυτό σημαίνει ότι το συγκεκριμένο κλειδί δε βρίσκεται μέσα στο δέντρο, αφού αν βρισκόταν θα έπρεπε να είναι στο μονοπάτι το οποίο μόλις διατρέξαμε.

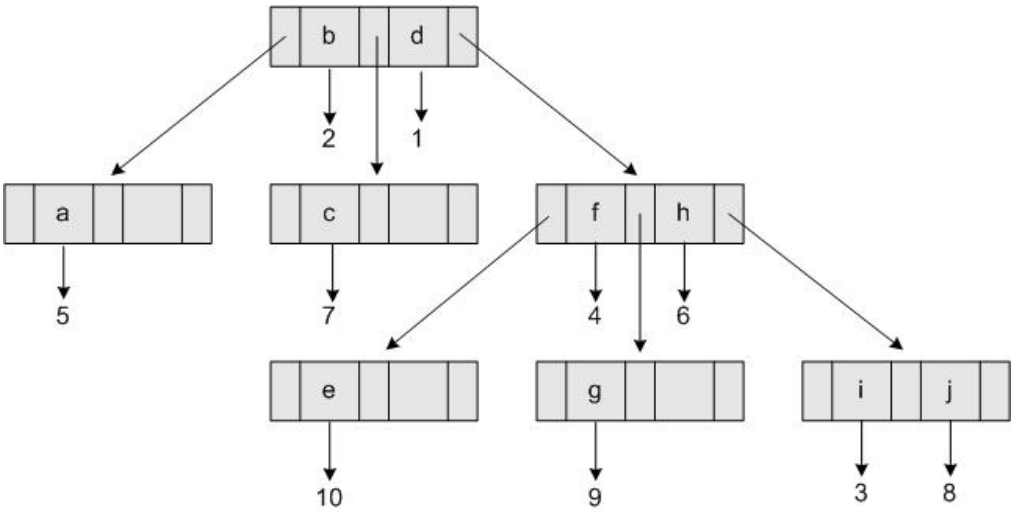
2.1.3.3 Ανανέωση

Για να εισάγουμε μία νέα δοσοληψία στη βάση και άρα μία νέα τιμή στο Β-Δέντρο διασχίζουμε το δέντρο από τη ρίζα για να βρούμε τον κόμβο στον οποίο πρέπει να εισαχθεί η νέα τιμή. Αν η νέα τιμή υπάρχει ήδη στον κόμβο αυτό, και υπό την προϋπόθεση ότι επιτρέπουμε την ύπαρξη πολλαπλών εγγραφών, προσθέτουμε έναν επιπλέον δείκτη προς τη νέα δοσοληψία της βάσης. Κάποιες υλοποιήσεις πιθανώς αντί για την προσθήκη ενός επιπλέον δείκτη εισάγουν για δεύτερη φορά την τιμή που μας ενδιαφέρει στο δέντρο. Αν τώρα η τιμή που θέλουμε να εισάγουμε δεν υπάρχει και υπάρχει χώρος στον κόμβο για την εισαγωγή ενός νέου κλειδιού, εισάγουμε τη νέα τιμή έτσι ώστε τα κλειδιά του κόμβου να παραμείνουν ταξινομημένα, και προσθέτουμε ένα δείκτη από τη νέα εγγραφή στο δέντρο στη νέα δοσοληψία. Στο σχήμα 2.2. φαίνεται πώς γίνεται το Β-Δέντρο του σχήματος 2.1. μετά την εισαγωγή στη βάση της νέας δοσοληψίας [9, g].



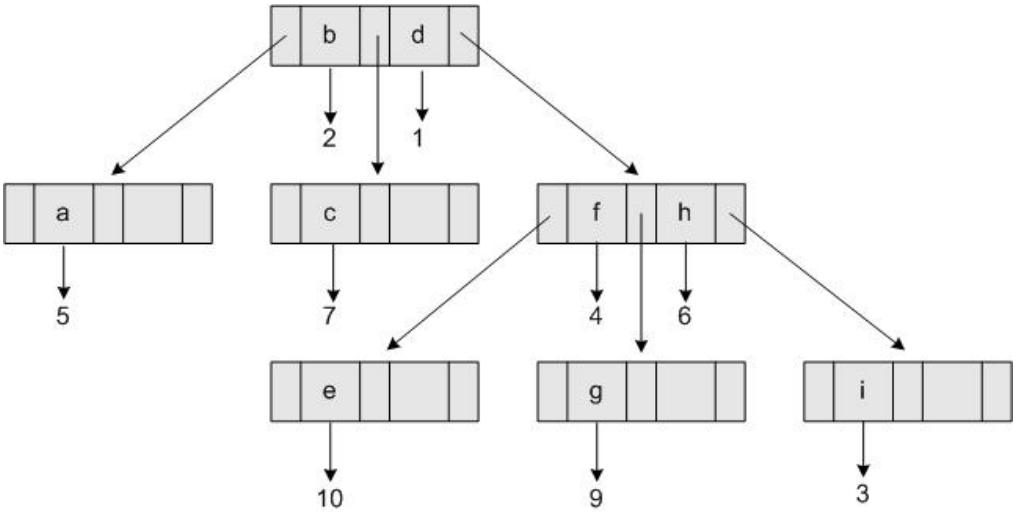
Σχήμα 2.2.: Το Β-Δέντρο του σχήματος 2.1. μετά την εισαγωγή της δοσοληψίας [9, g]

Στην περίπτωση που δεν υπάρχει αρκετός χώρος στον κόμβο στον οποίο πρέπει να εισαχθεί το νέο κλειδί διαιρούμε τον κόμβο αυτό σε δύο καινούργιους κόμβους, μετακινώντας το μεσαίο κλειδί του στον κόμβο-πατέρα και προσθέτοντας δείκτες προς τους δύο νέους κόμβους. Εάν ούτε ο κόμβος-πατέρας έχει χώρο για την εισαγωγή του νέου κλειδιού θα διαιρεθεί κι αυτός με τη σειρά του κ.ο.κ. Στη χειρότερη περίπτωση οι διαιρέσεις θα συνεχιστούν μέχρι να φτάσουμε στη ρίζα οπότε με τη διαίρεση της ρίζας το βάθος του δέντρου θα αυξηθεί κατά 1. Στο σχήμα 2.3. που ακολουθεί φαίνεται το Β-Δέντρο του σχήματος 2.2. μετά την εισαγωγή της δοσοληψίας [10, e], η οποία έχει ως αποτέλεσμα τη διαίρεση ενός κόμβου του δέντρου.



Σχήμα 2.3.: Το Β-Δέντρο του σχήματος 2.2. μετά την εισαγωγή της δοσοληψίας [10, e]

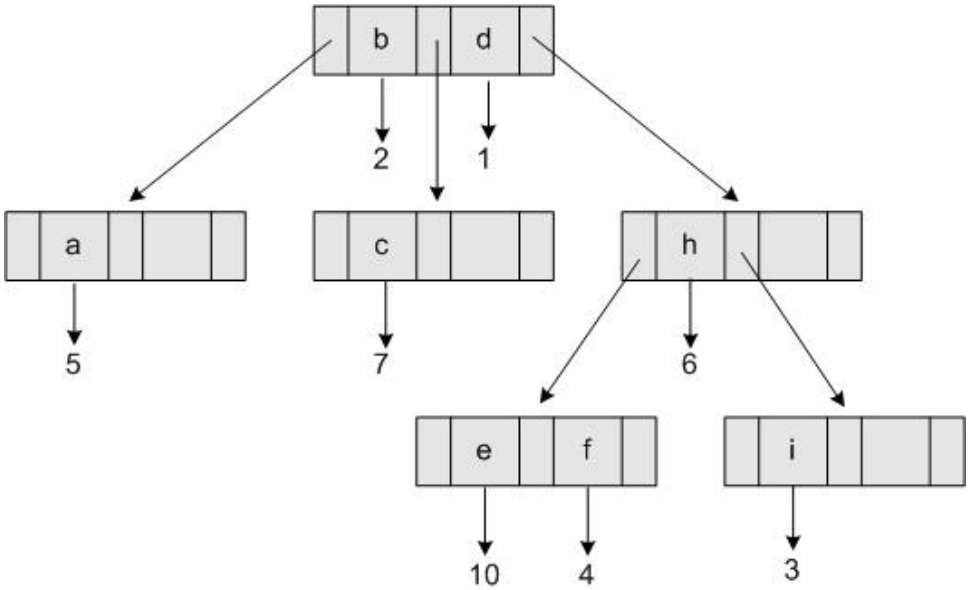
Για να διαγράψουμε μία δοσοληψία από τη βάση και άρα και από το Β-Δέντρο αρκεί να βρούμε διασχίζοντας το δέντρο τον κόμβο στον οποίο είναι αποθηκευμένο το κλειδί που αντιστοιχεί σε αυτή τη δοσοληψία και να το διαγράψουμε. Στο σχήμα 2.4. φαίνεται το Β-Δέντρο του σχήματος 2.3. μετά τη διαγραφή της δοσοληψίας [8, j].



Σχήμα 2.4.: Το Β-Δέντρο του σχήματος 2.3. μετά τη διαγραφή της δοσοληψίας [8, j]

Στην περίπτωση που στον κόμβο από τον οποίο διαγράφουμε το κλειδί μένουν λιγότερα από $\lceil n/2 \rceil$ κλειδιά μετά τη διαγραφή, πράγμα το οποίο απαγορεύεται, τότε, αν ο κόμβος αυτός είναι εσωτερικός κόμβος του δέντρου, βρίσκουμε από τα παιδιά του το αμέσως μεγαλύτερο κλειδί από αυτό που διαγράψαμε και το μεταφέρουμε στον τρέχοντα κόμβο ούτως ώστε αυτός να αποκτήσει ξανά τουλάχιστον $\lceil n/2 \rceil$ κλειδιά. Αν πάλι ο κόμβος από τον οποίο διαγράφουμε το κλειδί είναι φύλλο, τότε κοιτάμε κατ' αρχήν τους κόμβους που βρίσκονται αμέσως δεξιά και αριστερά από αυτόν και στην περίπτωση που κάποιος από αυτούς έχει τουλάχιστον $\lceil n/2 \rceil$+1 κλειδιά μεταφέρουμε ένα κλειδί (μαζί με τον αντίστοιχο δείκτη) στον

τρέχοντα κόμβο. Αν πάλι αυτό δε συμβαίνει, τότε πρέπει να συνενώσουμε τον τρέχοντα κόμβο με έναν από τους δύο διπλανούς του, πράγμα το οποίο γίνεται με το να μετακινήσουμε προς τα κάτω το κλειδί του κόμβου-πατέρα που βρίσκεται ανάμεσα στα κλειδιά των κόμβων που θα συνενωθούν. Στο σχήμα 2.5. φαίνεται ενδεικτικά το δέντρο του σχήματος 2.4. μετά τη διαγραφή της δοσοληψίας [9, g].



Σχήμα 2.5.: Το Β-Δέντρο του σχήματος 2.4. μετά τη διαγραφή της δοσοληψίας [9, g]

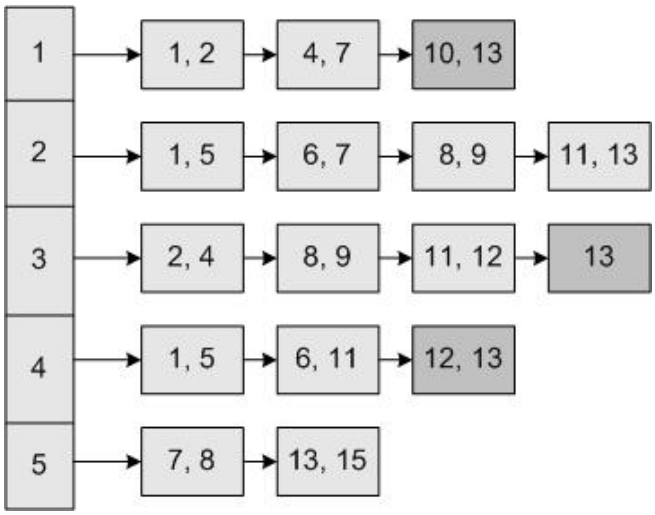
2.2 Σύγκριση - Στόχος

Οι δομές που παρουσιάστηκαν παραπάνω είναι, με εξαίρεση το Β-Δέντρο, οι ευρύτερα διαδεδομένες δομές ευρετηρίου για τιμές-σύνολα. Παρ' όλα αυτά η κάθε μία από αυτές έχει ένα ή περισσότερα μειονεκτήματα τα οποία μπορούν να μειώσουν σημαντικά την απόδοσή τους σε συγκεκριμένες εφαρμογές.

Το signature file έχει το σημαντικό μειονέκτημα των false matches τα οποία μας αναγκάζουν να προσπελάσουμε όλες τις δοσοληψίες που το ευρετήριο μας δίνει ως υποψήφια αποτελέσματα για να δούμε αν πράγματι πληρούν τις προϋποθέσεις του ερωτήματος. Επιπλέον, για την αποτίμηση των ερωτημάτων ή την ανανέωση του ευρετηρίου, απαιτείται η σειριακή ανάγνωση του signature file, η οποία, αν και αντιμετωπίζεται μερικώς με τεχνικές όπως το bitslicing, εξακολουθεί να παραμένει πρόβλημα, αποτελώντας το λόγο που οδήγησε σε διάφορες επεκτάσεις της δομής οι οποίες εξετάστηκαν παραπάνω. Τέλος, το signature file υστερεί σημαντικά σε βάσεις με δοσοληψίες τα μεγέθη των οποίων διαφέρουν κατά πολύ, αφού οι μεγάλες δοσοληψίες απαιτούν και μεγάλες υπογραφές για να ελαχιστοποιηθεί η πιθανότητα για false matches, με αποτέλεσμα να καταναλώνεται έτσι περισσότερος χώρος για τις υπογραφές των μικρών δοσοληψιών απ' όσο θα ήταν απαραίτητο, αφού όλες οι υπογραφές πρέπει να έχουν σταθερό μήκος.

Συγκριτικά με τα inverted files, σχετικές εργασίες έχουν δείξει ότι το μέγεθος ενός signature file πάνω σε μία βάση μπορεί να είναι και διπλάσιο του μεγέθους του αντίστοιχου inverted file αφού το μέγεθος ενός inverted file έχει υπολογιστεί στο 15% του μεγέθους της βάσης ενώ το μέγεθος ενός κατατμημένου signature file στο 25%-40%. Το inverted file βέβαια απαιτεί ένα τμήμα του (το λεξικό) να βρίσκεται στην κύρια μνήμη κατά την αποτίμηση των ερωτημάτων αλλά το μέγεθος του λεξικού είναι συνήθως αμελητέο σε σχέση με τα μεγέθη των μνημών που έχουν αναπτυχθεί τα τελευταία χρόνια. Επιπλέον το inverted file κατασκευάζεται συνήθως γρηγορότερα από το signature file, αλλά αν και, όπως είπαμε προηγουμένως, η ανανέωση του signature file απαιτεί σειριακή ανάγνωση των αρχείων που το αποτελούν, η ανανέωση του inverted file χρειάζεται ίσως μεγαλύτερη προσοχή για το χειρισμό των λιστών στο δίσκο το μέγεθος των οποίων είναι δυναμικό και άρα μεταβάλλεται. Η δυσκολία αυτή ξεπερνιέται συνήθως με μαζικές ανανεώσεις του inverted file ούτως ώστε να μην αλλάζει το μέγεθος των inverted lists συνεχώς.

Από τα παραπάνω φαίνεται ότι από τα είδη υπάρχοντα ευρετήρια το πιο αποδοτικό για χρήση με τα δεδομένα που μας ενδιαφέρουν είναι το inverted file. Όπως όμως κάποιος μπορεί να δει από τον τρόπο αποτίμησης των ερωτημάτων που περιγράφηκε προηγουμένως, αν και, σε αντίθεση με το signature file, δε χρειάζεται να διαβαστεί σειριακά ολόκληρο το ευρετήριο διαβάζονται ολόκληρες (ή σχεδόν ολόκληρες) οι λίστες των αντικειμένων που περιέχονται στο query set. Αν λοιπόν έχουμε για παράδειγμα το inverted file του σχήματος 2.9. και κάνουμε ένα ερώτημα το αποτέλεσμα του οποίου είναι η δοσοληψία με id 13, θα χρειαστεί να διαβάσουμε ολόκληρες τις λίστες των στοιχείων 1, 3 και 4, δηλαδή θα χρειαστεί να διαβαστούν 10 σελίδες ενώ τα αποτελέσματα είναι συγκεντρωμένα στις μόλις 3 σελίδες που φαίνονται με σκούρο χρώμα.



Σχήμα 2.6.: Παράδειγμα inverted file

Έτσι, ο στόχος της εργασίας αυτής είναι να επεκταθεί το inverted file έτσι ώστε να μειωθούν οι σελίδες που προσπελαύνονται κατά την αποτίμηση των ερωτημάτων, με το να μη

χρειάζεται να διατρέξουμε ολόκληρες τις λίστες των αντικειμένων που μας ενδιαφέρουν αλλά μόνο τμήματα αυτών. Αυτό γίνεται συνδυάζοντας το inverted file με το Β-Δέντρο, το οποίο θα μας βοηθήσει να αποκτήσουμε πρόσβαση σε συγκεκριμένα τμήματα των λιστών, όπως θα περιγραφεί στα επόμενα κεφάλαια. Στη συνέχεια γίνονται πειράματα που συγκρίνουν την απόδοση του νέου ευρετηρίου με το inverted file ούτως ώστε να αποδειχθεί ότι η μέθοδός μας είναι πράγματι αποτελεσματική.

3

Θεωρητική Περιγραφή Ευρετηρίου

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, στα πλαίσια της εργασίας αυτής αναπτύσσεται μία νέα δομή, η οποία συνδυάζει το inverted file με το Β-Δέντρο, σε μία προσπάθεια να γίνει πιο αποδοτικό το inverted file στην αποτίμηση των ερωτημάτων. Η δομή αυτή ονομάστηκε Ordered Inverted File και παρουσιάζεται λεπτομερώς στο κεφάλαιο αυτό.

Ο στόχος που προσπαθούμε να πετύχουμε με την ανάπτυξη του ordered inverted file είναι, όπως προείπαμε, να αποκτήσουμε πρόσβαση σε τμήματα των λιστών του inverted file χωρίς να χρειάζεται να διασχίσουμε ολόκληρες τις λίστες. Για το λόγο αυτό μετασχηματίζουμε το κλασσικό inverted file που παρουσιάστηκε προηγουμένως ως εξής:

- Ταξινομούμε τα αντικείμενα που περιέχονται στην κάθε δοσοληψία με βάση τη συχνότητα εμφάνισής τους.
- Ταξινομούμε τις δοσοληψίες μέσα στην κάθε ανεστραμμένη λίστα με βάση τη συχνότητα των αντικειμένων που περιέχουν. Η χρησιμότητα της ταξινόμησης αυτής θα φανεί κατά την ανάπτυξη των μεθόδων αποτίμησης των ερωτημάτων.
- Προσθέτουμε πάνω από κάθε ανεστραμμένη λίστα ένα Β-Δέντρο με κλειδιά την τελευταία δοσοληψία κάθε σελίδας της λίστας και δείκτες στις σελίδες της λίστας. Οι δείκτες αυτοί του Β-Δέντρου θα μας βοηθήσουν όπως θα δούμε αργότερα να αποκτήσουμε πρόσβαση σε συγκεκριμένες σελίδες των λιστών.

Θεωρούμε επίσης ότι πάνω από το λεξικό του inverted file έχει κατασκευαστεί ένα hash ευρετήριο με αποτέλεσμα να μπορούμε να βρούμε οποιοδήποτε αντικείμενο του λεξικού σε ένα μόνο βήμα, χωρίς να χρειαστεί να διατρέξουμε σειριακά όλο το λεξικό ή τμήμα του.

Στις επόμενες ενότητες παρουσιάζεται ο τρόπος κατασκευής του ordered inverted file, η αποτίμηση ερωτημάτων με βάση αυτό, καθώς και η συμπίεση που μπορεί να γίνει σε αυτό και οι μεταβολές που απαιτούνται κατά την ανανέωση της βάσης.

3.1 Κατασκευή

Έστω η βάση του πίνακα 3.1. πάνω στην οποία επιθυμούμε να κατασκευάσουμε ένα ordered inverted file.

ID	Items
1	a, c, e, f, g
2	a, b, f, j
3	a, c, d, e, j
4	b, d, h, j
5	c, d, e, j
6	a, b, c, e, g, i
7	a, b, f, h
8	e, g, h, j
9	b, e, g
10	a, c, e, f, h, i

Πίνακας 3.1.: Παράδειγμα βάσης

Κατ' αρχήν πρέπει να ταξινομήσουμε τα αντικείμενα που εμφανίζονται στη βάση ανάλογα με τη συχνότητα εμφάνισής τους. Για το λόγο αυτό, μετονομάζουμε τα αντικείμενα της βάσης ονομάζοντας 1 το αντικείμενο με τη μεγαλύτερη συχνότητα, 2 το αμέσως επόμενο, κ.ο.κ. Η διαδικασία αυτή φαίνεται στον πίνακα 3.2.

Αντικείμενο	Συχνότητα	Νέο Αναγνωριστικό
a	6	2
b	5	5
c	5	3
d	3	9
e	7	1
f	4	8
g	4	7
h	4	6
i	2	10
j	5	4

Πίνακας 3.2.: Συχνότητες και νέα ονόματα των αντικειμένων της βάσης του πίνακα 3.1.]

Έτσι, ταξινομώντας τα αντικείμενα στο εσωτερικό της κάθε δοσοληψίας κατά φθίνουσα συχνότητα, η βάση του πίνακα 3.1. παίρνει τη μορφή που φαίνεται στον πίνακα 3.3.

ID	Items
1	1, 2, 3, 7, 8
2	2, 4, 5, 8
3	1, 2, 3, 4, 9
4	4, 5, 6, 9
5	1, 3, 4, 9
6	1, 2, 3, 5, 7, 10
7	2, 5, 6, 8
8	1, 4, 6, 7
9	1, 5, 7
10	1, 2, 3, 6, 8, 10

Πίνακας 3.3.: Η βάση του πίνακα 3.1. με τα αντικείμενα διατεταγμένα κατά φθίνουσα συχνότητα

Προφανώς τα νέα αναγνωριστικά που δώσαμε στα αντικείμενα δεν επηρεάζουν τις απαντήσεις που το ευρετήριό μας δίνει στα διάφορα ερωτήματα, αφού η κάθε δοσοληψία εξακολουθεί να περιέχει τα ίδια αντικείμενα, και έγινε για χάρη ευκολίας, για την ευχερή διάκριση της διάταξης των αντικειμένων με βάση τη συχνότητά τους που χρησιμοποιείται στο ordered inverted file. Προφανώς θα μπορούσαν να διατηρηθούν τα αρχικά ονόματα των αντικειμένων, και η διάταξή τους που εδώ γίνεται με τους τελεστές σύγκρισης των ακεραίων να γίνει με άλλες συναρτήσεις. Για το λόγο αυτό δεν υπολογίζουμε ως επιπλέον κόστος το κόστος της μετονομασίας των αντικειμένων κατά την κατασκευή του ή της «μετάφρασης» των ερωτημάτων ώστε να χρησιμοποιούν αυτά τα ονόματα κατά την υποβολή τους στο ευρετήριο. Στις επόμενες ενότητες θα χρησιμοποιούμε λοιπόν μόνο αυτά τα νέα, συμβολικά ονόματα και θα θεωρούμε ότι τα ερωτήματα γίνονται με χρήση αυτών.

Πρέπει επίσης να σημειώσουμε ότι οι δοσοληψίες μας παραμένουν (αταξινόμητα) σύνολα δεδομένων και ότι η ταξινόμηση των αντικειμένων που κάναμε γίνεται εσωτερικά στο ordered inverted file κατά την κατασκευή του και χρησιμοποιείται μόνο για λόγους διάταξης των δοσοληψιών στις λίστες. Η ταξινόμηση αυτή δεν εμφανίζεται εκτός του ordered inverted file και έτσι για το χρήστη του ευρετηρίου είναι σα να μην υπάρχει.

Στη συνέχεια, διατάσσουμε τις δοσοληψίες μέσα στη βάση ανάλογα με τη συχνότητα των αντικειμένων που περιέχει η καθεμία, με αποτέλεσμα η βάση να γίνει όπως φαίνεται στον πίνακα 3.4.

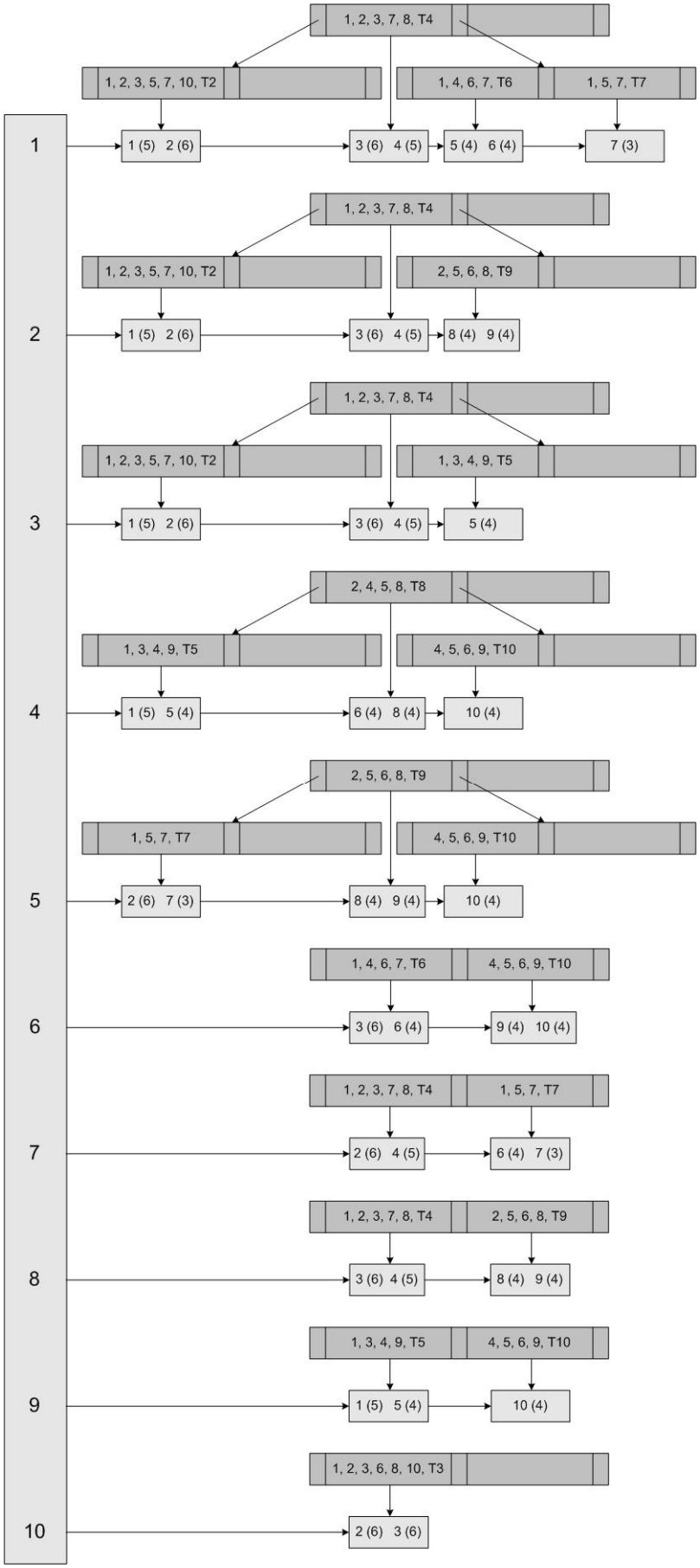
ID	Items
1	1, 2, 3, 4, 9
2	1, 2, 3, 5, 7, 10
3	1, 2, 3, 6, 8, 10
4	1, 2, 3, 7, 8
5	1, 3, 4, 9
6	1, 4, 6, 7
7	1, 5, 7
8	2, 4, 5, 8
9	2, 5, 6, 8
10	4, 5, 6, 9

Πίνακας 3.4.: Η βάση του πίνακα 3.3. με τις δοσοληψίες διατεταγμένες ανάλογα με τη συχνότητα των αντικειμένων τους

Η αλλαγή των ids των δοσοληψιών που φαίνεται στον πίνακα 3.4., αν και θα μπορούσε να επηρεάσει τα αποτελέσματα των ερωτημάτων, στην πραγματικότητα δεν το κάνει γιατί παρόλο που οι δοσοληψίες αποθηκεύονται στο ευρετήριο με βάση το id τους, ανακτώνται όταν χρειαστεί με βάση τη θέση τους στο δίσκο, η οποία βρίσκεται από έναν ενδιάμεσο πίνακα (intermediate table) που συσχετίζει το κάθε id με μία διεύθυνση στο δίσκο. Έτσι, τα νέα ids δεν έχουν επίπτωση στα αποτελέσματα εφόσον ο ενδιάμεσος πίνακας ενημερωθεί σωστά, αφού οι θέσεις των δοσοληψιών στο δίσκο παραμένουν αμετάβλητες. Εδώ επίσης πρέπει να σημειώσουμε ότι ο ενδιάμεσος πίνακας δε χρειάζεται να περιέχει τα ids των δοσοληψιών παρά μόνο τις θέσεις τους στο δίσκο, υπό τον όρο ότι αποθηκεύει τις διευθύνσεις με τη σειρά. Έτσι, η διεύθυνση της δοσοληψίας με id=1 βρίσκεται στην πρώτη θέση του πίνακα, αυτή της δοσοληψίας με id=2 στη δεύτερη, κ.ο.κ.

Αφού μετασχηματίσουμε τα δεδομένα μας όπως περιγράφηκε παραπάνω, διαβάζουμε τις δοσοληψίες με τη σειρά και τις εισάγουμε σε ένα inverted file. Έτσι, για κάθε δοσοληψία που διαβάζουμε εισάγουμε το id και το μήκος της στις ανεστραμμένες λίστες που αντιστοιχούν στα αντικείμενα που αυτή περιέχει. Παράλληλα, κατασκευάζουμε και ένα B-Δέντρο πάνω από κάθε λίστα το οποίο έχει δείκτες προς όλες τις σελίδες της λίστας, όπως αναφέρθηκε και προηγουμένως. Τα κλειδί που αντιστοιχεί σε κάθε δείκτη του B-Δέντρου είναι η τελευταία δοσοληψία της σελίδας στην οποία δείχνει ο δείκτης. Σε κάθε κλειδί αποθηκεύονται τα περιεχόμενα της δοσοληψίας αυτής, αφού με βάση αυτά γίνεται η αναζήτηση στο ευρετήριο, όπως επίσης και το id της δοσοληψίας, το οποίο είναι μοναδικό, ούτως ώστε να αποφύγουμε την πιθανότητα να εμφανιστούν δύο ίδια κλειδιά στο δέντρο.

Στο σχήμα 3.1. μπορούμε να δούμε το ordered inverted file που κατασκευάζεται με τον τρόπο που μόλις περιγράψαμε για τη βάση του πίνακα 3.4. Θεωρούμε ότι η κάθε σελίδα των λιστών χωράει μέχρι δύο δοσοληψίες και η κάθε σελίδα των B-Δέντρων χωράει μέχρι δύο κλειδιά. Τα ids των δοσοληψιών συμβολίζονται στα B-Δέντρα με το πρόθεμα T ενώ τα μήκη των δοσοληψιών που είναι αποθηκευμένα στις λίστες βρίσκονται μέσα σε παρενθέσεις.



Σχήμα 3.1.: Το Ordered Inverted File της βάσης του πίνακα 3.4.

Όπως μπορεί να φανεί και στο σχήμα 3.1., χάρη στη διάταξη των αντικειμένων και των δοσοληψιών που κάναμε πριν την κατασκευή του inverted file και τη σειριακή διάσχιση της βάσης που κάνουμε κατά την κατασκευή, οι δοσοληψίες βρίσκονται τοποθετημένες σε κάθε λίστα με την ίδια σειρά που τις έχουμε διατάξει μέσα στη βάση (κατά αύξον id). Επιπλέον οι δοσοληψίες με τα πιο συχνά αντικείμενα συναντώνται στην αρχή των λιστών ενώ αυτές με τα λιγότερο συχνά στο τέλος.

Εδώ πρέπει να πούμε ότι για την κατασκευή των Β-Δέντρων υπάρχουν δύο δυνατές προσεγγίσεις. Η πρώτη είναι κάθε φορά που γεμίζει μία σελίδα κάποιας ανεστραμμένης λίστας να προσθέτουμε το αντίστοιχο κλειδί στο Β-Δέντρο εκείνης της λίστας και αφού τελειώσει η κατασκευή του inverted file να προσθέσουμε τα κλειδιά που αντιστοιχούν στις τελευταίες εγγραφές των σελίδων που δε γέμισαν τελείως. Η δεύτερη είναι να κατασκευάσουμε πρώτα το inverted file και μετά να διατρέξουμε μία-μία τις λίστες κατασκευάζοντας τα αντίστοιχα Β-Δέντρα. Προφανώς προτιμάται η πρώτη προσέγγιση η οποία είναι πολύ πιο αποδοτική αφού δεν απαιτεί τη εκ νέου διάσχιση ολόκληρου του inverted file μετά την κατασκευή του, η οποία για μεγάλες βάσεις μπορεί να είναι πολύ χρονοβόρα.

3.2 Αποτίμηση ερωτημάτων

Η αποτίμηση των ερωτημάτων με χρήση του ordered inverted file γίνεται με τρόπο αντίστοιχο με αυτό που χρησιμοποιεί το κλασσικό inverted file. Έτσι, για να απαντήσουμε σε ένα ερώτημα πρέπει να προσπελάσουμε τις λίστες που αντιστοιχούν στους διάφορους όρους του ερωτήματος και να κάνουμε την κατάλληλη πράξη μεταξύ τους, ανάλογα με το είδος του ερωτήματος. Η διαφορά είναι ότι τα Β-Δέντρα που βρίσκονται πάνω από τις λίστες μας βοηθούν να αποκτήσουμε πρόσβαση στα τμήματα αυτών όπου βρίσκονται οι απαντήσεις στα ερωτήματά μας, χωρίς να χρειάζεται να τις διατρέξουμε από την αρχή. Τα τμήματα αυτά στο εξής θα τα ονομάζουμε περιοχές πιθανών απαντήσεων αφού εάν υπάρχουν απαντήσεις στο ερώτημα το οποίο αποτιμούμε κάθε φορά, τότε αυτές θα βρίσκονται μέσα σε αυτά. Στη συνέχεια παρουσιάζεται η μέθοδος που ακολουθούμε για να απαντήσουμε στα είδη των ερωτημάτων που μας απασχολούν σε αυτή την εργασία, δηλαδή στα equality queries, στα subset queries και στα superset queries.

3.2.1 Equality queries

Έστω ένα equality query, το query set του οποίου είναι το $qs = \{q_1, q_2, q_3, ..., q_n\}$ όπου τα q_1, q_2, …, q_n είναι διατεταγμένα κατά φθίνουσα συχνότητα εμφάνισης στη βάση. Στη γενική περίπτωση βέβαια αυτό δε συμβαίνει και για το λόγο αυτό, όπως θα δούμε και στο επόμενο

κεφάλαιο όπου εξετάζουμε την υλοποίηση του ευρετηρίου, πριν την επεξεργασία ενός ερωτήματος διατάσσουμε τους όρους του με βάση τη συχνότητά τους.

Για να απαντήσουμε στο ερώτημα αυτό πρέπει να βρούμε όλες τις δοσοληψίες που περιέχουν ακριβώς το qs. Για να το κάνουμε αυτό με βάση ένα απλό inverted file θα έπρεπε να πάρουμε την τομή των λιστών των αντικειμένων που περιέχονται στο qs, απαιτώντας επιπλέον τα αποτελέσματα να έχουν ίδιο μήκος με το ερώτημα, δηλαδή ίσο με n. Στο ordered inverted file όμως, μπορούμε να κάνουμε επιπλέον την ακόλουθη παρατήρηση. Εφόσον οι δοσοληψίες είναι ταξινομημένες μέσα στην κάθε λίστα με βάση τη συχνότητα των αντικειμένων τους και τα αντικείμενα είναι ταξινομημένα μέσα στην κάθε δοσοληψία επίσης με βάση τη συχνότητά τους, αν υπάρχουν πολλές δοσοληψίες στην κάθε λίστα που να περιέχουν ακριβώς το query set, τότε αυτές θα βρίσκονται συγκεντρωμένες στο ίδιο σημείο χωρίς άλλες δοσοληψίες ανάμεσά τους. Επιπλέον, θα βρίσκονται μετά την τελευταία δοσοληψία με περιεχόμενα τα $\{q_1, q_2, q_3, ..., q_n - 1\}$ και πριν την πρώτη δοσοληψία με περιεχόμενα τα $\{q_1, q_2, q_3, ..., q_n + 1\}$, υπό τον όρο βέβαια ότι οι δοσοληψίες αυτές υπάρχουν.

Έχοντας κατά νου αυτή την παρατήρηση, γίνεται πλέον φανερό ότι δεν είναι απαραίτητο να ψάξουμε ολόκληρες τις ανεστραμμένες λίστες των αντικειμένων του ερωτήματος για πιθανές απαντήσεις, αλλά μόνο τα τμήματα αυτών που βρίσκονται ανάμεσα στην πρώτη δοσοληψία με περιεχόμενα τα $\{q_1, q_2, q_3, ..., q_n\}$ και στην πρώτη δοσοληψία με περιεχόμενα τα $\{q_1, q_2, q_3, ..., q_n + 1\}$. Για να το κάνουμε αυτό χρησιμοποιούμε το B-Δέντρο της κάθε λίστας για να βρούμε σε ποια σελίδα της λίστας βρίσκεται η καθεμία από αυτές τις δοσοληψίες, με σκοπό να ψάξουμε μόνο ανάμεσα στις δύο αυτές σελίδες που πήραμε ως αποτέλεσμα. Οι δύο αυτές σελίδες αποτελούν έτσι τα όρια της περιοχής πιθανών απαντήσεων.

Η συνάρτηση εύρεσης που χρησιμοποιούμε διασχίζει το B-Δέντρο ξεκινώντας από τη ρίζα και ακολουθώντας κάθε φορά το κατάλληλο μονοπάτι ψάχνει να βρει το κλειδί με τη ζητούμενη δοσοληψία και επιστρέφει τον αντίστοιχο δείκτη. Στην περίπτωση που η δοσοληψία που ψάχνουμε δε βρίσκεται μέσα στο B-Δέντρο – μπορεί να μη βρίσκεται στη λίστα ή να μην είναι η τελευταία δοσοληψία κάποιας σελίδας και άρα να μην έχει εισαχθεί στο B-Δέντρο – τότε επιστρέφει το δείκτη που αντιστοιχεί στην αμέσως επόμενη δοσοληψία που υπάρχει στο δέντρο, η οποία θα είναι η τελευταία δοσοληψία της σελίδας που περιέχει τη δοσοληψία που ψάχνουμε με αποτέλεσμα να καταλήξουμε τελικά στην επιθυμητή σελίδα.

Καθώς λοιπόν το κάθε κλειδί των B-Δέντρων περιέχει τα περιεχόμενα της δοσοληψίας στην οποία αντιστοιχεί αλλά και το id της, για να βρούμε την πρώτη δοσοληψία με κάποιο συγκεκριμένο περιεχόμενο ψάχνουμε για την δοσοληψία με αυτό το περιεχόμενο και id=0. Μία τέτοια δοσοληψία προφανώς δεν πρόκειται ποτέ να βρεθεί αφού όλα τα ids στη βάση μας είναι μεγαλύτερα ή ίσα με το 1, αλλά αφού η συνάρτηση που μόλις περιγράψαμε βρίσκει

την αμέσως επόμενη δοσοληψία αν αυτή που ψάχνουμε δεν υπάρχει, θα πάρουμε τελικά το επιθυμητό αποτέλεσμα.

Έτσι, για να βρούμε τα αποτελέσματα του ερωτήματος, παίρνουμε την τομή των σελίδων που βρίσκονται ανάμεσα στα δύο παραπάνω όρια στις λίστες των αντικειμένων που περιέχονται στο qs και απαιτούμε επιπλέον οι δοσοληψίες που θα συμπεριληφθούν στο αποτέλεσμα να έχουν μήκος ίσο με αυτό του qs. Η τομή των λιστών γίνεται με χρήση του αλγορίθμου MergeJoin η λειτουργία του οποίου εμφανίζεται σε ψευδοκώδικα στον κώδικα 3.1., όπου RoI (Region of Interest) η περιοχή πιθανών απαντήσεων της κάθε λίστας.

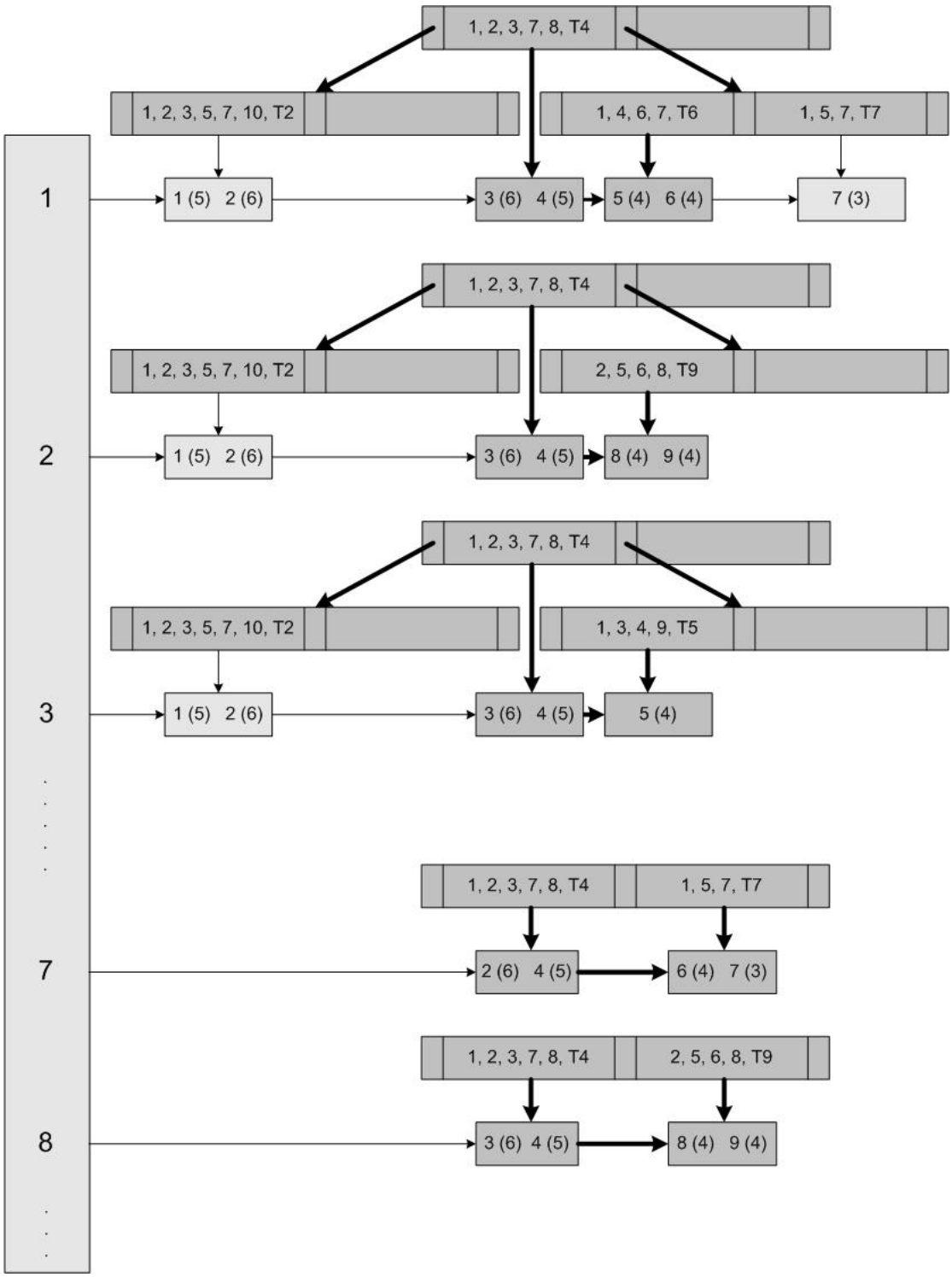
```
for each i in RoI of qn.list with (i.length=qs.length) {
    for (j=n-1; j≥1; j--) {
          found=false;
          get next item c from oj.list;
          while (c<i) {
                if we reached the enf of the list end;
                 get next item c from oj.list;
                 if (c=i) then found=true;
          }
          if (not found) break;
    }
    if (found) output i;
}
```

Κώδικας 3.1.: Η λειτουργία της συνάρτησης MergeJoin για τον προσδιορισμό των απαντήσεων ενός equality query

Εδώ μπορούμε να δούμε ότι η συνάρτηση τερματίζει όταν κάποια από τις λίστες εξαντληθεί, οπότε οι δοσοληψίες που δεν έχουν ελεγχθεί ακόμη στις υπόλοιπες λίστες δε μπορούν να ανήκουν στην τομή των λιστών. Με αυτό τον τρόπο βλέπουμε ότι μπορεί σε κάποιες περιπτώσεις να μην προσπελάσουμε ολόκληρες τις περιοχές πιθανών απαντήσεων που έχουμε προσδιορίσει για κάποιο ερώτημα αλλά μόνο τμήματα αυτών.

Έστω για παράδειγμα η βάση του πίνακα 3.4. πάνω στην οποία δημιουργείται το ordered inverted file που φαίνεται στο σχήμα 3.1., στην οποία θέλουμε να κάνουμε ένα equality query με query set το $qs = \{1,2,3,7,8\}$. Η απάντηση στο ερώτημα αυτό είναι η δοσοληψία με id=4 όπως μπορούμε εποπτικά να δούμε από τον πίνακα 3.4. Για να απαντήσουμε στο ερώτημα με χρήση του ευρετηρίου θα ψάξουμε αρχικά στα B-Δέντρα των αντικειμένων 1, 2, 3, 7 και 8 για τα κλειδιά 1, 2, 3, 7, 8, T0 και 1, 2, 3, 7, 9, T0. Στη συνέχεια θα κάνουμε MergeJoin τα τμήματα των λιστών που βρίσκονται ανάμεσα στις σελίδες που θα πάρουμε από την παραπάνω αναζήτηση. Στο σχήμα 3.2. βλέπουμε με έντονους δείκτες και σκούρο χρώμα τη

διαδρομή που ακολουθούμε στα δέντρα και τις σελίδες των λιστών μέσα στις οποίες βρίσκονται οι πιθανές απαντήσεις. Για λόγους απλότητας του σχήματος εμφανίζονται μόνο τα τμήματα του ευρετηρίου που έχουν σχέση με τα αντικείμενα που μας ενδιαφέρουν.



Σχήμα 3.2.: Αποτίμηση equality query με $qs = \{1,2,3,7,8\}$

Ενδεικτικά εξετάζουμε τον τρόπο με τον οποίο γίνεται η προσπέλαση των σελίδων για το αντικείμενο 1. Αρχικά ψάχνουμε στο B-Δέντρο για το 1, 2, 3, 7, 8, T0. Ξεκινάμε από τη ρίζα

όπου βρίσκουμε το 1, 2, 3, 7, 8, T4 και προχωρούμε στο αριστερό παιδί της ρίζας αφού το κλειδί που ψάχνουμε προηγείται αυτού που βρήκαμε στη ρίζα. Εκεί βρίσκουμε μόνο ένα κλειδί, μικρότερο από αυτό που ψάχνουμε, και άρα συμπεραίνουμε ότι το 1, 2, 3, 7, 8, T0 δεν υπάρχει στο δέντρο και ότι το αμέσως μεγαλύτερο είναι το 1, 2, 3, 7, 8, T4. Έτσι, ακολουθώντας των αντίστοιχο δείκτη οδηγούμαστε στη δεύτερη σελίδα της ανεστραμμένης λίστας του αντικειμένου 1, η οποία αποτελεί έτσι την αρχή της περιοχής πιθανών απαντήσεων για τη λίστα αυτή. Στη συνέχεια ψάχνουμε πάλι στο B-Δέντρο για το 1, 2, 3, 7, 9, T0 αυτή τη φορά. Ξεκινάμε πάλι από τη ρίζα, το μοναδικό κλειδί της οποίας είναι μικρότερο από αυτό που ψάχνουμε και έτσι πηγαίνουμε στο δεξί παιδί. Το μικρότερο κλειδί που βρίσκουμε εκεί είναι το 1, 4, 6, 7, T6 το οποίο είναι συνεπώς το αμέσως επόμενο κλειδί από αυτό που ψάχνουμε (το οποίο δεν υπάρχει στο δέντρο) και με τον αντίστοιχο δείκτη οδηγούμαστε στην τρίτη σελίδα της ανεστραμμένης λίστας, η οποία θα είναι το τέλος της περιοχής πιθανών απαντήσεων. Έτσι, τα πιθανά αποτελέσματα στη λίστα αυτή θα βρίσκονται από τη δεύτερη μέχρι και την τρίτη σελίδα. Με τον ίδιο τρόπο εργαζόμαστε και για τα άλλα αντικείμενα. Παίρνοντας τέλος την τομή των περιοχών πιθανών απαντήσεων που προέκυψαν και απαιτώντας τα αποτελέσματά μας να έχουν μήκος ίσο με 5 βρίσκουμε πράγματι σα μοναδικό αποτέλεσμα τη δοσοληψία με id=4.

3.2.2 Subset queries

Έστω ένα subset query αυτή τη φορά με query set το $qs = \{q_1, q_2, q_3, ..., q_n\}$ τα αντικείμενα του οποίου θεωρούμε πάλι ότι είναι διατεταγμένα κατά φθίνουσα συχνότητα. Για να βρούμε τις απαντήσεις σε αυτό το ερώτημα με βάση ένα απλό inverted file θα έπρεπε απλά να πάρουμε την τομή των λιστών των αντικειμένων που περιέχονται στο query set. Στην περίπτωση του ordered inverted file όμως, ψάχνουμε να βρούμε τα όρια ανάμεσα στα οποία θα βρίσκονται οι πιθανές απαντήσεις, όπως κάναμε και με το equality query, έτσι ώστε χρησιμοποιώντας τα B-Δέντρα να προσπελάσουμε πάλι κατευθείαν τα τμήματα των λιστών που θα περιέχουν τις πιθανές απαντήσεις και όχι ολόκληρες τις λίστες.

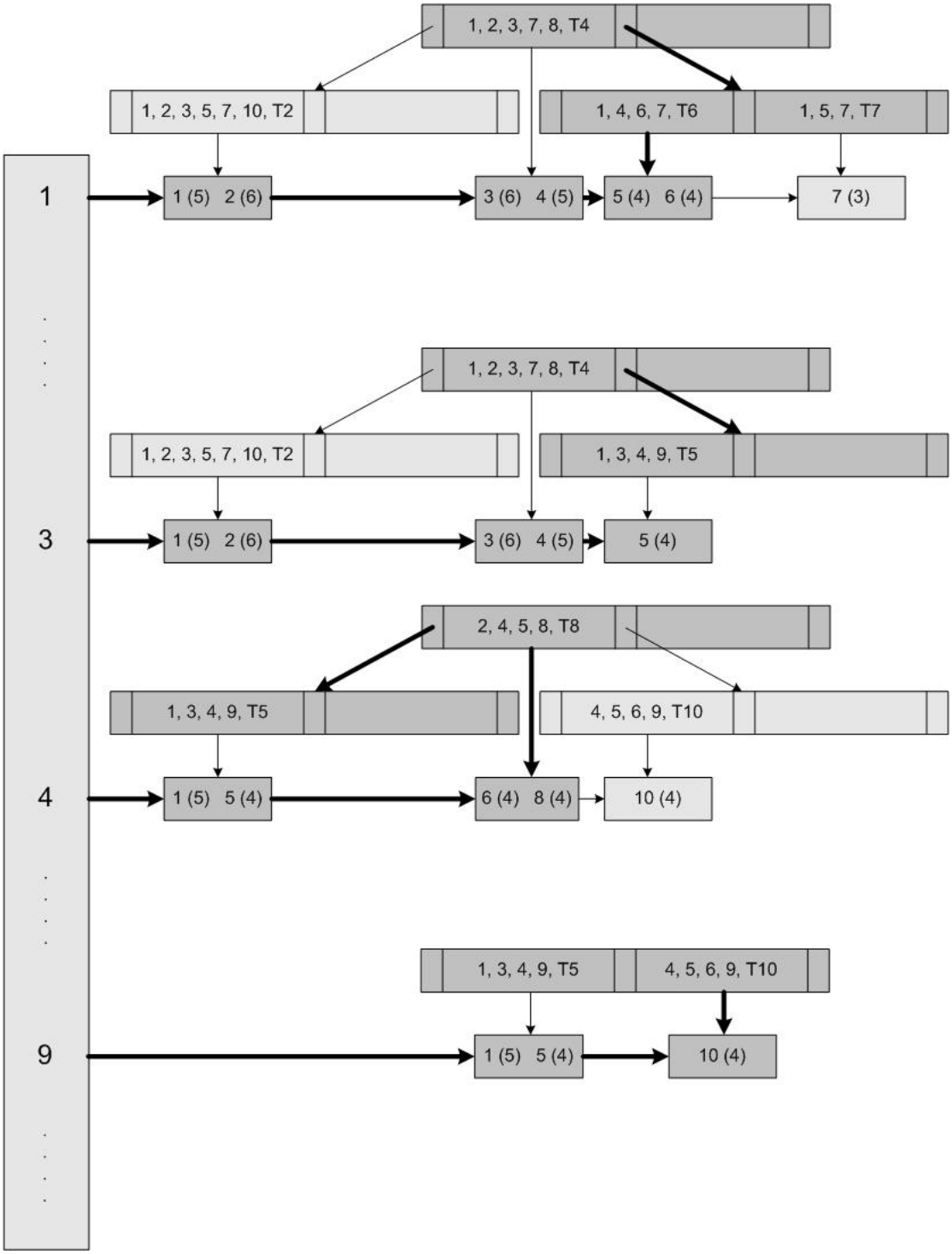
Ψάχνουμε αρχικά να προσδιορίσουμε για κάθε λίστα την αρχή της περιοχής με τις πιθανές απαντήσεις. Εφόσον στην περίπτωση του subset query μας ενδιαφέρει οι απαντήσεις μας να περιέχουν ολόκληρο το query set αλλά μπορούν επιπλέον να περιέχουν και άλλα αντικείμενα, η πρώτη απάντηση, έστω δ_1, που θα μπορούσε να εμφανιστεί σε μία λίστα θα περιέχει το query set αλλά και όλα τα αντικείμενα που είναι πιο συχνά από το τελευταίο αντικείμενο του query set. Δοσοληψίες που περιέχουν και αντικείμενα λιγότερο συχνά από το τελευταίο αντικείμενο του query set αποτελούν και αυτές απαντήσεις, αλλά με βάση την ταξινόμηση που κάναμε κατά την κατασκευή του ευρετηρίου θα βρίσκονται μετά από τη δ_1 μέσα στις λίστες. Έτσι, η περιοχή με τις πιθανές απαντήσεις ξεκινάει με τη δοσοληψία που περιέχει τα

$\{1,2,...,q_1-1,q_1,q_1+1,...,q_2-1,q_2,q_2+1,...,q_n-1,q_n\}$. Παρατηρούμε εδώ ότι η δοσοληψία αυτή, ειδικά αν τα q_1 έως q_n έχουν μεγάλη συχνότητα, θα βρίσκεται πολύ κοντά στην αρχή της λίστας και έτσι για να απλοποιήσουμε τη διαδικασία της αποτίμησης θεωρούμε ότι η περιοχή με τις πιθανές απαντήσεις ξεκινάει από την αρχή της κάθε λίστας.

Έτσι, μας ενδιαφέρει τώρα να βρούμε το τέλος της περιοχής των πιθανών απαντήσεων. Με βάση την ταξινόμηση που περιγράψαμε παραπάνω, μπορούμε να δούμε ότι από τη δοσοληψία με περιεχόμενα τα $\{q_1,q_2,q_3,...,q_n+1\}$ και μετά, τα αντικείμενα που μας ενδιαφέρουν δεν πρόκειται να ξαναεμφανιστούν όλα μαζί. Έτσι, το τέλος της περιοχής των πιθανών απαντήσεων θα είναι όπως και στην περίπτωση του equality query η δοσοληψία με περιεχόμενα τα $\{q_1,q_2,q_3,...,q_n+1\}$.

Θυμίζουμε επίσης ότι, όπως και στο equality query, για να βρούμε την πρώτη δοσοληψία με κάποια συγκεκριμένα περιεχόμενα σε ένα B-Δέντρο ψάχνουμε για τη δοσοληψία με τα περιεχόμενα αυτά και id=0. Αφού λοιπόν βρούμε τις σελίδες που αποτελούν τα άκρα της περιοχής πιθανών απαντήσεων στις λίστες που αντιστοιχούν στα αντικείμενα του query, βρίσκουμε τα αποτελέσματα παίρνοντας την τομή όλων αυτών των περιοχών. Για την τομή αυτή χρησιμοποιούμε τη συνάρτηση MergeJoin που παρουσιάστηκε στον κώδικα 3.1., χωρίς όμως αυτή τη φορά να απαιτούμε το μήκος των δοσοληψιών να είναι ίσο με το μήκος του query set.

Έστω για παράδειγμα η βάση του πίνακα 3.4. και το subset query με $qs = \{1,3,4,9\}$. Όπως μπορούμε πολύ εύκολα να δούμε από τον πίνακα 3.4. η απάντηση στο ερώτημα αυτό είναι οι δοσοληψίες με ids 1 και 5. Για να απαντήσουμε όμως στο ερώτημα με χρήση της μεθόδου που μόλις περιγράψαμε θα ψάξουμε στα B-Δέντρα των λιστών των αντικειμένων 1, 3, 4 και 9 για το κλειδί 1, 3, 4, 10, T0 το οποίο θα μας δώσει το τέλος των περιοχών πιθανών απαντήσεων και θα πάρουμε την τομή των τμημάτων που ξεκινούν από την αρχή των λιστών και φτάνουν μέχρι εκεί. Οι σελίδες που θα χρησιμοποιήσουμε τελικά και οι διαδρομές που ακολουθούμε για την εύρεσή τους φαίνονται με έντονο χρώμα στο σχήμα 3.3. Για άλλη μια φορά στο σχήμα δεν εμφανίζεται ολόκληρο το ordered inverted file της βάσης για λόγους απλότητας.



Σχήμα 3.3.: Αποτίμηση subset query με $qs = \{1,3,4,9\}$

Το ψάξιμο στο δέντρο για τα επιθυμητά κλειδιά και η προσπέλαση των σελίδων γίνεται με τρόπο ανάλογο με αυτόν που είδαμε προηγουμένως για το equality query, με μόνη διαφορά ότι για να βρούμε τις αρχές των περιοχών με τα πιθανά αποτελέσματα δε χρησιμοποιούμε πλέον τα Β-Δέντρα, όπως είπαμε και προηγουμένως, αλλά τους δείκτες που περιέχονται στο λεξικό του inverted file και δείχνουν στην πρώτη σελίδα της κάθε λίστας.

Ιδιαίτερο ενδιαφέρον στο παραπάνω παράδειγμα έχει ο τρόπος με τον οποίο βρίσκουμε το τέλος της περιοχής πιθανών αποτελεσμάτων της λίστας που αντιστοιχεί στο αντικείμενο 3. Σύμφωνα με τη μέθοδο που αναπτύξαμε παραπάνω, ψάχνουμε στο Β-Δέντρο της λίστας αυτής για το κλειδί 1, 3, 4, 10, T0. Ξεκινάμε από τη ρίζα, το μοναδικό κλειδί της οποίας είναι μικρότερο από αυτό που ψάχνουμε, και έτσι οδηγούμαστε στο δεξί παιδί της ρίζας, το οποίο όμως έχει μόνο ένα κλειδί, μικρότερο και αυτό από εκείνο που ψάχνουμε. Έτσι, συμπεραίνουμε ότι η τελευταία δοσοληψία που περιλαμβάνεται στη λίστα του αντικειμένου 3 προηγείται αυτής που μας ενδιαφέρει και έτσι πρέπει να χρησιμοποιήσουμε ολόκληρη τη λίστα κατά την τομή των λιστών. Στην περίπτωση που αυτό συνέβαινε ενώ ψάχναμε την αρχή της περιοχής πιθανών απαντήσεων και όχι το τέλος, θα συμπεραίναμε ότι εφόσον η πρώτη σελίδα της περιοχής αυτής θα έπρεπε να βρίσκεται μετά την τελευταία σελίδα της λίστας, δεν υπάρχει περιοχή πιθανών απαντήσεων σε αυτή τη λίστα. Ο τρόπος που αυτό υλοποιείται στην πράξη θα φανεί στο επόμενο κεφάλαιο κατά την ανάλυση του κώδικα του ευρετηρίου.

Στο σχήμα 3.3. μπορούμε να δούμε επίσης ότι αν πάρουμε την τομή των τμημάτων των λιστών που έχουμε σημειώσει με έντονο χρώμα, θα οδηγηθούμε πράγματι στο επιθυμητό αποτέλεσμα, δηλαδή στις δοσοληψίες με ids 1 και 5.

3.2.3 Superset queries

Η αποτίμηση των superset queries είναι πιο περίπλοκη από την αποτίμηση των άλλων δύο ειδών ερωτημάτων γιατί, όπως θα δούμε και στη συνέχεια, υπάρχουν περισσότερες από μία περιοχές πιθανών απαντήσεων σε καθεμία από τις λίστες που μας ενδιαφέρουν. Έστω λοιπόν το superset query με query set το $qs = \{q_1, q_2, q_3, ..., q_n\}$ όπου τα q_1 έως q_n είναι διατεταγμένα κατά φθίνουσα σειρά συχνότητας. Για να απαντήσουμε στο ερώτημα αυτό πρέπει να βρούμε τις δοσοληψίες που απαρτίζονται μόνο από αντικείμενα από το qs, χωρίς όμως να τα περιέχουν κατ' ανάγκην όλα. Προφανώς οι δοσοληψίες αυτές θα βρίσκονται στις λίστες των αντικειμένων του qs. Έτσι, ορίζουμε πολλές διαφορετικές περιοχές πιθανών απαντήσεων σε καθεμία από αυτές τις λίστες, ανάλογα με το πιο συχνό αντικείμενο που αυτές περιέχουν.

Προσπαθούμε αρχικά να προσδιορίσουμε τις περιοχές των λιστών στις οποίες βρίσκονται οι απαντήσεις στο ερώτημα που μας ενδιαφέρει οι οποίες περιέχουν σίγουρα το q_1. Στη λίστα του αντικειμένου q_1 η πρώτη δοσοληψία που μπορεί να αποτελεί απάντηση στο ερώτημά μας (και προφανώς περιέχει το q_1 αφού βρίσκεται στη λίστα αυτή) θα είναι η $\{q_1\}$, ενώ η τελευταία θα είναι η $\{q_1, q_n\}$ αφού όπως έχουμε ήδη πει οι δοσοληψίες είναι τοποθετημένες σε κάθε λίστα κατά φθίνουσα συχνότητα των αντικειμένων που περιέχουν. Στη λίστα του

αντικειμένου q_2 όλες οι απαντήσεις θα περιέχουν υποχρεωτικά και το q_2 και έτσι η πρώτη δοσοληψία που περιέχει το q_1 και αποτελεί απάντηση στο ερώτημα θα είναι η $\{q_1, q_2\}$ και η τελευταία η $\{q_1, q_2, q_n\}$. Συνεχίζουμε με τη λίστα του αντικειμένου q_3 όπου όλες οι απαντήσεις θα περιέχουν υποχρεωτικά το q_3. Εκεί η πρώτη δοσοληψία που αποτελεί απάντηση στο ερώτημα και περιέχει το q_1 θα είναι η $\{q_1, q_2, q_3\}$. Βλέπουμε εδώ ότι στη λίστα αυτή η πρώτη δοσοληψία που ψάχνουμε θα περιέχει και το q_2 αφού αυτό είναι πιο συχνό από το q_3 και έτσι η δοσοληψία $\{q_1, q_2, q_3\}$ προηγείται της $\{q_1, q_3\}$. Επίσης η τελευταία δοσοληψία που αποτελεί απάντηση στο ερώτημα και περιέχει το q_1 θα είναι η $\{q_1, q_3, q_n\}$. Εργαζόμενοι με τον ίδιο τρόπο και στις υπόλοιπες λίστες βλέπουμε τελικά ότι οι περιοχές πιθανών απαντήσεων που περιέχουν σίγουρα το q_1 είναι αυτές που φαίνονται στον πίνακα 3.5. Με τον όρο αρχή στον πίνακα αυτό αναφερόμαστε στην πρώτη δοσοληψία της περιοχής πιθανών απαντήσεων, η οποία είναι η πρώτη δοσοληψία της λίστας με περιεχόμενα αυτά που φαίνονται στον πίνακα, ενώ με τον όρο τέλος αναφερόμαστε στην τελευταία δοσοληψία της περιοχής πιθανών απαντήσεων, η οποία είναι η τελευταία δοσοληψία της λίστας με περιεχόμενα αυτά που φαίνονται στον πίνακα.

Λίστα	Περιοχή πιθανών απαντήσεων	
	Αρχή	Τέλος
q_1	$\{q_1\}$	$\{q_1, q_n\}$
q_2	$\{q_1, q_2\}$	$\{q_1, q_2, q_n\}$
q_3	$\{q_1, q_2, q_3\}$	$\{q_1, q_3, q_n\}$
q_4	$\{q_1, q_2, q_3, q_4\}$	$\{q_1, q_4, q_n\}$
q_5	$\{q_1, q_2, q_3, q_4, q_5\}$	$\{q_1, q_5, q_n\}$
q_n	$\{q_1, q_2, q_3, q_4, q_5, ..., q_n\}$	$\{q_1, q_n\}$

Πίνακας 3.5.: Περιοχές πιθανών απαντήσεων για το superset query με $qs = \{q_1, q_2, q_3, ..., q_n\}$ που περιέχουν το q_1

Από τη στιγμή που συγκεντρώσαμε όλες τις περιοχές πιθανών απαντήσεων που περιέχουν το q_1 ψάχνουμε για τις περιοχές πιθανών απαντήσεων που δεν το περιέχουν. Αυτό όμως είναι σαν να ψάχνουμε για τις περιοχές πιθανών απαντήσεων του superset query με $qs_2 = \{q_2, q_3, ..., q_n\}$, τις οποίες μπορούμε να προσδιορίσουμε βρίσκοντας πρώτα τις περιοχές πιθανών απαντήσεων που περιέχουν το q_2 και στη συνέχεια τις περιοχές πιθανών απαντήσεων του superset query με $qs_3 = \{q_3, ..., q_n\}$. Με τον τρόπο αυτό βρίσκουμε αναδρομικά όλες τις περιοχές πιθανών απαντήσεων του αρχικού ερωτήματος ($qs = \{q_1, q_2, q_3, ..., q_n\}$), οι οποίες φαίνονται στον πίνακα 3.6. που ακολουθεί. Για κάθε περιοχή δίνονται ως άκρα τα περιεχόμενα της πρώτης δοσοληψίας και τα περιεχόμενα της

τελευταίας. Όπως είπαμε και προηγουμένως, η πρώτη δοσοληψία της περιοχής είναι η πρώτη δοσοληψία με αυτά τα περιεχόμενα που εμφανίζεται στη λίστα, ενώ η τελευταία δοσοληψία της περιοχής είναι η τελευταία δοσοληψία με αυτά τα περιεχόμενα που εμφανίζεται στη λίστα.

Λίστα	Περιοχές πιθανών απαντήσεων	
	Περιέχουν το q_1	Περιέχουν το q_2
q_1	$\{q_1\}$ - $\{q_1, q_n\}$	
q_2	$\{q_1, q_2\}$ - $\{q_1, q_2, q_n\}$	$\{q_2\}$ - $\{q_2, q_n\}$
q_3	$\{q_1, q_2, q_3\}$ - $\{q_1, q_3, q_n\}$	$\{q_2, q_3\}$ - $\{q_2, q_3, q_n\}$
q_4	$\{q_1, q_2, q_3, q_4\}$ - $\{q_1, q_4, q_n\}$	$\{q_2, q_3, q_4\}$ - $\{q_2, q_4, q_n\}$
q_5	$\{q_1, q_2, q_3, q_4, q_5\}$ - $\{q_1, q_5, q_n\}$	$\{q_2, q_3, q_4, q_5\}$ - $\{q_2, q_5, q_n\}$
q_n	$\{q_1, q_2, q_3, q_4, q_5, ..., q_n\}$ - $\{q_1, q_n\}$	$\{q_2, q_3, q_4, q_5, ..., q_n\}$ - $\{q_2, q_n\}$

Λίστα	Περιοχές πιθανών απαντήσεων		
	Περιέχουν το q_3	Περιέχουν το q_4	Περιέχουν το q_n
q_1			
q_2			
q_3	$\{q_3\}$ - $\{q_3, q_n\}$		
q_4	$\{q_3, q_4\}$ - $\{q_3, q_4, q_n\}$	$\{q_4\}$ - $\{q_4, q_n\}$	
q_5	$\{q_3, q_4, q_5\}$ - $\{q_3, q_5, q_n\}$	$\{q_4, q_5\}$ - $\{q_4, q_5, q_n\}$	
q_n	$\{q_3, q_4, q_5, ..., q_n\}$ - $\{q_3, q_n\}$	$\{q_4, q_5, ..., q_n\}$ - $\{q_4, q_n\}$	$\{q_n\}$ - $\{q_n\}$

Πίνακας 3.6.: Περιοχές πιθανών απαντήσεων για το superset query με $qs = \{q_1, q_2, q_3, ..., q_n\}$

Όπως είδαμε και κατά την αποτίμηση των δύο άλλων ειδών ερωτημάτων, ψάχνοντας στο B-Δέντρο μιας λίστας για το κλειδί με κάποια συγκεκριμένα περιεχόμενα και id = 0 βρίσκουμε τη σελίδα της λίστας στην οποία βρίσκεται η πρώτη δοσοληψία με τα περιεχόμενα αυτά, ή η αμέσως επόμενη αν αυτή δεν υπάρχει. Με τον τρόπο αυτό μπορούμε πολύ εύκολα να προσδιορίσουμε την αρχή μιας περιοχής πιθανών απαντήσεων, αφού μας ενδιαφέρει η πρώτη εμφάνιση της δοσοληψίας με τα περιεχόμενα που φαίνονται στον πίνακα 3.6., αλλά όχι το τέλος της, αφού εκεί μας ενδιαφέρει η τελευταία εμφάνιση της δοσοληψίας με τα περιεχόμενα που βλέπουμε στον πίνακα 3.6. Για το λόγο αυτό αντικαθιστούμε την τελευταία δοσοληψία της περιοχής πιθανών απαντήσεων με κάποια δοσοληψία που έπεται αυτής και η πρώτη εμφάνιση της οποίας μπορεί να χρησιμοποιηθεί ως όριο για το τέλος της περιοχής πιθανών απαντήσεων. Ξαναβλέπουμε λοιπόν στον πίνακα 3.7. τις περιοχές πιθανών απαντήσεων, με τη δοσοληψία που ορίζει το τέλος της περιοχής να είναι αυτή τη φορά η πρώτη δοσοληψία με περιεχόμενα αυτά που εμφανίζονται στον πίνακα.

Λίστα	Περιοχές πιθανών απαντήσεων	
	Περιέχουν το q_1	Περιέχουν το q_2
q_1	$\{q_1\} - \{q_1, q_n + 1\}$	
q_2	$\{q_1, q_2\} - \{q_1, q_2, q_n + 1\}$	$\{q_2\} - \{q_2, q_n + 1\}$
q_3	$\{q_1, q_2, q_3\} - \{q_1, q_3, q_n + 1\}$	$\{q_2, q_3\} - \{q_2, q_3, q_n + 1\}$
q_4	$\{q_1, q_2, q_3, q_4\} - \{q_1, q_4, q_n + 1\}$	$\{q_2, q_3, q_4\} - \{q_2, q_4, q_n + 1\}$
q_5	$\{q_1, q_2, q_3, q_4, q_5\} - \{q_1, q_5, q_n + 1\}$	$\{q_2, q_3, q_4, q_5\} - \{q_2, q_5, q_n + 1\}$
q_n	$\{q_1, q_2, q_3, q_4, q_5, ..., q_n\} - \{q_1, q_n + 1\}$	$\{q_2, q_3, q_4, q_5, ..., q_n\} - \{q_2, q_n + 1\}$

Λίστα	Περιοχές πιθανών απαντήσεων		
	Περιέχουν το q_3	Περιέχουν το q_4	Περιέχουν το q_n
q_1			
q_2			
q_3	$\{q_3\} - \{q_3, q_n + 1\}$		
q_4	$\{q_3, q_4\} - \{q_3, q_4, q_n + 1\}$	$\{q_4\} - \{q_4, q_n + 1\}$	
q_5	$\{q_3, q_4, q_5\} - \{q_3, q_5, q_n + 1\}$	$\{q_4, q_5\} - \{q_4, q_5, q_n + 1\}$	
q_n	$\{q_3, q_4, q_5, ..., q_n\} - \{q_3, q_n + 1\}$	$\{q_4, q_5, ..., q_n\} - \{q_4, q_n + 1\}$	$\{q_n\} - \{q_n + 1\}$

Πίνακας 3.7.: Περιοχές πιθανών απαντήσεων για το superset query με $qs = \{q_1, q_2, q_3, ..., q_n\}$

Παρατηρώντας τον πίνακα 3.7. μπορούμε να δούμε ότι στη γενική περίπτωση, αν έχουμε το superset query με query set το $qs = \{q_1, q_2, q_3, ..., q_n\}$, για να βρούμε την περιοχή πιθανών απαντήσεων στη λίστα i η οποία περιέχει σίγουρα το πιο συχνό αντικείμενο του ερωτήματος (q_1), προσδιορίζουμε την αρχή της ψάχνοντας στο B-Δέντρο της λίστας για το κλειδί με περιεχόμενα τα $\{q_1, q_2, ..., q_i\}$ και id = 0, και το τέλος της ψάχνοντας για το κλειδί με περιεχόμενα τα $\{q_1, q_i, q_n + 1\}$ και id = 0. Η εύρεση της περιοχής πιθανών απαντήσεων που περιέχει ένα λιγότερο συχνό αντικείμενο (έστω q_i), μπορεί να αναχθεί όπως είδαμε προηγουμένως στην εύρεση της περιοχής πιθανών απαντήσεων που περιέχει το πιο συχνό αντικείμενο του query set $qs_i = \{q_i, q_{i+1}, ..., q_n\}$.

Αφού όπως είδαμε παραπάνω οι περιοχές πιθανών απαντήσεων του ερωτήματος μπορούν να προσδιοριστούν αναδρομικά, το ίδιο θα συμβαίνει και με τις απαντήσεις του ερωτήματος. Έτσι, για να απαντήσουμε στο ερώτημα με $qs = \{q_1, q_2, q_3, ..., q_n\}$ αρκεί να βρούμε τις απαντήσεις που περιέχουν το q_1 και στη συνέχεια να απαντήσουμε στο ερώτημα με $qs_2 = \{q_2, q_3, ..., q_n\}$. Αυτό πάλι θα γίνει βρίσκοντας τις απαντήσεις που περιέχουν το q_2 και απαντώντας στη συνέχεια στο ερώτημα με $qs_3 = \{q_3, ..., q_n\}$, κ.ο.κ. Με αυτόν τον τρόπο, παίρνοντας την ένωση των απαντήσεων που βρίσκουμε στα διάφορα στάδια της αναδρομής, έχουμε τελικά όλες τις απαντήσεις του αρχικού ερωτήματος.

Εξετάζουμε τώρα τον τρόπο με τον οποίο βρίσκουμε τις απαντήσεις του ερωτήματος με $qs = \{q_1, q_2, q_3, ..., q_n\}$ που περιέχουν το q_1. Αυτό που πρέπει να κάνουμε είναι να επισκεφτούμε όλες τις λίστες των αντικειμένων που περιέχονται στο query set και να βρούμε την περιοχή πιθανών απαντήσεων της κάθε λίστας που περιέχει το q_1, όπως περιγράφηκε στα προηγούμενα. Στη συνέχεια διατρέχουμε την περιοχή πιθανών απαντήσεων της λίστας του q_1, η οποία περιέχει σίγουρα όλες τις απαντήσεις που περιέχουν το q_1 αφού περιέχει όλες τις δοσοληψίες που περιλαμβάνουν το q_1, και για κάθε δοσοληψία που συναντάμε απαιτούμε να βρίσκεται σε τόσες λίστες (συμπεριλαμβανομένης και αυτής του q_1) όσο είναι το μήκος της. Η απαίτηση αυτή αποκλείει την πιθανότητα να περιέχονται στη δοσοληψία αυτή και άλλα αντικείμενα εκτός από αυτά του query set. Έτσι, για κάθε δοσοληψία με μήκος μεγαλύτερο από 1 που συναντάμε στην περιοχή πιθανών απαντήσεων της λίστας του q_1 διατρέχουμε μία-μία τις υπόλοιπες λίστες μέχρι να βρούμε τόσες εμφανίσεις της δοσοληψίας όσες το μήκος της ή να αποφανθούμε ότι αυτό δεν πρόκειται να γίνει. Ένα τέτοιο συμπέρασμα μπορεί να βγει αν οι λίστες που δεν έχουμε εξετάσει ακόμη είναι λιγότερες από τις εμφανίσεις της δοσοληψίας που θέλουμε ακόμη να βρούμε.

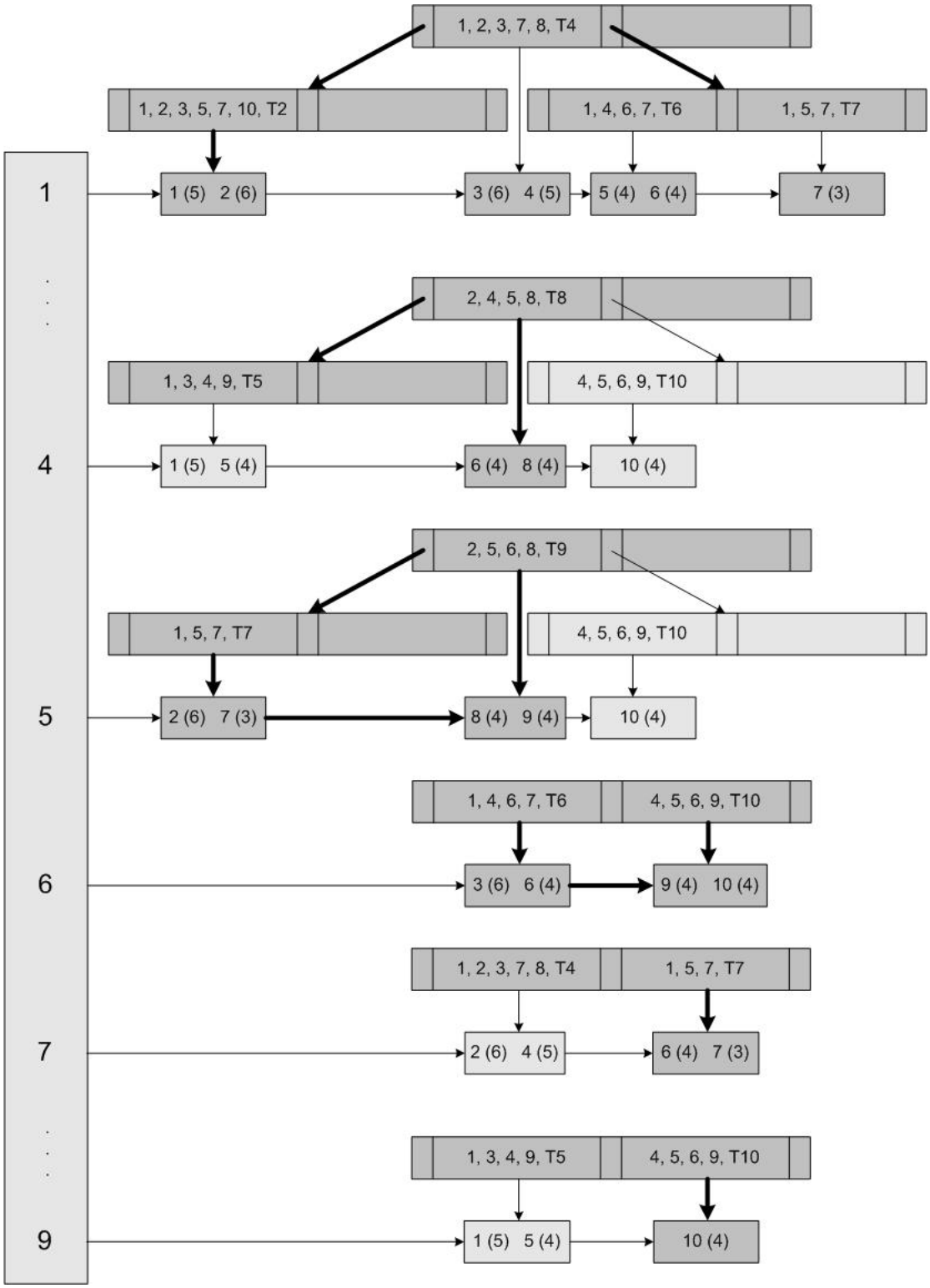
Η συνάρτηση που χρησιμοποιούμε για το σκοπό αυτό εμφανίζεται σε ψευδοκώδικα στον κώδικα 3.2., όπου με RoI συμβολίζουμε για άλλη μια φορά τις περιοχές πιθανών απαντήσεων των λιστών. Έχουμε ονομάσει τη συνάρτηση αυτή MergeJoin, σε αντιστοιχία με την αντίστοιχη συνάρτηση που χρησιμοποιούμε για την αποτίμηση των equality και subset queries.

```
for each i in RoI of qn.list {
    tolMisses = n-qn.length;
    for (j=2; j≤n; j++) {
          if tolMisses<0 break;
          tolMisses--;
          for each item c<i in RoI of qj.list {
                if (c=i) tolMisses++;
          }
    }
    if tolMisses≤0 output i;
}
```

Κώδικας 3.2.: Η λειτουργία της συνάρτησης MergeJoin για τον προσδιορισμό των απαντήσεων ενός superset query

Η διαδικασία αυτή μπορεί να γίνει καλύτερα κατανοητή με ένα παράδειγμα. Έστω η βάση που φαίνεται στον πίνακα 3.5. στην οποία θέλουμε να κάνουμε ένα superset query με $qs = \{1,4,5,6,7,9\}$. Βρίσκουμε αρχικά τις περιοχές πιθανών απαντήσεων των λιστών των

αντικειμένων 1, 4, 5, 6, 7 και 9 που περιέχουν το αντικείμενο 1, οι οποίες φαίνονται στο σχήμα 3.4. μαζί με τη διαδρομή που ακολουθούμε για να τις βρούμε. Για άλλη μια φορά δε φαίνεται στο σχήμα ολόκληρο το ordered inverted file, παρά μόνο τα τμήματα αυτού που μας ενδιαφέρουν κατά την αποτίμηση του ερωτήματος, για λόγους απλότητας.



Σχήμα 3.4.: Εύρεση περιοχών πιθανών απαντήσεων για το superset query με $qs = \{1,4,5,6,7,9\}$ **που περιέχουν το αντικείμενο 1**

Για τον προσδιορισμό των περιοχών πιθανών απαντήσεων που φαίνονται στο σχήμα 3.4. εργαστήκαμε με τρόπο αντίστοιχο με αυτόν που περιγράψαμε κατά την αποτίμηση των equality queries και subset queries. Για να βρούμε τώρα ποιες από τις παραπάνω αποτελούν πράγματι απαντήσεις του ερωτήματος θα πρέπει να διατρέξουμε την περιοχή πιθανών απαντήσεων της λίστας του αντικειμένου 1 και να δούμε στις περιοχές πιθανών απαντήσεων πόσων λιστών εμφανίζεται η κάθε δοσοληψία. Τα αποτελέσματα αυτής της έρευνας φαίνονται πιο παραστατικά στον πίνακα 3.8. που ακολουθεί.

Δοσοληψία	Μήκος	Εμφανίσεις	Είναι απάντηση
1	5	1	ΟΧΙ
2	6	2	ΟΧΙ
3	6	2	ΟΧΙ
4	5	1	ΟΧΙ
5	4	1	ΟΧΙ
6	4	4	ΝΑΙ
7	3	3	ΝΑΙ

Πίνακας 3.8.:Εύρεση απαντήσεων στο superset query με $qs = \{1,4,5,6,7,9\}$ που περιέχουν το αντικείμενο 1

Εδώ μπορούμε να παρατηρήσουμε ότι ψάχνοντας για τις εμφανίσεις της δοσοληψίας 1 η οποίο έχει μήκος ίσο με 5, βλέπουμε κατ' αρχήν ότι αυτή δεν εμφανίζεται στη λίστα του αντικειμένου 4 και στη συνέχεια ότι δεν εμφανίζεται ούτε στη λίστα του αντικειμένου 5. Από εκεί και πέρα δε χρειάζεται να συνεχίσουμε το ψάξιμο και στις επόμενες λίστες αφού, ακόμη και αν το βρούμε σε όλες τις λίστες που ακολουθούν, το πλήθος των εμφανίσεών του θ παραμείνει μικρότερο από 5. Με τον ίδιο τρόπο εργαζόμαστε και για τα υπόλοιπα αντικείμενα.

Βλέπουμε λοιπόν ότι οι απαντήσεις στο ερώτημά μας που περιέχουν το αντικείμενο 1 είναι οι δοσοληψίες 6 και 7. Με τον ίδιο τρόπο βρίσκουμε ότι η μοναδική απάντηση στο superset query με $qs = \{4,5,6,7,9\}$ που περιέχουν το αντικείμενο 4 είναι η δοσοληψία 10, ενώ απαντήσεις στα superset queries με $qs = \{5,6,7,9\}$, $qs = \{6,7,9\}$, $qs = \{7,9\}$ και $qs = \{9\}$ που να περιέχουν το πιο συχνό αντικείμενο του query set δεν υπάρχουν. Έτσι, οι απαντήσεις στο αρχικό ερώτημα θα είναι η ένωση των απαντήσεων που βρήκαμε εδώ, δηλαδή οι δοσοληψίες 6, 7 και 10. Η ορθότητα του αποτελέσματός μας μπορεί να ελεγχθεί και από τον πίνακα 3.4. όπου μπορεί κάποιος εποπτικά να δει ότι οι δοσοληψίες αυτές είναι πράγματι οι μοναδικές απαντήσεις του ερωτήματος που θέσαμε προηγουμένως.

3.2.4 Κόστος αποτίμησης ερωτημάτων

Στα πειράματα για την εύρεση της απόδοσης του ordered inverted file σε σχέση με το απλό inverted file που θα παρουσιαστούν στο Κεφάλαιο 5, υπολογίζουμε την απόδοση των δύο ευρετηρίων ως το πλήθος σελίδων που προσπελαύνονται κατά την αποτίμηση των

ερωτημάτων. Όσο μικρότερο το πλήθος αυτό τόσο πιο αποδοτικό το ευρετήριο αφού η καθυστέρηση κατά τη μεταφορά σελίδων από το δίσκο στη μνήμη θα είναι μικρότερη. Είναι λοιπόν σκόπιμο να αναφέρουμε εδώ ότι οι σελίδες που προσπελαύνονται κατά την αποτίμηση ενός ερωτήματος είναι τριών ειδών, όπως μπορούμε να δούμε και από τα παραπάνω.

- Σελίδες που ανήκουν σε Β-Δέντρα
- Σελίδες που ανήκουν σε ανεστραμμένες λίστες
- Σελίδες που ανήκουν στον ενδιάμεσο πίνακα και χρησιμοποιούνται μετά την εύρεση των ids των απαντήσεων για να βρεθεί η θέση των απαντήσεων αυτών στο δίσκο.

Θυμίζουμε εδώ ότι ο ενδιάμεσος πίνακας περιέχει τις σελίδες του δίσκου στις οποίες είναι αποθηκευμένες οι δοσοληψίες, ταξινομημένες κατά αύξον id των δοσοληψιών, χωρίς όμως να κρατάει επιπλέον και τα ids.

Έτσι, για να υπολογίσουμε τις σελίδες που προσπελαύνονται κατά την αποτίμηση κάποιου ερωτήματος μετράμε τις σελίδες των Β-Δέντρων που προσπελαύνουμε κατά τον προσδιορισμό των ορίων των περιοχών πιθανών απαντήσεων και τις σελίδες των ανεστραμμένων λιστών που προσπελαύνουμε κατά την εκτέλεση της κατάλληλης πράξης ανάμεσα στις λίστες για την αποτίμηση του ερωτήματος. Σε αυτές προσθέτουμε το πλήθος των σελίδων του ενδιάμεσου πίνακα στις οποίες βρίσκονται οι διευθύνσεις των απαντήσεων που βρήκαμε, οι οποίες μπορούν να υπολογιστούν με μία απλή διαίρεση των ids των δοσοληψιών με το πλήθος των διευθύνσεων που χωράνε σε μία σελίδα, αφού η διεύθυνση μίας δοσοληψίας βρίσκεται σε διαφορετική σελίδα του πίνακα από αυτή της προηγούμενής της αν και μόνον αν τα υπόλοιπα της παραπάνω διαίρεσης για τις δύο δοσοληψίες είναι διαφορετικά.

3.3 Συμπίεση – Ανανέωση

Όπως συμβαίνει και με το απλό inverted file, έτσι και το ordered inverted file μπορεί να συμπιεστεί αν αντικαταστήσουμε τα ids των δοσοληψιών στις ανεστραμμένες λίστες με τη διαφορά του id της κάθε δοσοληψίας από την προηγούμενη που εμφανίζεται στη λίστα αυτή, αφού, όπως αναφέρθηκε κατά την περιγραφή της κατασκευής του ευρετηρίου, οι δοσοληψίες σε κάθε λίστα είναι διατεταγμένες κατά αύξον id. Έτσι, μπορούμε στη συνέχεια να χρησιμοποιήσουμε κάποια από τις μεθόδους κωδικοποίησης που παρουσιάστηκαν στο Κεφάλαιο 2 για να μειώσουμε το χώρο που καταλαμβάνουν οι λίστες.

Θεωρώντας λοιπόν ότι σε κάθε ανεστραμμένη λίστα αποθηκεύονται οι διαφορές μεταξύ των ids, η ανανέωση του ευρετηρίου μπορεί να γίνει πολύ εύκολα ως εξής. Έστω ότι θέλουμε να προσθέσουμε στο ευρετήριο μία νέα δοσοληψία. Θεωρώντας ότι οι συχνότητες των αντικειμένων της βάσης δε θα μεταβληθούν πολύ και άρα δε χρειάζεται να αλλάξουμε τη

διάταξη των αντικειμένων με βάση τη συχνότητα που κάναμε κατά την κατασκευή του ευρετηρίου, το μόνο που πρέπει να κάνουμε πριν εισάγουμε τη νέα δοσοληψία είναι να διατάξουμε τα αντικείμενα στο εσωτερικό της με βάση τη συχνότητά τους. Η παραδοχή που αναφέραμε εδώ μπορεί να γίνει ακόμη και αν εισάγουμε στη βάση μεγάλο πλήθος δοσοληψιών, αφού τα αντικείμενα που εμφανίζονται συχνότερα στη βάση στατιστικά θα εμφανίζονται συχνότερα και στις νέες δοσοληψίες, με αποτέλεσμα οι σχετικές συχνότητες των αντικειμένων μέσα στη βάση να μη μεταβάλλονται πολύ.

Αφού διατάξουμε τα αντικείμενα της δοσοληψίας, πρέπει να βρούμε τη θέση που θα πρέπει να έχει η νέα αυτή δοσοληψία σε σχέση με τις ήδη υπάρχουσες, ούτως ώστε οι δοσοληψίες να συνεχίσουν να είναι διατεταγμένες κατά φθίνουσα συχνότητα των αντικειμένων που περιέχουν. Αυτό σημαίνει ότι θα πρέπει να δώσουμε στη νέα δοσοληψία ένα νέο id, έστω i, και να μετονομάσουμε τις δοσοληψίες από την υπάρχουσα δοσοληψία με id = i και μετά αυξάνοντας τα ids τους κατά ένα, έτσι ώστε το id της κάθε δοσοληψίας να παραμείνει μοναδικό.

Στη συνέχεια, η νέα δοσοληψία πρέπει να εισαχθεί στις λίστες των αντικειμένων που περιέχει, στη θέση που ορίζει το νέο id που της δώσαμε, δηλαδή μετά τη δοσοληψία i-1 και πριν τη δοσοληψία i+1 (πρώην δοσοληψία i). Για να γίνει αυτό θα έπρεπε κανονικά να ολισθήσουμε όλες τις δοσοληψίες από την i+1 και μετά κατά μία θέση, αδειάζοντας έτσι χώρο για την αποθήκευση της νέας δοσοληψίας στο σωστό σημείο. Καθώς όμως η πρακτική αυτή μας υποχρεώνει κάθε φορά που εισάγουμε μία νέα δοσοληψία να διατρέχουμε μεγάλα τμήματα των λιστών στις οποίες αυτή θα εισαχθεί για να ολισθήσουμε τις υπόλοιπες δοσοληψίες, ειδικά αν η νέα δοσοληψία πρέπει να εισαχθεί κοντά στην αρχή της λίστας, προτιμούμε την προσέγγιση που παρουσιάζεται παρακάτω.

Αντί να κρατάμε όλες τις σελίδες μίας λίστας εκτός από την τελευταία τελείως γεμάτες, επιτρέπουμε σε όλες τις σελίδες να περιέχουν το λιγότερο $\lceil n/2 \rceil$ δοσοληψίες, όπου n ο μέγιστος αριθμός δοσοληψιών που χωράνε σε μία σελίδα, όπως συμβαίνει και με τα κλειδιά ενός B-Δέντρου. Έτσι, για να εισάγουμε μία νέα δοσοληψία σε μία σελίδα, αν η σελίδα αυτή περιέχει λιγότερες από n δοσοληψίες τότε την εισάγουμε στη σωστή θέση ολισθαίνοντας αυτές που την ακολουθούν κατά μία θέση. Αφού η σελίδα έχει κενό χώρο, η ολίσθηση αυτή θα σταματήσει στο τέλος της σελίδας και όχι στο τέλος της λίστας όπως συνέβαινε με την προηγούμενη προσέγγιση. Αν πάλι η σελίδα είναι γεμάτη, τότε τη σπάμε σε δύο σελίδες μοιράζοντας τις δοσοληψίες που η σελίδα περιείχε μαζί με την καινούργια δοσοληψία ισότιμα στις δύο σελίδες έτσι ώστε να είναι και οι δύο τουλάχιστον μισογεμάτες. Σε κάθε περίπτωση, η εισαγωγή της νέας δοσοληψίας απαιτεί την προσπέλαση το πολύ δύο σελίδων από κάθε λίστα.

Μένει τώρα να δούμε πώς μπορούμε να μετονομάσουμε τις δοσοληψίες που έπονται της καινούργιας δοσοληψίας. Στην περίπτωση που στις ανεστραμμένες λίστες αποθηκεύουμε τις διαφορές ανάμεσα στα ids των δοσοληψιών όπως είπαμε παραπάνω, αυτό είναι πολύ απλό αφού για να μετονομάσουμε τις δοσοληψίες από την i και μετά αρκεί να επισκεφτούμε όλες τις λίστες και να αλλάξουμε τη διαφορά της δοσοληψίας i (ή της αμέσως μεγαλύτερης αν η i δεν εμφανίζεται στη συγκεκριμένη λίστα) από την προηγούμενή της καθώς το γεγονός ότι τα ids όλων αυτών των δοσοληψιών θα αυξηθούν κατά τον ίδιο αριθμό σημαίνει ότι οι διαφορές ανάμεσα στις υπόλοιπες δοσοληψίες δε θα αλλάξουν.

Με τη μέθοδο αυτή φαίνεται εκ πρώτης όψεως ότι για να βρούμε το πραγματικό id μίας τυχαίας δοσοληψίας πρέπει να διατρέξουμε ολόκληρη τη λίστα στην οποία βρίσκεται προσθέτοντας τις διαφορές από την αρχή μέχρι να φτάσουμε στη δοσοληψία που μας ενδιαφέρει, χάνοντας έτσι το πλεονέκτημα που το ordered inverted file προσφέρει έναντι το απλού inverted file. Αυτό όμως στην πραγματικότητα δε συμβαίνει αφού η μόνη περίπτωση να προσπελάσουμε μία σελίδα κάποιας λίστας χωρίς να έχουμε προηγουμένως διατρέξει όλη τη λίστα από την αρχή είναι να προσπελάσουμε τη συγκεκριμένη σελίδα μέσω του Β-Δέντρου της λίστας. Έτσι, το γεγονός ότι σε κάθε κλειδί του Β-Δέντρου αναγράφουμε το id της τελευταίας δοσοληψίας της σελίδας στην οποία μας οδηγεί μας επιτρέπει να βρούμε το id μιας οποιασδήποτε δοσοληψίας μέσα στη σελίδα αυτή αφαιρώντας από το id της τελευταίας δοσοληψίας που παίρνουμε από το δέντρο τις διαφορές των ids που εμφανίζονται στη λίστα μέχρι να φτάσουμε στη δοσοληψία που μας ενδιαφέρει.

Αντίστοιχα βέβαια, κατά τη μετονομασία ενός συνόλου δοσοληψιών θα πρέπει να φροντίσουμε να μεταβάλλουμε και τα ids αυτών που εμφανίζονται στα Β-Δέντρα. Αν και αυτό απαιτεί την προσπέλαση σαφώς λιγότερων σελίδων απ' όσες θα απαιτούσε η μετονομασία στη λίστα αν κρατούσαμε τα πραγματικά ids των δοσοληψιών αντί για τις διαφορές ανάμεσά τους, μπορούμε να κάνουμε τη μετονομασία αυτή ακόμη πιο αποδοτική αποθηκεύοντας και στα Β-Δέντρα τις διαφορές ανάμεσα στις δοσοληψίες αντί για τα ids τους. Έτσι, όλα τα κλειδιά μίας σελίδας εκτός από το πρώτο, θα περιέχουν τη διαφορά του id της δοσοληψίας στην οποία αντιστοιχούν από αυτή του αμέσως προηγούμενου κλειδιού. Το πρώτο κλειδί κάθε σελίδας, θα περιέχει τη διαφορά του id της δοσοληψίας στην οποία αντιστοιχεί από αυτή του κλειδιού i της σελίδας – πατέρα, όπου i+1 ο δείκτης που οδηγεί από τη σελίδα – πατέρα στην τρέχουσα σελίδα. Στην περίπτωση που στη σελίδα αυτή οδηγούμαστε από τον πρώτο δείκτη της σελίδας – πατέρα (i = –1) και έτσι δεν υπάρχει το κλειδί i, θα περιέχει τη διαφορά από το id της δοσοληψίας του κλειδιού 0. Έτσι, μπορούμε διατρέχοντας το δέντρο για να βρούμε το επιθυμητό κλειδί να βρούμε και το πραγματικό id της δοσοληψίας στην οποία αντιστοιχεί προσθέτοντας (ή αφαιρώντας αν ακολουθήσουμε το δείκτη 0 κάποιας σελίδας) τις διαφορές που συναντάμε στη διαδρομή που ακολουθούμε. Με

τον τρόπο αυτό η αύξηση των ids των δοσοληψιών για τις οποίες ισχύει $id \geq i$ κατά ένα ανάγεται πάλι στην αύξηση κατά ένα μίας και μόνο διαφοράς όπως συμβαίνει και στην ανεστραμμένη λίστα.

Σημειώνουμε εδώ ότι σε κάθε περίπτωση θα πρέπει να ανανεώσουμε και τον ενδιάμεσο πίνακα αφού όπως είπαμε προηγουμένως οι διευθύνσεις των δοσοληψιών αποθηκεύονται εκεί με βάση τα ids των δοσοληψιών, αλλά και σε αυτή την περίπτωση θα μπορούσαμε πιθανώς να ακολουθήσουμε κάποια προσέγγιση ανάλογη με αυτή που χρησιμοποιούμε στις ανεστραμμένες λίστες για να αποφύγουμε την ανάγκη ολίσθησης των διευθύνσεων όλων των δοσοληψιών που θέλουμε να μετονομάσουμε. Σε κάθε περίπτωση θα πρέπει να ελεγχθεί η πιθανή μείωση της απόδοσης του ευρετηρίου κατά την αποτίμηση ερωτημάτων από μία τέτοια προσέγγιση.

Η διαγραφή μίας δοσοληψίας από τη λίστα μπορεί να υλοποιηθεί ως η ακριβώς αντίστροφη διαδικασία από την εισαγωγή που περιγράψαμε παραπάνω. Οποιαδήποτε άλλη μεταβολή στη βάση μπορεί να θεωρηθεί συνδυασμός εισαγωγών και διαγραφών. Τέλος, αναφέρουμε ότι οι μέθοδοι για συμπίεση και ανανέωση του ευρετηρίου που αναπτύχθηκαν εδώ δεν έχουν υλοποιηθεί στον κώδικα του ευρετηρίου που παρουσιάζεται στο επόμενο κεφάλαιο, αλλά μπορούν να συμπεριληφθούν σε αυτόν με αρκετά μεγάλη ευκολία στο μέλλον.

4

Υλοποίηση

Μετά τη θεωρητική μελέτη του ordered inverted file και των λειτουργιών του, αναπτύσσεται κώδικας που υλοποιεί την κατασκευή του ordered inverted file καθώς και την αποτίμηση ερωτημάτων με χρήση αυτού. Η συμπίεση και ανανέωση του ευρετηρίου, όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, δεν υλοποιήθηκε σε αυτή τη φάση. Στο κεφάλαιο αυτό περιγράφουμε πώς έγινε η υλοποίηση παραθέτοντας χαρακτηριστικά αποσπάσματα του κώδικα

4.1 Πλατφόρμες και προγραμματιστικά εργαλεία

Το ordered inverted file υλοποιήθηκε στα πλαίσια της διπλωματικής αυτής σε γλώσσα προγραμματισμού C++ και σε περιβάλλον SuSe Linux με μεταγλωττιστή τον gcc του gnu project. Δοκιμάστηκε επίσης και σε περιβάλλον Windows XP Professional SP2 με το μεταγλωττιστή για C++ της Borland, έκδοση 5.5.1. Κάποια βοηθητικά προς την υλοποίησή μας προγράμματα για τη διάταξη των δεδομένων που θα αναφερθούν παρακάτω αναπτύχθηκαν και εκτελέστηκαν σε περιβάλλον Windows XP Professional SP2 με το Java SDK 1.5.0.

Η υλοποίησή μας αποθηκεύει τα B-Δέντρα στο σκληρό δίσκο ενώ για το inverted file τμήμα του ευρετηρίου προσφέρει την επιλογή να αποθηκευτεί είτε στο δίσκο είτε στην κύρια μνήμη. Αυτό συμβαίνει επειδή στην περίπτωση που το inverted file αποθηκευτεί και αυτό στο δίσκο αυξάνει δραματικά ο χρόνος που απαιτείται για την αποτίμηση των ερωτημάτων, με

αποτέλεσμα η υλοποίηση αυτή να μην είναι εύχρηστη για τη διενέργεια μεγάλου αριθμού πειραμάτων. Έτσι, η μοναδική απαίτηση για την εκτέλεση του τελικού προγράμματος είναι να υπάρχει επαρκής χώρος στο δίσκο για την αποθήκευση των αρχείων και το σύστημα να μας επιτρέπει την κατασκευή των αρχείων αυτών.

4.2 Λεπτομέρειες υλοποίησης

Όπως είπαμε και προηγουμένως η υλοποίηση της δομής του ευρετηρίου έγινε σε γλώσσα C++. Η υλοποίησή μας διαβάζει τις δοσοληψίες που αποτελούν τη βάση από ένα αρχείο (input file) και κατασκευάζει ένα ordered inverted file πάνω στη βάση αυτή. Τα Β-Δέντρα του ordered inverted file αποθηκεύονται στο δίσκο με τη μορφή δυαδικών αρχείων, ένα για κάθε Β-Δέντρο, ενώ οι λίστες του inverted file μπορούν να κρατηθούν στη μνήμη ή να αποθηκευτούν και αυτές στο δίσκο, πάλι σε ένα δυαδικό αρχείο η καθεμία. Το πρόγραμμά μας μπορεί στη συνέχεια να διαβάσει από ένα αρχείο (query file) τα ερωτήματα που θέλουμε να θέσουμε στη βάση και αφού τα αποτιμήσει να μας δώσει ένα αρχείο με τα αποτελέσματα των ερωτημάτων (results file) και ένα δεύτερο αρχείο με πληροφορίες για την αποτίμηση των ερωτημάτων (stats file), όπως για παράδειγμα τις σελίδες που προσπελάστηκαν για κάθε ερώτημα, το πλήθος των απαντήσεων κ.λ.π.

Η συγκεκριμένη υλοποίηση θεωρεί ότι τα δεδομένα δίνονται στο input file στην επιθυμητή μορφή, δηλαδή με τα αντικείμενα και τις δοσοληψίες διατεταγμένα κατά φθίνουσα σειρά συχνότητας, όπου 1 το περισσότερο συχνό αντικείμενο. Για να γίνει αυτό με τα τυχαία δεδομένα που χρησιμοποιούμε για τα πειράματά μας, δημιουργήθηκαν δύο προγράμματα σε Java τα οποία μεταβάλλουν τα αρχικά δεδομένα φέρνοντάς τα στην επιθυμητή μορφή, τα οποία δε θεωρείται σκόπιμο να εξεταστούν περαιτέρω εδώ αφού δεν παρουσιάζουν ιδιαίτερο ενδιαφέρον. Αυτό δε συμβαίνει και με τα ερωτήματα, αφού αν και θεωρούμε ότι περιέχουν τις ίδιες ονομασίες αντικειμένων που χρησιμοποιούνται και από το ευρετήριο (1 το πιο συχνό αντικείμενο, 2 το αμέσως λιγότερο συχνό, κ.ο.κ.), φροντίζουμε στην υλοποίησή μας να διατάξουμε τα αντικείμενα στο εσωτερικό τους με βάση τη συχνότητά τους πριν την αποτίμηση.

4.2.1 Κατασκευή ευρετηρίου

Η βασική κλάση της υλοποίησης του ευρετηρίου μας είναι η κλάση InvertedFile, ο ορισμός της οποίας φαίνεται στον κώδικα 4.1. που ακολουθεί.

```
class InvertedFile {
public:
    InvertedFile(char * in);
```

```
    ~InvertedFile(void);
    void EvaluateQueries (char * datafile, char * in, char *
                          out, char * statsfile);
    void Print (void);
private:
    InvertedItem ** items;
    void AddTransaction (Transaction trans);
    MemList * SubsetQuery (Query query, int &btp, int &ifp);
    MemList * EqualityQuery (Query query, int &btp, int &ifp);
    MemList * SupersetQuery (Query query, int &btp, int &ifp);
    Statistics * CreateStats (char * statsfile);
    void CountPages (int &btp, int &ifp, int &itp);
};
```

Κώδικας 4.1.: Ο ορισμός της κλάσης InvertedFile

Η κλάση InvertedFile είναι αυτή που υλοποιεί το ευρετήριο και περιέχει έναν πίνακα με δείκτες σε InvertedItems. Το InvertedItem υλοποιεί το τμήμα του ευρετηρίου που αντιστοιχεί σε ένα συγκεκριμένο αντικείμενο, δηλαδή την ανεστραμμένη λίστα του αντικειμένου αυτού και το B-Δέντρο που βρίσκεται πάνω από αυτήν, και ο ορισμός του φαίνεται στον κώδικα 4.2. Ο λόγος που χρησιμοποιούμε πίνακα είναι για να προσομοιώσουμε τη λειτουργία ενός hash ευρετηρίου πάνω από το λεξικό του ευρετηρίου μας, αφού όπως είπαμε και στο προηγούμενο κεφάλαιο θεωρούμε ότι το οποιοδήποτε αντικείμενο μπορεί να προσπελαστεί σε ένα μόνο βήμα.

```
class InvertedItem {
public:
    InvertedItem (int item);
    ~InvertedItem (void);
    void AddTransaction (Transaction trans);
    void CheckLastEntry (void);
    void SearchTree (KeyFieldType startkey, KeyFieldType endkey,
                     PagePtr & startpage, PagePtr & endpage);
    void SearchTree (KeyFieldType endkey, PagePtr & startpage,
                     PagePtr & endpage);
    void GetPages (int &btpages, int &ifpages);
    int CountBTUsed (void);
    int CountIFUsed (void);
    ListPtr getList(void);
    void Print(void);
```

```
private:
    int itemno;
    BTTableClass * btree;
    ListPtr list;
    ItemType tree_item;
};
```

Κώδικας 4.2.: Ορισμός της κλάσης InvertedItem

Έτσι, όπως βλέπουμε, το κάθε InvertedItem περιέχει το id του αντικειμένου στο οποίο αντιστοιχεί (itemno), ένα δείκτη στην ανεστραμμένη λίστα του (list) και ένα δείκτη στο αντίστοιχο Β-Δέντρο (btree). Για να κατασκευάσουμε λοιπόν ένα ordered inverted file καλούμε τον κατασκευαστή της κλάσης InvertedFile ο οποίος παίρνει σαν όρισμα το input file και το διαβάζει φροντίζοντας να εισαχθούν οι δοσοληψίες που βρίσκει στις σωστές λίστες και στα αντίστοιχα Β-Δέντρα όπου αυτό χρειάζεται. Αυτό γίνεται με τη συνάρτηση AddTransaction του InvertedFile και τη συνάρτηση AddTransaction του InvertedItem.

Όπως είπαμε και στο προηγούμενο κεφαλαίο, η προσέγγιση που χρησιμοποιούμε για τη δημιουργία των Β-Δέντρων είναι να εισάγουμε την τελευταία δοσοληψία κάθε σελίδας μιας λίστας που γεμίζει στο αντίστοιχο Β-Δέντρο, και αφού τελειώσει η κατασκευή του ευρετηρίου να ελέγξουμε ποιες σελίδες δε γέμισαν τελείως για να εισάγουμε τις τελευταίες δοσοληψίες και αυτών στο δέντρο. Αυτό επιτυγχάνεται με τη χρήση της μεταβλητής tree_item που συναντούμε στο InvertedItem. Η μεταβλητή αυτή είναι τύπου εγγραφής στο Β-Δέντρο και παίρνει την τιμή της τελευταίας δοσοληψίας που εισήχθη στη λίστα αν αυτή δεν έχει εισαχθεί στο Β-Δέντρο. Τελειώνοντας λοιπόν την κατασκευή του ευρετηρίου δε χρειάζεται να διατρέξουμε ολόκληρες τις λίστες για να φτάσουμε στην τελευταία σελίδα της καθεμίας και να δούμε αν αυτή είναι γεμάτη ή όχι – αν δηλαδή η τελευταία δοσοληψία της λίστας έχει εισαχθεί στο Β-Δέντρο ή όχι. Αρκεί για κάθε λίστα να ελέγξουμε τη μεταβλητή tree_item η οποία θα είναι κενή αν η τελευταία δοσοληψία της λίστας έχει εισαχθεί στο Β-Δέντρο ή αλλιώς θα έχει την εγγραφή που πρέπει να εισάγουμε. Ο έλεγχος αυτός γίνεται με τη συνάρτηση CheckLastEntry του InvertedItem.

Μένει λοιπόν τώρα να δούμε πώς υλοποιούνται στον κώδικά μας τα Β-Δέντρα και οι ανεστραμμένες λίστες. Η υλοποίηση των Β-Δέντρων γίνεται με την κλάση BTTableClass, η οποία κληρονομεί από την κλάση TableBaseClass. Οι δύο αυτές κλάσεις φαίνονται στον κώδικα 4.3., όπου έχουμε παραλείψει τις συναρτήσεις της TableBaseClass και ένα μεγάλο τμήμα των συναρτήσεων της κλάσης BTTableClass, για λόγους απλότητας.

```
class TableBaseClass{
protected:
      fstream DataFile;   // the file stream for the table data
```

```
        long NumItems;      // number of records of type ItemType
                              in the table
        char OpenMode;      // r or w (read or write) mode for the
                              table
};
typedef struct {
   int Count;       // Number of keys stored in the current node
   bool Used;       //Whether this node has been accessed
                      while evaluating the current query or not
   ItemType Key[MaxKeys];
   long Branch[MaxKeysPlusOne];  // Fake pointers to child nodes
} NodeType;

class BTTableClass: public TableBaseClass {
public:
      long NumNodes;   // number of nodes in the B-tree
      BTTableClass(char Mode, char * FileName);
      ~BTTableClass(void);
      DataFieldType Search(KeyFieldType Target);
private:
      long Root;       // fake pointer to the root node
      int NodeSize;    // number of bytes per node
      NodeType CurrentNode;   // storage for current node being
                                 worked on
      bool SearchNode(const KeyFieldType Target, int & location)
                                                          const;
};
```

Κώδικας 4.3.: Ορισμός κλάσεων που υλοποιούν το Β-Δέντρο

Οι συναρτήσεις που έχουμε παραλείψει από τον ορισμό της BTTableClass είναι αυτές που υλοποιούν την προσθήκη και διαγραφή κόμβων στο δέντρο, φροντίζοντας αυτό να παραμένει ισοσκελισμένο και όλοι οι κόμβοι να είναι τουλάχιστον μισογεμάτοι, και προέρχονται από το [Car05]. Το σημείο της κατασκευής του Β-Δέντρου που παρουσιάζει το μεγαλύτερο ενδιαφέρον είναι η αποθήκευσή του στο δίσκο.

Όπως προείπαμε, το κάθε Β-Δέντρο αποθηκεύεται σε ένα δυαδικό αρχείο. Αυτό γίνεται με το να γράφουμε στο αρχείο αυτό σε δυαδική μορφή τους κόμβους του δέντρου με τη σειρά που αυτοί κατασκευάζονται και σε κάθε κόμβο αντί να αποθηκεύουμε δείκτες προς τους κόμβους – παιδιά του, να αποθηκεύουμε αριθμούς που δείχνουν τη θέση των κόμβων – παιδιών στο

αρχείο. Αυτό φαίνεται στη δομή NodeType του κώδικα 4.3. όπου ο πίνακας Branch που περιλαμβάνει τους δείκτες του κόμβου προς τα παιδιά του είναι τύπου long και όχι τύπου δείκτη. Έτσι, η αναζήτηση ενός συγκεκριμένου κόμβου του δέντρου στο αρχείο και το διάβασμα και γράψιμο αυτού γίνονται με τα αποσπάσματα κώδικα που ακολουθούν (κώδικας 4.4.).

```
DataFile.seekg(p * NodeSize, ios::beg);

DataFile.read(reinterpret_cast <char *> (&CurrentNode),
                                            NodeSize);

DataFile.write(reinterpret_cast <char *> (&CurrentNode),
                                             NodeSize);
```

Κώδικας 4.4.: Αναζήτηση, διάβασμα και εγγραφή κόμβου του Β-Δέντρου στο αντίστοιχο αρχείο

Αξίζει επιπλέον να αναφέρουμε τον τρόπο με τον οποίο αναπαριστούμε τα κλειδιά του Β-Δέντρου, καθένα από τα οποία θα πρέπει όπως έχουμε ήδη πει να περιέχει τα δεδομένα της δοσοληψίας στην οποία αντιστοιχεί καθώς επίσης και το id της. Αρχικά, τα κλειδιά αυτά αναπαραστάθηκαν σα strings, έτσι ώστε η δοσοληψία με id = i και περιεχόμενα τα αντικείμενα a, b, c, d και e να αντιστοιχεί στο string a_b_c_d_e_Ti. Το πρόβλημα της αναπαράστασης αυτής είναι ότι είναι πολύ απαιτητική σε χώρο, αφού για το κάθε αντικείμενο της κάθε δοσοληψίας πρέπει να δεσμευθεί χώρος ίσο με το πλήθος των ψηφίων του μεγαλύτερου αντικειμένου της βάσης. Υποθέτοντας ένα λεξικό με περισσότερα από 1000 στοιχεία το οποίο είναι αναμενόμενο και ισχύει στα πειράματά μας πρέπει να δεσμεύσουμε 4 bytes για κάθε αντικείμενο. Έτσι, αποφασίσαμε τελικά να χρησιμοποιήσουμε για την αποθήκευση των αντικειμένων της δοσοληψίας έναν πίνακα από short int, μειώνοντας έτσι το χώρο που απαιτείται για το κάθε αντικείμενο στα 2 bytes, ενώ το id της δοσοληψίας αποθηκεύεται ξεχωριστά σαν long int. Η δομή που χρησιμοποιείται για τα κλειδιά του δέντρου φαίνεται στον κώδικα 4.5.

```
typedef struct {
    short int items [KeyFieldMax];
    int length;
    long int tid;
}KeyFieldType;
```

Κώδικας 4.5.: Ο ορισμός των κλειδιών του Β-Δέντρου

Αντίστοιχα, οι ανεστραμμένες λίστες υλοποιούνται με τις δομές και τις κλάσεις που φαίνονται στον κώδικα 4.6.

```
class TransactionData {           //Transaction
public:
    long int id;                  //Transaction identifier
    short int length;             //Transaction length
    TransactionData(void);
    TransactionData (long int tid, short int tlength);
    void Print(void);
};

typedef union pageptr {
    long disk;
    struct page * mem;
} PagePtr;

typedef struct page{
    TransactionData * table_of_trans;
    bool used;                //shows whether this page has been
                                used during the current query
    PagePtr next;            //pointer to next page
} PageData;

class DiskList {            //List of inverted file pages stored
                              in a file in the hard disk
public:
    int numpages;           //how many pages this list has
    long start;             //the first page of the list
    PageData CurrentPage;   //the current page being worked on
    DiskList(char * file);
    void OpenFile (char * file);
    void CloseFile (void);
    bool isOpen (void);
    void WritePage (long place);
    void ReadPage (long place);
    long AddTransaction (TransactionData data, int& offset);
    long getNextPage(long current);
    void SetUsed (long page);
    int CountUsed (void);
    void Print(void);
```

```
private:
    fstream DataFile;
    void InitCurrentPage (void);
};

class MemList { //List of inverted file pages in the main memory
public:
    int numpages;                 //how many pages this list has
    PageData * start;
    MemList (void);
    PageData * AddTransaction (TransactionData data,
                               int & offset);
    void MergeJoin (PagePtr * startarray, PagePtr * endarray,
                    int * items, union listptr * lists,
                    int listlength, int qlength);
    void MergeJoin (PagePtr * startarray, PagePtr * endarray,
                int * items, union listptr * lists, int length);
    void Copy (MemList * page);
    void AddMemList (MemList * list);
    int CountUsed (void);
    void Print (void);
private:
    void InitPage (PageData ** current);
};

typedef union listptr {
    DiskList * disk;
    MemList * mem;
} ListPtr;
```

Κώδικας 4.6.: Ορισμοί κλάσεων και δομών που υλοποιούν τις ανεστραμμένες λίστες

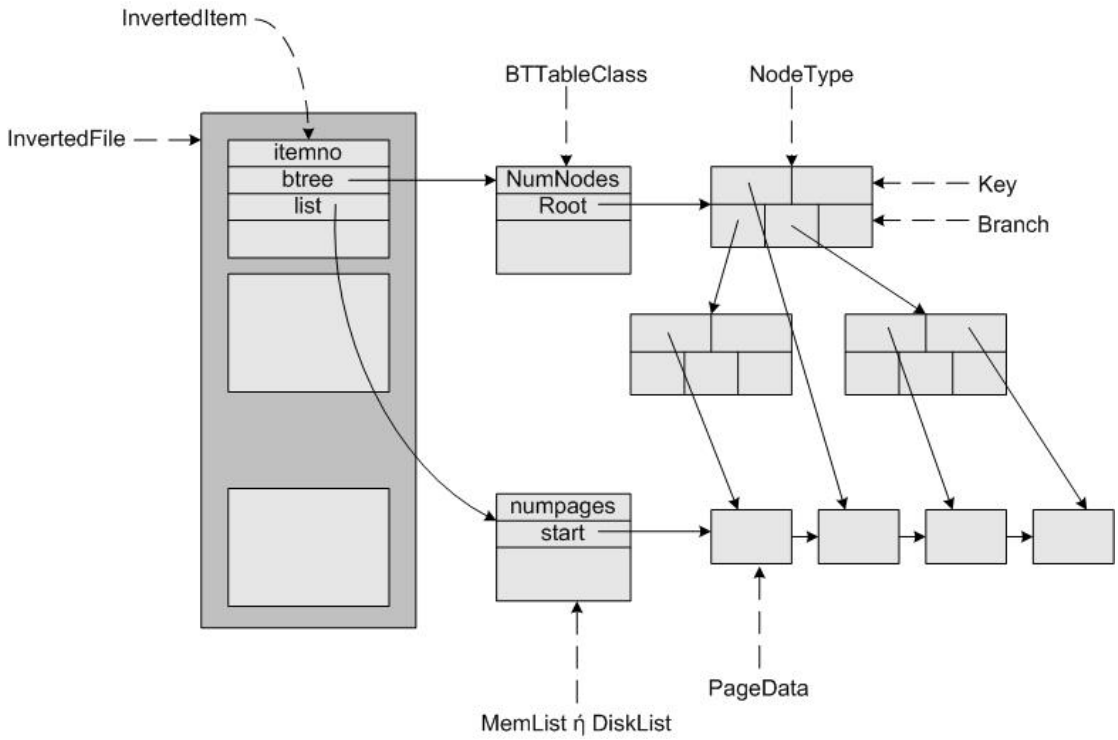
Από τις παραπάνω κλάσεις, η κλάση TransactionData υλοποιεί μία εγγραφή μίας λίστας που αντιστοιχεί σε μία δοσοληψία. Έτσι, περιέχει όπως είναι αναμενόμενο, εκτός από τους καρτασκευαστές της, μία μεταβλητή για την αποθήκευση του id της δοσοληψίας (id) και μία για την αποθήκευση του μήκους της (length). Η δομή PageData κρατάει τα δεδομένα που βρίσκονται αποθηκευμένα σε μία σελίδα της λίστας, δηλαδή έναν πίνακα με δοσοληψίες (table_of_trans) και ένα δείκτη προς την επόμενη σελίδα (next). Ο δείκτης αυτός (δομή PagePtr) είναι τύπου union και η τιμή του μπορεί να είναι ή pointer ή long, αφού, όπως

εξηγήσαμε και κατά την ανάλυση της υλοποίησης του Β-Δέντρου, στην περίπτωση που θέλουμε να αποθηκεύσουμε τις λίστες σε αρχεία στο δίσκο πρέπει να αντικαταστήσουμε όλους τους δείκτες με αριθμούς που προσδιορίζουν τη θέση της κάθε σελίδας στο αρχείο. Το πεδίο used της δομής PageData χρησιμοποιείται κατά την αποτίμηση ερωτημάτων και θα εξεταστεί αργότερα.

Τέλος, οι κλάσεις DiskList και MemList περιλαμβάνουν τις συναρτήσεις και μεταβλητές που είναι απαραίτητες για το χειρισμό των σελίδων των ανεστραμμένων λιστών όταν αυτές είναι αποθηκευμένες στο δίσκο ή στη μνήμη αντίστοιχα. Έτσι, το κάθε InvertedItem έχει ένα δείκτη σε μία από αυτές τις δύο κλάσεις (μία μεταβλητή τύπου ListPtr), ανάλογα με τη μέθοδο αποθήκευσης των λιστών που έχει επιλεγεί.

Και στις δύο κλάσεις υπάρχει μία μεταβλητή στην οποία αποθηκεύεται το πλήθος των σελίδων που περιέχει η λίστα (numpages) καθώς και ένας δείκτης προς την πρώτη σελίδα της λίστας (start) ο οποίος στην περίπτωση της DiskList είναι τύπου long για τους λόγους που εξηγήσαμε παραπάνω, ενώ στην περίπτωση της MemList είναι ένας απλός δείκτης σε μία σελίδα (PageData *). Η DiskList περιλαμβάνει επίσης μία μεταβλητή για το αρχείο στο οποίο αποθηκεύεται η λίστα (DataFile) και μία όπου κρατιέται η σελίδα στην οποία εργαζόμαστε ανά πάσα στιγμή, αφού οι όποιες αλλαγές μπορεί να θέλουμε να κάνουμε σε μία σελίδα δεν μπορούν να γίνουν απευθείας στο αρχείο όπου οι σελίδες αποθηκεύονται σε δυαδική μορφή. Οι εντολές που χρησιμοποιούμε για την εγγραφή και το διάβασμα των σελίδων από το αρχείο είναι αντίστοιχες με αυτές που χρησιμοποιούνται και από το Β-Δέντρο. Τέλος, η προσθήκη δοσοληψιών στη λίστα κατά την κατασκευή του ευρετηρίου γίνεται με τη συνάρτηση AddTransaction που υπάρχει και στις δύο κλάσεις και φροντίζει επιπλέον να προσθέτει καινούργιες σελίδες στο τέλος της λίστας όποτε αυτή εξαντλείται.

Έτσι, μπορούμε να φανταστούμε την υλοποίηση του ordered inverted file περίπου όπως φαίνεται στο σχήμα 4.1.



Σχήμα 4.1.: Υλοποίηση ordered inverted file

Σημειώνουμε εδώ ότι έχουμε κατασκευάσει επίσης τις κλάσεις Data και Transaction οι οποίες αντιστοιχούν σε ένα αρχείο εισόδου και μία δοσοληψία που περιέχεται σε ένα τέτοιο αρχείο, και χρησιμεύουν στο διάβασμα των αρχικών δεδομένων για την κατασκευή του ευρετηρίου. Αντίστοιχα έχει κατασκευαστεί η κλάση Query που αντιστοιχεί σε ένα ερώτημα του αρχείου ερωτημάτων, και οι κλάσεις Results και Statistics που βοηθούν στην έξοδο των αποτελεσμάτων και των δεδομένων αποτίμησης ερωτημάτων στα κατάλληλα αρχεία. Οι κλάσεις αυτές δεν αναλύονται περαιτέρω καθώς δεν παρουσιάζουν ιδιαίτερο προγραμματιστικό ενδιαφέρον και η ακριβής λειτουργία τους δε μας απασχολεί στα πλαίσια της ανάλυσης του ευρετηρίου που κάνουμε εδώ.

Τέλος, είναι σκόπιμο να κάνουμε μία σύντομη αναφορά των global σταθερών και μεταβλητών που χρησιμοποιούμε ως παραμέτρους για την κατασκευή και λειτουργία του ευρετηρίου. Αυτές φαίνονται στον κώδικα 4.7.

```
const long NilPtr = -1L;
const bool DISK = false;          // if true the inverted file
                                     is kept on disk


//constants
#define MAX_TRANS_LENGTH 25       //maximum transaction length
```

```
#define MAX_TRANS 100000          //maximum number of
                                    transactions
#define MAX_TRANS_DIGITS 6        //maximum number of
                                    transaction digits
#define MAX_ITEM 5000             //maximum number of items
#define MAX_ITEM_DIGITS 4         //maximum number of item
                                   digits
#define MAX_LINE_LENGTH
MAX_TRANS_LENGTH*(MAX_ITEM_DIGITS+1)+MAX_TRANS_DIGITS+1
                         //maximum length an input line can have

#define PAGE_SIZE 682             //number of TransactionData
                                    entries a diskpage can have
#define IT_PAGE_SIZE 1024         //number of transactions an
                                intermediate table page can hold

//BTree constants
const int MaxKeys = 62;   // max number of keys in a node
const int MaxKeysPlusOne = MaxKeys + 1;
const int MinKeys = 31;    // min number of keys in a node

//Query evaluation parameters
int BTREE_THRESH;           //if the btree does not have more
                    pages than this do not use it in the queries
int IFILE_THRESH;           //if the inverted file does not have
                             more pages than this do not use the
                             btree in the queries
```

Κώδικας 4.7.: Παράμετροι κατασκευής και λειτουργίας του ordered inverted file

Η σταθερά NilPtr αντιστοιχεί στο -1 για να προσομοιώσει έτσι το Null όταν εργαζόμαστε με αριθμούς long και όχι δείκτες κατά την αποθήκευση των Β-Δέντρων ή των λιστών σε αρχεία, ενώ η σταθερά DISK έχει την τιμή true όταν θέλουμε οι ανεστραμμένες λίστες να αποθηκευτούν στο δίσκο και όχι στην κύρια μνήμη. Οι σταθερές που ακολουθούν ορίζουν κάποια σταθερά στοιχεία της βάσης πάνω στην οποία θα λειτουργήσει το ευρετήριο, όπως το πλήθος δοσοληψιών και το μέγεθος του λεξικού, διευκολύνοντας έτσι την κατασκευή του. Η σταθερά PAGE_SIZE ορίζει το πλήθος των δοσοληψιών (TransactionData) που χωράνε σε μία σελίδα μιας ανεστραμμένης λίστας, δηλαδή το μέγεθος του πίνακα table_of_trans που είδαμε στη δομή PageData, ενώ η σταθερά IT_PAGE_SIZE ορίζει το πλήθος των εγγραφών

που χωράνε σε μία σελίδα του ενδιάμεσου πίνακα. Ο πίνακας αυτός δεν υλοποιείται εδώ, αλλά κατά την αποτίμηση των δοσοληψιών μετράμε τις σελίδες αυτού που θα χρειαζόταν να διαβάσουμε για να προσπελάσουμε τα αποτελέσματά μας, πράγμα που κάνει τη σταθερά αυτή απαραίτητη. Οι τιμές που δίνουμε σε αυτές τις σταθερές θα εξηγηθούν στο κεφάλαιο 5. Αντίστοιχα, οι σταθερές MaxKeys και MinKeys ορίζουν το μέγιστο και το ελάχιστο πλήθος κλειδιών που μπορεί να έχει ένας κόμβος του Β-Δέντρου. Τέλος, οι μεταβλητές BTREE_THRESH και IFILE_THRESH ορίζουν ένα κατώφλι το οποίο αν είναι μεγαλύτερο από το πλήθος σελίδων του Β-Δέντρου ή της ανεστραμμένης λίστας ενός αντικειμένου αντίστοιχα, δε χρησιμοποιείται το Β-Δέντρο του αντικειμένου αυτού κατά την αποτίμηση ερωτημάτων, αλλά χρησιμοποιείται ολόκληρη η λίστα. Η λειτουργία αυτή θα αναλυθεί περεταίρω στην επόμενη ενότητα.

4.2.2 Επεξεργασία ερωτημάτων

Στη συνέχεια εξετάζουμε τον τρόπο που υλοποιείται η αποτίμηση των ερωτημάτων. Αυτό γίνεται με την κλήση της συνάρτησης EvaluateQueries της κλάσης InvertedFile, η οποία φροντίζει να καλέσει τις συναρτήσεις SubsetQuery, EqualityQuery και SupersetQuery η καθεμία από τις οποίες αποτιμά το αντίστοιχο ερώτημα, και να γράψει τα αποτελέσματα της αποτίμησης στα κατάλληλα αρχεία. Οι συναρτήσεις SubsetQuery, EqualityQuery και SupersetQuery, μαζί με κάποιες από τις συναρτήσεις που καλούν όπως η συνάρτηση εύρεσης ενός κλειδιού στο Β-Δέντρο και η MergeJoin, είναι αυτές που παρουσιάζουν το μεγαλύτερο ενδιαφέρον και θα εξεταστούν λεπτομερειακά παρακάτω.

4.2.2.1 Subset queries

Η αποτίμηση των subset queries γίνεται όπως είπαμε παραπάνω με τη συνάρτηση SubsetQuery. Η συνάρτηση αυτή φτιάχνει αρχικά δύο πίνακες από PagePtr (δείκτες σε σελίδες της ανεστραμμένης λίστας), τους startarray και endarray, στους οποίους θα αποθηκευτούν τα άκρα των περιοχών πιθανών απαντήσεων των λιστών των αντικειμένων που περιλαμβάνονται στο query set. Ο προσδιορισμός των περιοχών αυτών γίνεται με την κλήση της συνάρτησης SearchTree (KeyFieldType endkey, PagePtr & startpage, PagePtr & endpage) του InvertedItem, η οποία φαίνεται στον κώδικα 4.8.

```
void InvertedItem::SearchTree (KeyFieldType endkey, PagePtr &
startpage, PagePtr & endpage) {
    btree->OpenFile('w', itoa(itemno));
    if ((btree->NumNodes<=BTREE_THRESH)||((DISK)&&(list.disk-
>numpages<=IFILE_THRESH))||((!DISK)&&(list.mem-
>numpages<=IFILE_THRESH))) {
```

```
            btree->CloseFile();
            if (DISK)
                  startpage.disk=list.disk->start;
            else
                  startpage.mem=list.mem->start;
            if (DISK)
                  endpage.disk=NilPtr;
            else
                  endpage.mem=NULL;
            return;
      }
      if (DISK)
            startpage.disk=list.disk->start;
      else
            startpage.mem=list.mem->start;
      DataFieldType enddata=btree->Search(endkey);
      endpage=enddata;
      btree->CloseFile();
  }
```

Κώδικας 4.8.: Η συνάρτηση SearchTree (KeyFieldType endkey, PagePtr & startpage, PagePtr & endpage) της κλάσης InvertedItem

Η συνάρτηση αυτή επιστρέφει στη μεταβλητή startpage την πρώτη σελίδα της λίστας του αντικειμένου αυτού, και στην endpage τη σελίδα την οποία παίρνουμε αν ψάξουμε στο Β-Δέντρο του αντικειμένου για το κλειδί endkey. Αν το Β-Δέντρο έχει λιγότερους κόμβους από το περιεχόμενο της μεταβλητής BTREE_THRESH ή η λίστα έχει λιγότερες σελίδες από το IFILE_THRESH τότε δε χρησιμοποιούμε το Β-Δέντρο για να βρούμε το τέλος της περιοχής πιθανών απαντήσεων, αλλά χρησιμοποιούμε ολόκληρη τη λίστα κατά την αποτίμηση του ερωτήματος. Αυτό γίνεται με τον έλεγχο που κάνει το if στην τέταρτη γραμμή του κώδικα 4.8. Ο έλεγχος αυτός μας δίνει τη δυνατότητα να μη χρησιμοποιούμε το Β-Δέντρο σε περιπτώσεις όπου αυτό ή η αντίστοιχη λίστα είναι πολύ μικρά και άρα η χρήση της περιοχής πιθανών απαντήσεων αντί για ολόκληρη τη λίστα δε μας προσφέρει κάποιο σημαντικό πλεονέκτημα. Στην περίπτωση που δε χρησιμοποιηθεί κανένα Β-Δέντρο κατά την αποτίμηση κάποιου ερωτήματος τότε αυτή γίνεται όπως θα γινόταν και με χρήση του κλασσικού inverted file.

Οι σελίδες που επιστρέφει η συνάρτηση SearchTree που είδαμε παραπάνω, υπό τον όρο ότι έχουμε ορίσει σωστά το endkey σύμφωνα με τα όσα είπαμε στο κεφαλαίο 3, αποτελούν τα άκρα της περιοχής πιθανών απαντήσεων της λίστας του αντικειμένου αυτού. Έτσι, καλώντας

τη συνάρτηση αυτή για όλα τα αντικείμενα του query set μπορούμε να αποθηκεύσουμε στον πίνακα startarray τις πρώτες σελίδες των περιοχών πιθανών απαντήσεων, και στο endarray τις τελευταίες. Οι σελίδες τοποθετούνται στου δύο αυτούς πίνακες με βάση τη διάταξη των αντικειμένων που έχουμε ήδη κάνει, δηλαδή πρώτα η σελίδα που αντιστοιχεί στο πιο συχνό αντικείμενο και τελευταία αυτή που αντιστοιχεί στο λιγότερο συχνό.

Στη συνέχεια, περνάμε τους πίνακες αυτούς ως ορίσματα στη συνάρτηση MergeJoin (PagePtr * startarray, PagePtr * endarray, int * items, union listptr * lists, int listlength, int qlength) της κλάσης MemList η οποία θα μας δώσει την τομή των περιοχών που ορίζονται από τα άκρα αυτά, άρα και την απάντηση του ερωτήματος. Η συνάρτηση αυτή λειτουργεί με τον τρόπο που είδαμε στο προηγούμενο κεφάλαιο (κώδικας 3.1.). Διασχίζει τη λίστα του λιγότερο συχνού αντικειμένου από αυτά που της δώσαμε σαν ορίσματα, ξεκινώντας από την σελίδα του πίνακα startarray που αντιστοιχεί στο αντικείμενο αυτό και σταματώντας σε αυτή του endarray. Για κάθε δοσοληψία που συναντάει στη λίστα αυτή ελέγχει αν το μήκος της είναι ίσο με την παράμετρο qlength της συνάρτησης. Αν αυτό συμβαίνει, ή αν το qlength είναι ίσο με το μηδέν, τότε ψάχνει για τη δοσοληψία αυτή στη λίστα του αμέσως πιο συχνού αντικειμένου, πάλι μέσα στα όρια που ορίζονται από τους πίνακες startarray και endarray. Σταματάει το ψάξιμο στη λίστα αυτή αν βρει τη δοσοληψία που ψάχνει ή αν βρει μία με μεγαλύτερο id, πράγμα το οποίο σημαίνει ότι η δοσοληψία που ψάχνει δεν υπάρχει αφού όλες οι δοσοληψίες αποθηκεύονται στις ανεστραμμένες λίστες κατά αύξον id. Αν η δοσοληψία βρεθεί συνεχίζει με την επόμενη λίστα, μετά με τη μεθεπόμενη κ.ο.κ. μέχρι να βρει κάποια λίστα η οποία δεν περιλαμβάνει τη δοσοληψία πράγμα το οποίο σημαίνει ότι η δοσοληψία δεν ανήκει στην τομή των λιστών, ή μέχρι να ανακαλύψει ότι η δοσοληψία περιέχεται σε όλες τις λίστες, οπότε θα την εισάγει και στην τομή τους. Η συνάρτηση τερματίζει όταν κάποια από τις λίστες εξαντληθεί, οπότε οι δοσοληψίες που δεν έχουν ελεγχθεί ακόμη στις υπόλοιπες λίστες δε μπορούν να ανήκουν στην τομή των λιστών.

Έτσι, καλώντας τη συνάρτηση αυτή με qlength = 0 παίρνουμε το αποτέλεσμα του subset query που μας ενδιαφέρει, αφού θα μας επιστραφεί η τομή των περιοχών πιθανών απαντήσεων ανεξαρτήτως του μήκους των δοσοληψιών. Μένει τέλος να δούμε πώς υλοποιείται η αναζήτηση για κάποιο κλειδί στο Β-Δέντρο, την οποία όπως αναφέραμε προηγουμένων χρησιμοποιεί τη συνάρτηση SearchTree για τον προσδιορισμό των άκρων της περιοχής πιθανών απαντήσεων. Η αναζήτηση αυτή γίνεται με τη συνάρτηση Search της κλάσης BTTableClass που φαίνεται στον κώδικα 4.9.

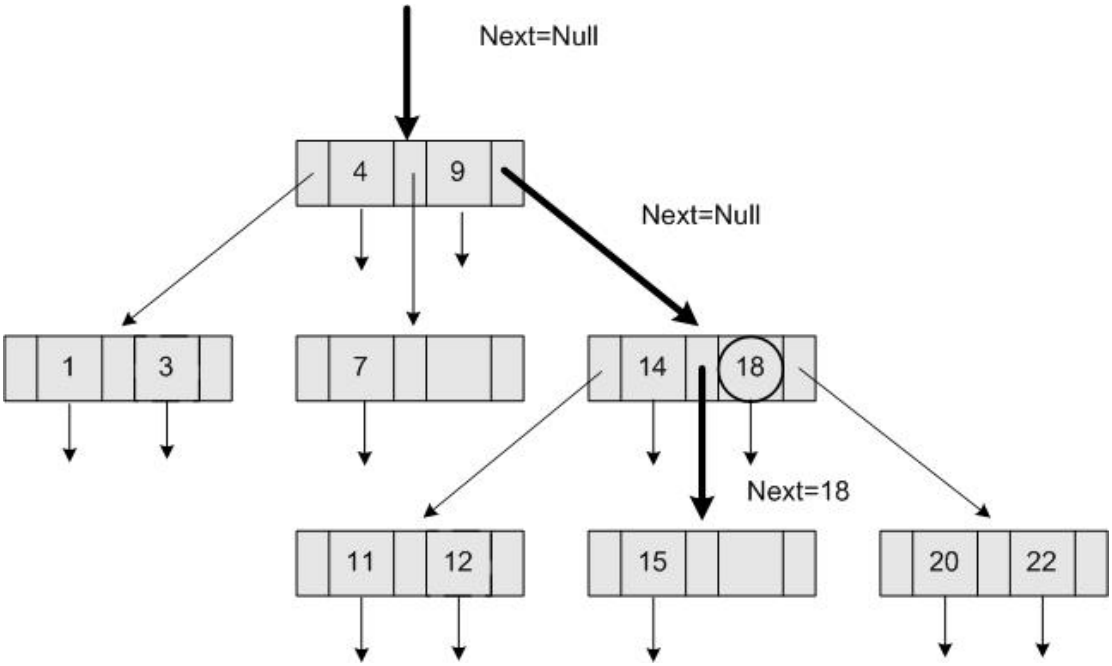
```
DataFieldType BTTableClass::Search(KeyFieldType Target) {
    long CurrentRoot = Root;
    int Location;
    bool Found=false;
```

```
DataFieldType Result, Next;
if (DISK)
      Next.disk=NilPtr;
else
      Next.mem=NULL;
if (CurrentRoot==NilPtr)
      error ("BTTableClass::Search", "The BTree is empty");
while (!Found) {
      DataFile.seekg(CurrentRoot * NodeSize, ios::beg);
      DataFile.read(reinterpret_cast <char *>
                                  (&CurrentNode), NodeSize);
      CurrentNode.Used=true;
      DataFile.seekg(CurrentRoot * NodeSize, ios::beg);
      DataFile.write(reinterpret_cast <char *>
                                  (&CurrentNode), NodeSize);
      if (SearchNode(Target, Location)) {
                                      //If the key was found
            Found=true;
            Result=CurrentNode.Key[Location].DataField;
      }
      else if (CurrentNode.Branch[Location+1]==NilPtr) {
      //If we reached the end of the path where it should be
            Found=true;
            if (Location+1>CurrentNode.Count-1)
                  Result=Next;
            else
               Result=CurrentNode.Key[Location+1].DataField;
      }
      else {
            CurrentRoot=CurrentNode.Branch[Location+1];
            if (Location<CurrentNode.Count-1)
                 Next=CurrentNode.Key[Location+1].DataField;
      }
}
return Result;}
```

Κώδικας 4.9.: Η συνάρτηση Search της κλάσης BTTableClass

Η Search ξεκινάει από τη ρίζα του δέντρου και καλώντας τη συνάρτηση SearchNode ψάχνει να βρει αν το κλειδί που ψάχνουμε βρίσκεται στον τρέχοντα κόμβο του δέντρου. Αν δε

βρεθεί η SearchNode θα επιστρέψει τα κλειδιά του τρέχοντα κόμβου ανάμεσα στα οποία θα έπρεπε να βρίσκεται, έστω i και i+1. Έτσι, αν το κλειδί βρεθεί, η Search επιστρέφει το δείκτη που αντιστοιχεί στο κλειδί αυτό. Αλλιώς προχωράει στον κόμβο – παιδί του τρέχοντος τα κλειδιά του οποίου βρίσκονται ανάμεσα στα κλειδιά i και i+1 του τρέχοντος κόμβου. Παράλληλα, αποθηκεύει το κλειδί i+1 – αν αυτό υπάρχει – στη μεταβλητή Next. Η διαδικασία αυτή συνεχίζεται μέχρι να βρεθεί το κλειδί που ψάχνουμε ή μέχρι να φτάσουμε σε κάποιο κόμβο – φύλλο. Στην περίπτωση που το κλειδί δε βρεθεί επιστρέφουμε το δείκτη που αντιστοιχεί στο αμέσως επόμενο κλειδί που υπάρχει στο φύλλο στο οποίο καταλήξαμε, αφού αυτό θα είναι το αμέσως μεγαλύτερο κλειδί από αυτό που ψάχνουμε. Αν αυτό δεν υπάρχει τότε επιστρέφουμε την τιμή της μεταβλητής Next η οποία με βάση τον τρόπο που διασχίσαμε το δέντρο θα περιέχει το αμέσως μεγαλύτερο κλειδί από αυτό που ψάχνουμε. Σημειώνουμε εδώ ότι η μεταβλητή Next αρχικοποιείται στην αρχή της συνάρτησης με Null, και αν τελικά αυτή είναι η τιμή που θα επιστραφεί από τη συνάρτηση, αυτό σημαίνει ότι δεν υπάρχει στο δέντρο – άρα και στη λίστα – κλειδί μεγαλύτερο από αυτό που ψάχνουμε, αφού στην περίπτωση αυτή η τιμή Null θα είχε αντικατασταθεί από κάποια άλλη. Η διαδικασία αυτή φαίνεται παραστατικά στο σχήμα 4.2.



Σχήμα 4.2.: Η λειτουργία της συνάρτησης Search της BTTableClass

Στο σχήμα αυτό παρουσιάζεται η αναζήτηση του κλειδιού 16. Με έντονο χρώμα φαίνεται η διαδρομή που ακολουθούμε στο δέντρο, όπως επίσης φαίνονται οι διαδοχικές τιμές που παίρνει η μεταβλητή Next. Τελικά, αφού το 16 δε βρίσκεται και δεν υπάρχει μεγαλύτερο κλειδί από αυτό στο φύλλο στο οποίο καταλήξαμε θα επιστραφεί ο δείκτης που αντιστοιχεί στην τελευταία τιμή του Next, δηλαδή το 18

4.2.2.2 Equality queries

Η αποτίμηση των equality queries γίνεται με τη συνάρτηση EqualityQuery της κλάσης InvertedFile η οποία λειτουργεί με τον ίδιο τρόπο με τη SubsetQuery, με εξαίρεση τις δύο παρακάτω διαφορές

- Αντί για τη συνάρτηση SearchTree (KeyFieldType endkey, PagePtr & startpage, PagePtr & endpage) καλείται για την εύρεση των ορίων των περιοχών πιθανών απαντήσεων η SearchTree (KeyFieldType startkey, KeyFieldType endkey, PagePtr & startpage, PagePtr & endpage) η οποία ψάχνει και για την αρχή της περιοχής στο Β-Δέντρο με βάση το κλειδί startkey αντί να επιστρέφει την πρώτη σελίδα της λίστας.
- Κατά την κλήση της MergeJoin θέτουμε την παράμετρο qlength ίση με το μήκος του ερωτήματος ούτως ώστε να συμπεριληφθούν στην τομή των λιστών μόνο οι δοσοληψίες με μήκος ίσο με αυτό του query.

Με τον τρόπο αυτό βλέπουμε ότι υλοποιείται ακριβώς η μέθοδος που περιγράψαμε στο κεφάλαιο 3 για την αποτίμηση των equality queries.

4.2.2.3 Superset queries

Η αποτίμηση των superset query γίνεται με τη συνάρτηση SupersetQuery, η οποία παρουσιάζει ορισμένες βασικές διαφορές σε σχέση με τις SubsetQuery και EqualityQuery αφού, όπως είδαμε και στο προηγούμενο κεφάλαιο, η μέθοδος αποτίμησης των superset queries διαφέρει αρκετά από αυτές των δύο άλλων ειδών ερωτημάτων. Έτσι, η συνάρτηση SupersetQuery λειτουργεί ως εξής.

Αρχικά αποθηκεύει και αυτή σε δύο πίνακες (startarray και endarray) τα άκρα των περιοχών πιθανών απαντήσεων, οι οποίες όμως αυτή τη φορά αναφέρονται μόνο στις πιθανές απαντήσεις που περιέχουν σίγουρα το πιο συχνό αντικείμενο του ερωτήματος, τις οποίες είδαμε στο κεφάλαιο 3. Στη συνέχεια περνάει τους δύο αυτούς πίνακες ως ορίσματα στη συνάρτηση MergeJoin (PagePtr * startarray, PagePtr * endarray, int * items, union listptr * lists, int length) η οποία, αν και έχει το ίδιο όνομα με τη συνάρτηση που χρησιμοποιούμε για να πάρουμε την τομή των λιστών κατά την αποτίμηση των subset και equality queries, έχει διαφορετική λειτουργία αφού ο ρόλος της είναι να κάνει τις απαραίτητες πράξεις ανάμεσα στις περιοχές πιθανών απαντήσεων που αναφέραμε παραπάνω για την αποτίμηση ενός superset query και λειτουργεί με τον τρόπο που φάνηκε στο κεφάλαιο 3 (κώδικας 3.2.).

Έτσι, η MergeJoin αυτή διατρέχει την περιοχή πιθανών απαντήσεων του περισσότερο συχνού αντικειμένου από αυτά που της δώσαμε, ελέγχοντας το μήκος κάθε δοσοληψίας που συναντάει. Αν το μήκος της δοσοληψίας είναι ίσο με ένα τότε αυτή εισάγεται αυτόματα στο αποτέλεσμα, ενώ σε αντίθετη περίπτωση αρχίζει να διατρέχει με τη σειρά και τις υπόλοιπες

λίστες μέχρι να βρει τη δοσοληψία αυτή σε τόσες λίστες όσο το μήκος της ή να αποφανθεί ότι αυτό δεν πρόκειται να γίνει με τον τρόπο που περιγράψαμε στην παράγραφο 3.2.3. Και σε αυτή την περίπτωση μπορεί να μη χρειαστεί να προσπελάσουμε ολόκληρες τις περιοχές πιθανών απαντήσεων των λιστών που χρησιμοποιούμε αφού η συνάρτηση τερματίζει μόλις φτάσει στο τέλος της περιοχής πιθανών απαντήσεων της λίστας του πιο συχνού αντικειμένου.

Στη συνέχεια, η συνάρτηση SupersetQuery καλεί αναδρομικά τον εαυτό της αφαιρώντας όμως το περισσότερο συχνό αντικείμενο του ερωτήματος, και επιστρέφει τελικά ως αποτέλεσμα την ένωση των αποτελεσμάτων που προκύπτουν από όλες τις αναδρομικές κλήσεις. Σημειώνουμε τέλος ότι ο προσδιορισμός των άκρων των περιοχών πιθανών απαντήσεων γίνεται με την ίδια συναρτήσεις που χρησιμοποιούνται και από την EqualityQuery.

4.2.2.4 Κόστος αποτίμησης ερωτημάτων

Στην ενότητα αυτή εξετάζουμε τον τρόπο υπολογισμού του κόστους της αποτίμησης των ερωτημάτων, ο οποίος μας χρησιμεύει κατά τη διενέργεια των πειραμάτων μας για να συγκρίνουμε την απόδοση του ordered inverted file με αυτή του απλού inverted file. Το κόστος αυτό μετριέται ως πλήθος σελίδων που προσπελαύνονται κατά την αποτίμηση ενός ερωτήματος, αφού η μεταφορά μίας σελίδας από το δίσκο στη μνήμη είναι σαφώς πιο χρονοβόρα από οποιαδήποτε άλλη πράξη που εκτελείται αποκλειστικά στη μνήμη, και μετριέται ξεχωριστά για τις σελίδες των Β-Δέντρων, των ανεστραμμένων λιστών και του ενδιάμεσου πίνακα.

Όπως είδαμε κατά την ανάλυση του τρόπου αποτίμησης των ερωτημάτων από την υλοποίησή μας, η μοναδική συνάρτηση που προσπελαύνει κόμβους των Β-Δέντρων του ευρετηρίου είναι η συνάρτηση Search της BTTableClass. Έτσι, για να μετρήσουμε τους κόμβους που προσπελαύνουμε σε κάθε αναζήτηση σε ένα Β-Δέντρο αρκεί η συνάρτηση αυτή να θέτει κάποιο flag στις σελίδες που προσπελαύνει. Για το σκοπό αυτό χρησιμοποιούμε τη μεταβλητή Used της δομής NodeType την οποία η Search κάνει ίση με true στους κόμβους που προσπελαύνει ($16^{η}$ γραμμή του κώδικα 4.9.). Έτσι, αφού προσδιορίσουμε τις περιοχές πιθανών απαντήσεων μίας λίστας διατρέχουμε το δέντρο της και μετράμε τους κόμβους με Used = true, θέτοντας ταυτόχρονα το Used ξανά ίσο με false. Αυτό γίνεται με τη συνάρτηση CountBTUsed της κλάσης InvertedItem. Έτσι, ακόμη και αν κατά τον προσδιορισμό μίας περιοχής πιθανών απαντήσεων κάποια σελίδα του Β-Δέντρου χρησιμοποιηθεί περισσότερες από μία φορές, θα μετρηθεί μόνο μία φορά, αφού θεωρούμε ότι η κύρια μνήμη που έχουμε στη διάθεσή μας είναι αρκετή για να κρατάμε ολόκληρο το Β-Δέντρο πάνω στο οποίο εργαζόμαστε.

Αντίστοιχα, για τη μέτρηση των σελίδων των ανεστραμμένων λιστών που προσπελαύνουμε χρησιμοποιούμε τη μεταβλητή used της δομής PageData, την οποία θέτουν οι συναρτήσεις MergeJoin που είναι οι μόνες που προσπελαύνουν σελίδες των λιστών κατά την αποτίμηση. Η μέτρηση των χρησιμοποιημένων σελίδων και ο μηδενισμός των μεταβλητών used γίνεται αυτή τη φορά με τη συνάρτηση CountIFUsed της κλάσης InvertedItem. Κατά την αποτίμηση των subset και equality queries η μέτρηση των σελίδων που προσπελάσαμε γίνεται πριν το τέλος της αντίστοιχης συνάρτησης και η κάθε σελίδα μετράται μία φορά αφού σε οποιαδήποτε περίπτωση δεν πρόκειται να προσπελαστεί περισσότερες από μία φορές. Κατά την αποτίμηση των superset queries μετράμε πριν την αναδρομική κλήση της συνάρτησης SupersetQuery μόνο τις σελίδες που προσπελάσαμε στη λίστα του περισσότερο συχνού αντικειμένου, αφού τις υπόλοιπες λίστες θα τις προσπελάσουμε ξανά κατά τις αναδρομικές κλήσεις. Έτσι, η κάθε σελίδα μίας λίστας θα μετρηθεί ακριβώς μία φορά αν την έχουμε προσπελάσει, αυτό όμως δε μας ενοχλεί ούτε σε αυτή την περίπτωση, καθώς όπως είδαμε στο κεφάλαιο 3 σε κάθε βήμα της αναδρομής οι περιοχές πιθανών απαντήσεων των λιστών τις οποίες προσπελαύνουμε είναι διαφορετικές από αυτές που προσπελάσαμε στο προηγούμενο βήμα. Η μοναδική περίπτωση να προσπελάσουμε μία σελίδα περισσότερες από μία φορές είναι η τελευταία σελίδα της περιοχής πιθανών απαντήσεων κάποιας λίστας στο βήμα i της αναδρομής να ταυτίζεται με την πρώτη σελίδα της περιοχής πιθανών απαντήσεων της ίδιας λίστας στο βήμα i+1 της αναδρομής, αλλά και σε αυτή την περίπτωση δε χρειάζεται να μετρήσουμε τη σελίδα αυτή δύο φορές γιατί θεωρούμε ότι έχουμε αρκετή κύρια μνήμη ώστε να κρατάμε εκεί μία σελίδα από κάθε λίστα και άρα στην περίπτωση αυτή η εν λόγω σελίδα δε θα χρειαστεί να μεταφερθεί ξανά από το δίσκο.

Τέλος, ο υπολογισμός των σελίδων του ενδιάμεσου πίνακα γίνεται κατά την εκτύπωση των αποτελεσμάτων αφού όπως προείπαμε ο πίνακας αυτός δεν υλοποιείται, απλά προσομοιώνεται η λειτουργία του. Έτσι, σύμφωνα με αυτά που είπαμε στην παράγραφο 3.2.4., για να μετρήσουμε τις σελίδες αυτές παίρνουμε το υπόλοιπο της διαίρεσης του κάθε id με τη σταθερά IT_PAGE_SIZE που είδαμε παραπάνω. Αν το υπόλοιπο αυτό είναι διαφορετικό από εκείνο του προηγούμενου id τότε η τρέχουσα δοσοληψία βρίσκεται σε διαφορετική σελίδα από την προηγούμενή της και έτσι αυξάνουμε το πλήθος των σελίδων του ενδιάμεσου πίνακα που προσπελάστηκαν κατά ένα.

5

Πειράματα

Στο κεφάλαιο αυτό παρουσιάζουμε πειράματα που εξετάζουν την απόδοση του ordered inverted file και τη συγκρίνουν με αυτή του κλασσικού inverted file. Η απόδοση των δύο ευρετηρίων μετράται με βάση το κόστος αποτίμησης ερωτημάτων, με τον τρόπο που αναλύθηκε στην παράγραφο 4.2.2.4. Σημειώνουμε επίσης, ότι ο κώδικας που δημιουργήσαμε για την υλοποίηση του ordered inverted file μπορεί να λειτουργήσει και σαν απλό inverted file αν θέσουμε μία από τις μεταβλητές BTREE_THRESH και IFILE_THRESH ίση με έναν πολύ μεγάλο αριθμό, πράγμα που θα μας διασφαλίσει ότι δεν πρόκειται να χρησιμοποιηθεί κανένα από τα Β-Δέντρα κατά την αποτίμηση των ερωτημάτων, και μεταβάλλουμε ελαφρώς τον τρόπο που μετριούνται οι σελίδες που προσπελάστηκαν κατά την αποτίμηση του superset query, ούτως ώστε η κάθε σελίδα να μετράται τόσες φορές όσες προσπελάστηκε, αφού τώρα σε κάθε βήμα της αναδρομής θα προσπελαύνονται ολόκληρες (ή σχεδόν ολόκληρες) οι λίστες τις οποίες χρησιμοποιούμε στο βήμα αυτό. Έτσι, τα πειράματα που αναφέρονται στο απλό inverted file στη συνέχεια του κεφαλαίου έχουν γίνει με τον τρόπο αυτό.

5.1 Μεθοδολογία

5.1.1 Τα δεδομένα

Τα πειράματα που παρουσιάζονται στη συνέχεια έγιναν πάνω σε συνθετικά δεδομένα που κατασκευάστηκαν ειδικά για αυτό το σκοπό. Τα δεδομένα που κατασκευάσαμε ακολουθούν

zipfian κατανομή με Z=1, Z=50 και Z=99. Επιπλέον, για κάθε διαφορετική τιμή του Z κατασκευάσαμε δεδομένα με μέγεθος λεξιλογίου 2.000, 5.000, και 10.000 και με πλήθος δοσοληψιών 100.000, 250.000 και 1.000.000. Τέλος, φροντίσαμε οι δοσοληψίες σε κάθε περίπτωση να έχουν μήκη από 2 μέχρι και 23 αντικείμενα.

5.1.2 Τα ερωτήματα

Στα πειράματα αυτά αποφασίσαμε να μετρήσουμε το κόστος αποτίμησης ερωτημάτων τα οποία έχουν απάντηση στη βάση μας, προσέγγιση που έχει ήδη ακολουθηθεί και σε άλλες εργασίες [HM99]. Για το λόγο αυτό χρησιμοποιούμε ως query sets για τα ερωτήματά μας κάποιες από τις δοσοληψίες που βρίσκονται μέσα στη βάση. Οι δοσοληψίες αυτές φροντίζουμε να είναι ομοιόμορφα κατανεμημένες μέσα στη βάση πριν την αναδιάταξη των δοσοληψιών που κάνουμε με βάση τη συχνότητα των αντικειμένων τους, αφού η αναδιάταξη αυτή είναι χαρακτηριστικό της δικής μας μεθόδου και όχι της λειτουργίας του inverted file εν γένει. Με τον τρόπο αυτό διασφαλίζουμε ότι οι απαντήσεις στα αντίστοιχα ερωτήματα θα είναι διάσπαρτες μέσα στις ανεστραμμένες λίστες και δε θα βρίσκονται όλες στην αρχή ή όλες στο τέλος. Έτσι, από κάθε σύνολο δεδομένων επιλέγουμε δοσοληψίες – ερωτήματα από κάθε μήκος, για όλα τα μήκη από 2 μέχρι και 20.

5.1.3 Ορισμός σταθερών

Όπως είδαμε στο κεφάλαιο 4, έχουμε ορίσει στον κώδικά μας κάποιες σταθερές, οι τιμές των οποίων μπορούν να επηρεάσουν τη λειτουργία του ευρετηρίου μας. Έτσι εξετάζουμε εδώ τις τιμές που δίνουμε στις σταθερές αυτές κατά τη διενέργεια των πειραμάτων.

Η σταθερά PAGE_SIZE ορίζει, όπως είπαμε και προηγουμένως, το πλήθος των δοσοληψιών (TransactionData) που χωράνε σε μία σελίδα μιας ανεστραμμένης λίστας. Θεωρώντας λοιπόν μέγεθος σελίδας ίσο με 4KBytes και αφού το μέγεθος ενός TransactionData είναι $sizeof(id) + sizeof(length) = sizeof(long) + sizeof(short) = 4 + 2 = 6Bytes$ ορίζουμε τη σταθερά αυτή ίση με $4096/6 \cong 682$.

Αντίστοιχα, για τη σταθερά IT_PAGE_SIZE που ορίζει το πλήθος των εγγραφών που χωράνε σε μία σελίδα του ενδιάμεσου πίνακα, θεωρούμε ότι αφού κάθε εγγραφή του πίνακα αποτελείται μόνο από μία διεύθυνση στο δίσκο (η οποία θα είναι ένας long int), σε μία σελίδα θα χωράνε $4096/4 \cong 1024$ τέτοιες εγγραφές, η οποία είναι και η τιμή της σταθεράς.

Η τιμή της σταθεράς MaxKeys, η οποία ορίζει το μέγιστο αριθμό κλειδιών που χωράνε σε μία σελίδα του Β-Δέντρου, θα εξαρτάται από το μέγεθος των κλειδιών, των δεικτών προς τις σελίδες της λίστας που αντιστοιχούν σε αυτά, αλλά και των δεικτών προς τους άλλους κόμβους του δέντρου, αφού όλα αυτά συνυπάρχουν σε μία σελίδα του Β-Δέντρου. Τα κλειδιά του Β-Δέντρου υλοποιούνται όπως έχουμε δει με τη δομή KeyFieldType για την οποία ισχύει

$$sizeof(KeyFieldType) = sizeof(items) + sizeof(length) + sizeof(tid) =$$
$$= KeyFieldMax \cdot sizeof(short) + sizeof(\text{int}) + sizeof(long) = 58Bytes$$

όπου έχουμε θεωρήσει ότι το μέγιστο μήκος μίας δοσοληψίας (KeyFieldMax) είναι τα 25 αντικείμενα. Αντίστοιχα, οι δείκτες προς τις σελίδες των λιστών θα πρέπει να είναι τύπου PagePtr και έτσι το μέγεθός τους θα είναι $sizeof(PagePtr) = \max(sizeof(long), sizeof(PageData*)) = \max(4,4) = 4Bytes$.

Επιπλέον, οι δείκτες προς τους άλλους κόμβους του Β-Δέντρου είναι τύπου long όπως έχουμε ήδη δει και άρα το μέγεθος του καθενός είναι ίσο με 4Bytes. Έτσι, θεωρώντας ότι η κάθε σελίδα του Β-Δέντρου έχει μέγεθος 4Kbytes και περιλαμβάνει i κλειδιά (δηλαδή i+1 δείκτες προς άλλες σελίδες του δέντρου), έχουμε τελικά $(58+4) \cdot i + 4 \cdot (i+1) = 4.096 \Rightarrow i = 62$. Έτσι $MaxKeys = i = 62$ ενώ $MinKeys = MaxKeys / 2 = 31$.

Οι υπόλοιπες σταθερές απεικονίζουν όπως είπαμε και πριν κάποια από τα χαρακτηριστικά της βάσης (πλήθος δοσοληψιών, μέγεθος λεξικού, κ.λ.π.) και έτσι η εύρεση της τιμής τους για τα σύνολα δεδομένων που χρησιμοποιούμε δεν αναλύεται εδώ αφού είναι μάλλον τετριμμένη.

Τέλος, σημειώνουμε ότι τα πειράματα έγιναν με IFILE_THRESH = 1, δηλαδή χρησιμοποιήσαμε τα Β-Δέντρα όλων των λιστών για την αποτίμηση των ερωτημάτων, εκτός από τις περιπτώσεις που κάποια λίστα είχε μόνο μία σελίδα, οπότε η χρήση του Β-Δέντρου δεν είχε νόημα.

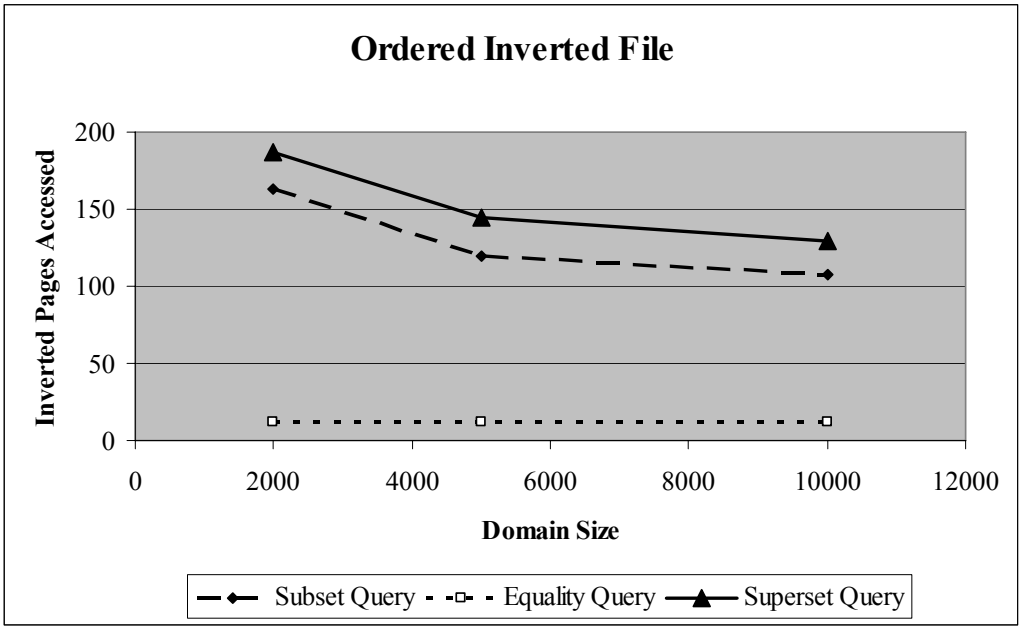
5.2 Αποτελέσματα

5.2.1 Απόδοση Ordered Inverted File

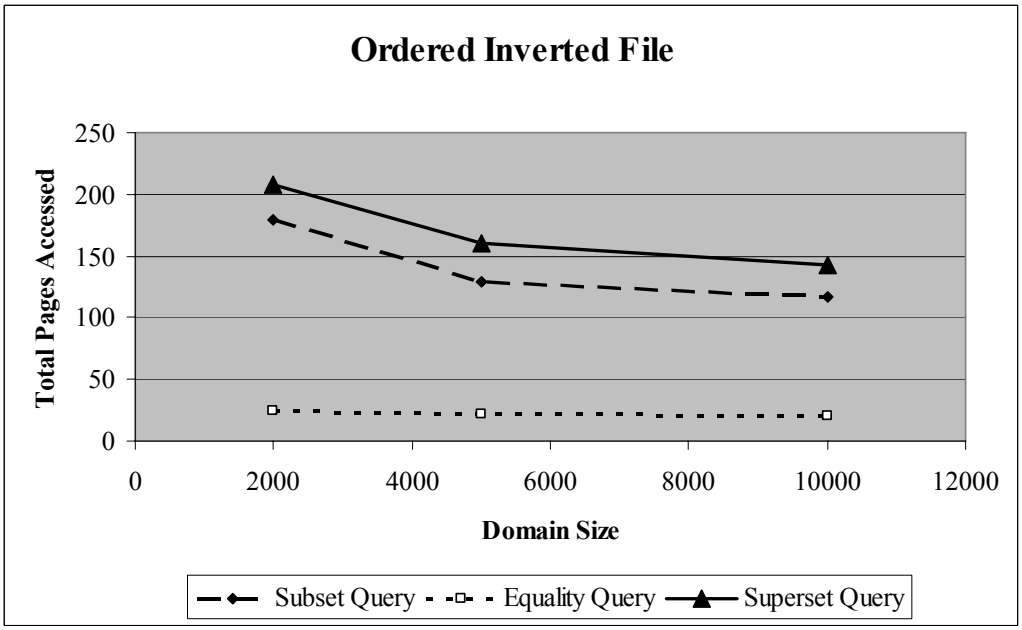
Εξετάζουμε αρχικά την απόδοση του ordered inverted file, σαν πλήθος σελίδων που προσπελαύνονται κατά την αποτίμηση των ερωτημάτων, και το πώς αυτή επηρεάζεται από τα χαρακτηριστικά της βάσης και των ερωτημάτων. Συγκεκριμένα, εξετάζουμε τον τρόπο με τον οποίο μεταβάλλεται η απόδοση του ευρετηρίου καθώς αυξάνεται το μέγεθος του λεξιλογίου (domain), το μέγεθος της βάσης, το μήκος των ερωτημάτων και η διασπορά των δεδομένων, δηλαδή το Z, αφού όπως είπαμε τα δεδομένα μας στα πειράματα αυτά ακολουθούν zipfian κατανομή. Σε κάθε περίπτωση εξετάζουμε το πλήθος των σελίδων των ανεστραμμένων λιστών που προσπελαύνονται, καθώς και τον αριθμό των συνολικών σελίδων.

5.2.1.1 Μεταβολή μεγέθους λεξιλογίου (domain)

Στα σχήματα 5.1. και 5.2. παρουσιάζεται η μεταβολή της απόδοσης του ευρετηρίου κατά τη μεταβολή του μεγέθους του λεξιλογίου της βάσης, από 2.000 αντικείμενα έως 10.000 αντικείμενα, για τους τρεις βασικούς τύπους ερωτημάτων που μας απασχολούν.



Σχήμα 5.1.: Επίδραση μεγέθους του λεξιλογίου στις σελίδες των λιστών που προσπελαύνονται



Σχήμα 5.2.: Επίδραση μεγέθους του λεξιλογίου στις συνολικές σελίδες που προσπελαύνονται

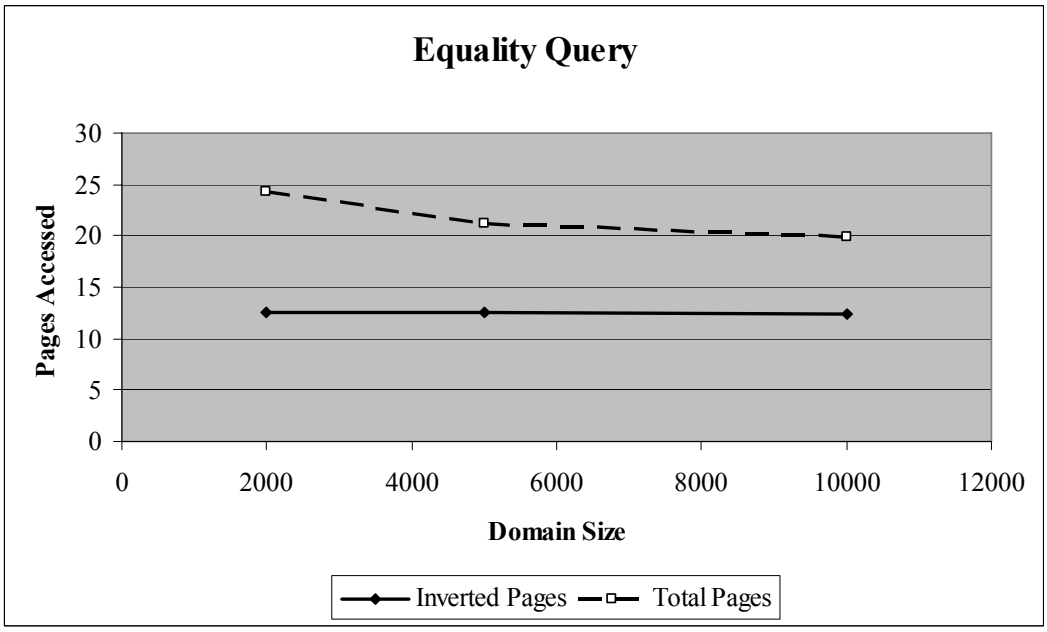
Παρατηρούμε αρχικά ότι η αποτίμηση των superset queries απαιτεί την προσπέλαση μεγαλύτερου πλήθους σελίδων από αυτή των άλλων δύο τύπων ερωτημάτων, αφού όπως

έχουμε ήδη πει τα superset queries έχουν περισσότερες από μία περιοχές πιθανών απαντήσεων σε κάθε λίστα. Αντίστοιχα, η αποτίμηση των equality queries απαιτεί την προσπέλαση σαφώς λιγότερων σελίδων, αφού οι περιοχές πιθανών απαντήσεων αυτού του ερωτήματος είναι πολύ μικρές, και περιορίζονται στις σελίδες όπου μπορεί να υπάρχουν δοσοληψίες τα περιεχόμενα των οποίων ταυτίζονται με το query set.

Βλέπουμε επίσης ότι το πλήθος των σελίδων στο σχήμα 5.2. είναι μεγαλύτερο, πράγμα αναμενόμενο αφού εκεί συνυπολογίζουμε και τις σελίδες που προσπελαύνονται από τα B-Δέντρα και τον ενδιάμεσο πίνακα. Παρ' όλα αυτά, τα δύο διαγράμματα έχουν την ίδια μορφή, πράγμα το οποίο σημαίνει ότι το πλήθος των σελίδων των B-Δέντρων και του ενδιάμεσου πίνακα που προσπελαύνονται κατά την αποτίμηση των ερωτημάτων δε μεταβάλλεται σημαντικά με τις μεταβολές του μεγέθους του λεξιλογίου.

Τέλος, μπορούμε να παρατηρήσουμε ότι εμφανίζεται μία μείωση των σελίδων που προσπελαύνονται κατά την αποτίμηση των subset και superset queries καθώς αυξάνεται το μέγεθος του λεξιλογίου. Αυτό συμβαίνει γιατί η αύξηση του μεγέθους του λεξιλογίου χωρίς ανάλογη αύξηση του μεγέθους της βάσης έχει σαν αποτέλεσμα το ίδιο πλήθος δοσοληψιών να είναι διασπαρμένο σε περισσότερες λίστες, με αποτέλεσμα η κάθε λίστα να είναι μικρότερη. Καθώς οι σελίδες που προσπελαύνονται κατά την αποτίμηση των δύο αυτών τύπων ερωτημάτων είναι πολύ περισσότερες από αυτές που προσπελαύνονται κατά την αποτίμηση των equality queries, είναι αναμενόμενο η διαφορά αυτή στο μήκος των λιστών να γίνεται περισσότερο εμφανής εκεί.

Στο σχήμα 5.3. παρουσιάζονται τα αποτελέσματα που έχει η παραπάνω μεταβολή μόνο στα equality queries.

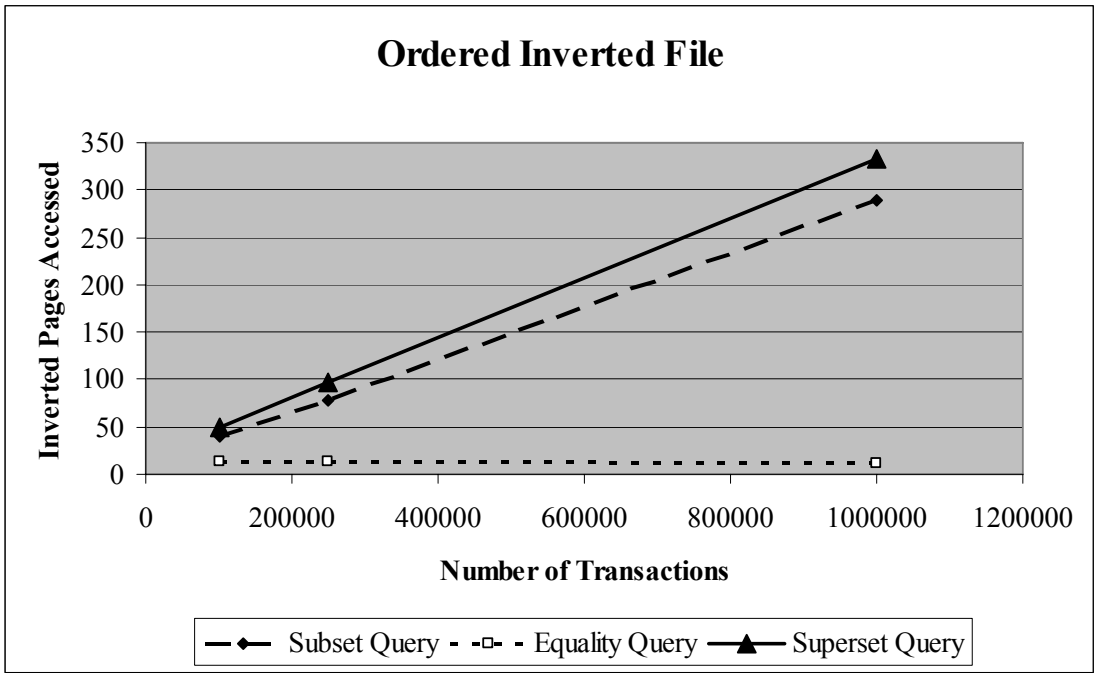


Σχήμα 5.3.: Επίδραση μεγέθους του λεξιλογίου στην απόδοση των equality queries

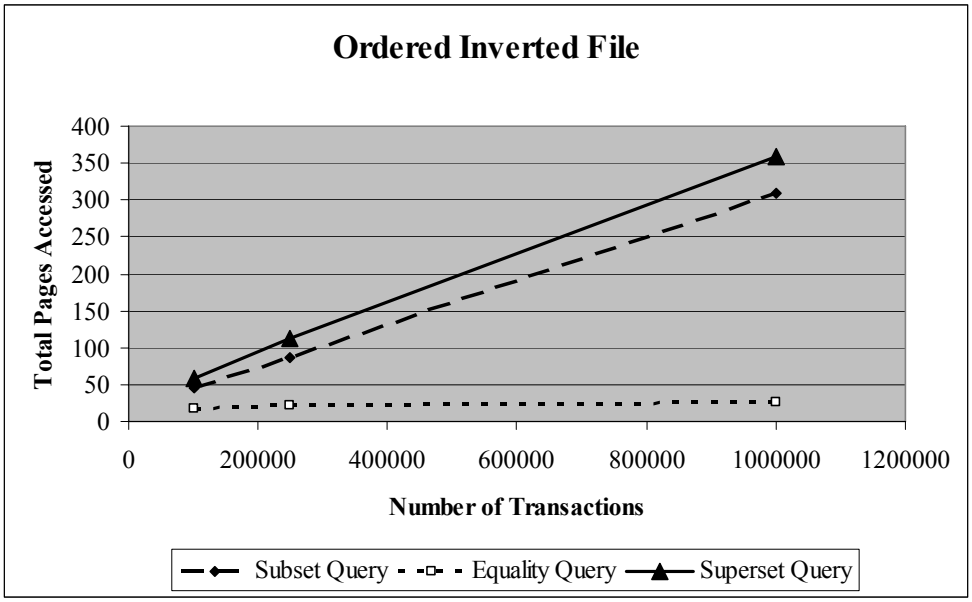
Βλέπουμε ότι οι συνολικές σελίδες που προσπελαύνονται επηρεάζονται από την αλλαγή του μεγέθους του λεξιλογίου, σε αντίθεση με τις σελίδες των inverted lists οι οποίες παραμένουν σχεδόν σταθερές. Αυτό σημαίνει ότι το μέγεθος των τμημάτων των λιστών που προσπελαύνονται κατά την αποτίμηση του ερωτήματος αυτού είναι ανεξάρτητο από το μέγεθος του λεξιλογίου και άρα των λιστών. Αντίθετα, αυτό που επηρεάζεται είναι το πλήθος των σελίδων που προσπελαύνονται στα Β-Δέντρα, αφού μικρότερες λίστες έχουν σαν αποτέλεσμα μικρότερα και άρα πιο ρηχά Β-Δέντρα, πράγμα που σημαίνει μικρότερα μονοπάτια που πρέπει να διανύσουμε κατά την αναζήτηση κάποιου κλειδιού.

5.2.1.2 Μεταβολή μεγέθους βάσης

Στα σχήματα 5.4. και 5.5. παρουσιάζουμε την επίδραση που έχει στην αποτίμηση των ερωτημάτων το μέγεθος της βάσης, δηλαδή το πλήθος των δοσοληψιών που αυτή περιέχει.



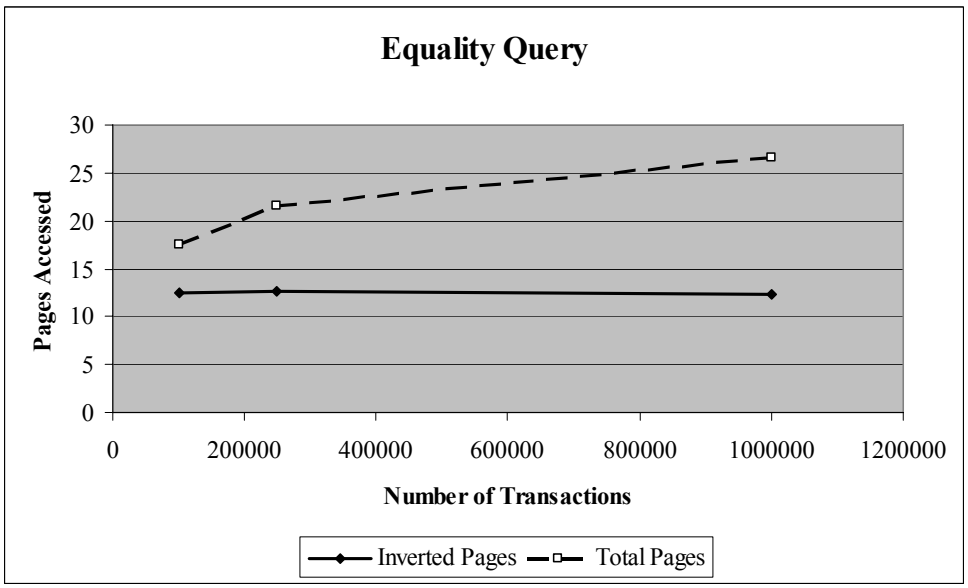
Σχήμα 5.4.: Επίδραση μεγέθους της βάσης στις σελίδες των λιστών που προσπελαύνονται



Σχήμα 5.5.: Επίδραση μεγέθους της βάσης στις συνολικές σελίδες που προσπελαύνονται

Παρατηρούμε αρχικά ότι η σχετική απόδοση κατά την αποτίμηση των τριών τύπων ερωτημάτων είναι ίδια με αυτή που είδαμε προηγουμένως, με τις διαφορά ανάμεσά τους να μεγαλώνει καθώς μεγαλώνει και η βάση. Στην περίπτωση αυτή, το μέγεθος της βάσης αυξάνεται χωρίς ανάλογη αύξηση του μεγέθους του λεξιλογίου και έτσι έχουμε περισσότερες δοσοληψίες οι οποίες τοποθετούνται στον ίδιο αριθμό λιστών, με αποτέλεσμα η καθεμία από αυτές τις λίστες να είναι μεγαλύτερη. Έτσι, το αποτέλεσμα της μεταβολής αυτής στην απόδοση του ευρετηρίου μας είναι να αυξάνεται γραμμικά το πλήθος των σελίδων που προσπελαύνονται κατά την αποτίμηση subset και superset queries.

Παραθέτουμε επιπλέον στο σχήμα 5.6. για άλλη μια φορά την επίδραση της μεταβολής αυτής αποκλειστικά στα equality queries.

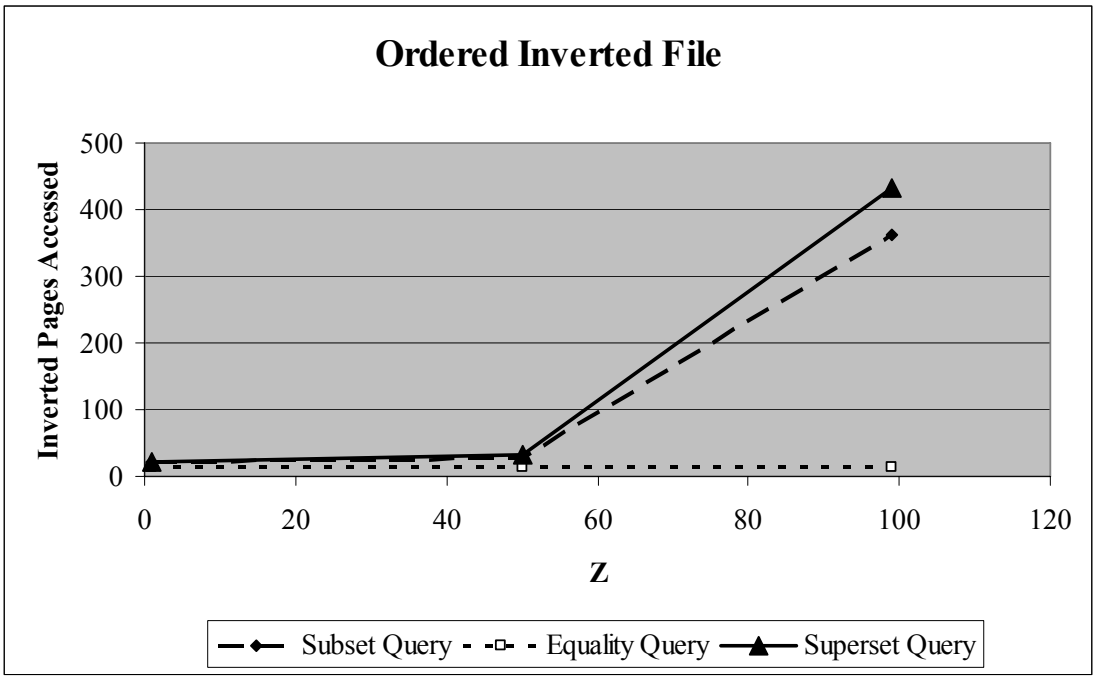


Σχήμα 5.6.: Επίδραση μεγέθους της βάσης στην απόδοση των equality queries

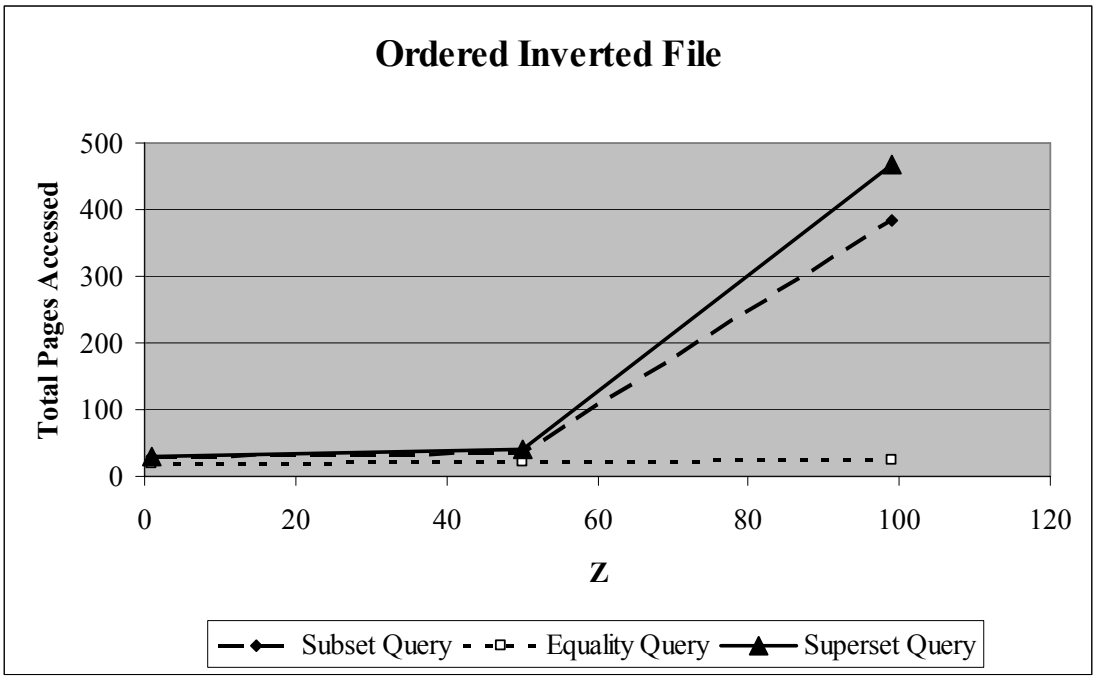
Παρατηρούμε ότι και αυτή η μεταβολή επιδρά στο συνολικό πλήθος των σελίδων που προσπελαύνονται κατά την αποτίμηση των equality queries, χωρίς να επηρεάζει το πλήθος των σελίδων των λιστών που προσπελαύνονται. Αυτό μπορούμε να πούμε ότι είναι αναμενόμενο, αφού τελικά η αύξηση του μεγέθους της βάσης ανάγεται όπως και η αύξηση του μεγέθους του λεξιλογίου σε μεταβολή του μεγέθους των ανεστραμμένων λιστών, και έτσι περιμένουμε η συμπεριφορά του ευρετηρίου μας και στις δύο αυτές περιπτώσεις να είναι αντίστοιχη.

5.2.1.3 Μεταβολή διασποράς αντικειμένων

Στη συνέχεια παραθέτουμε διαγράμματα στα οποία εμφανίζεται η απόδοση του ordered inverted file συναρτήσει της μεταβολής του Z, δηλαδή της διασποράς των αντικειμένων στις δοσοληψίες. Θυμίζουμε εδώ ότι όσο μεγαλύτερο είναι το Z, τόσο πιο συχνά εμφανίζονται τα πρώτα αντικείμενα του λεξιλογίου μας σε σχέση μα τα τελευταία, ενώ για πολύ μικρές τιμές του Z όλα τα αντικείμενα έχουν σχεδόν την ίδια πιθανότητα εμφάνισης.



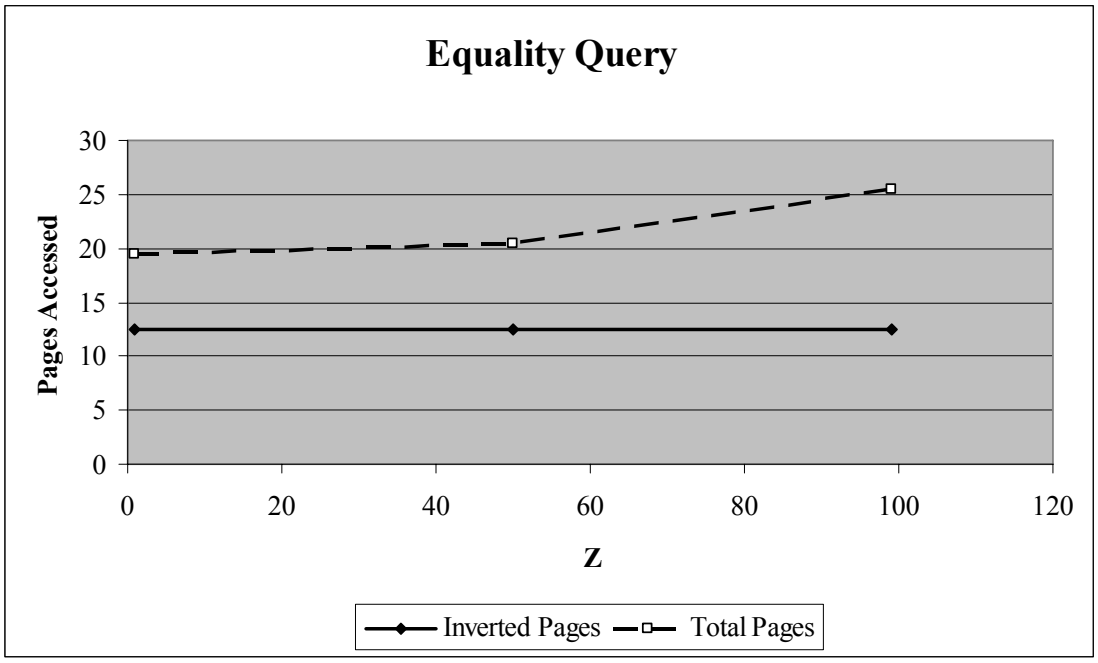
Σχήμα 5.7.: Επίδραση της διασποράς των αντικειμένων στις σελίδες των λιστών που προσπελαύνονται



Σχήμα 5.8.: Επίδραση της διασποράς των αντικειμένων στις συνολικές σελίδες που προσπελαύνονται

Βλέπουμε ότι το πλήθος των σελίδων που προσπελαύνονται κατά την αποτίμηση subset και superset queries αυξάνει σχεδόν εκθετικά για μεγάλες του Z. Η απότομη αυτή αύξηση μπορεί να εξηγηθεί αν σκεφτούμε ότι ένα μεγάλο Z έχει σαν αποτέλεσμα αφ' ενός τα πρώτα αντικείμενα του λεξιλογίου να εμφανίζονται πολύ συχνότερα στη βάση από τα υπόλοιπα και έτσι οι λίστες τους να είναι σημαντικά μεγαλύτερες και αφ' ετέρου τα αντικείμενα αυτά να εμφανίζονται συχνότερα και στα ερωτήματά μας, τα οποία όπως προείπαμε έχουν δημιουργηθεί από υπάρχουσες δοσοληψίες της βάσης. Έτσι, καθώς μεγαλώνει το Z αυξάνεται και η πιθανότητα να χρειαστεί να επεξεργαστούμε σημαντικά μεγαλύτερες λίστες για να απαντήσουμε στο εκάστοτε ερώτημα.

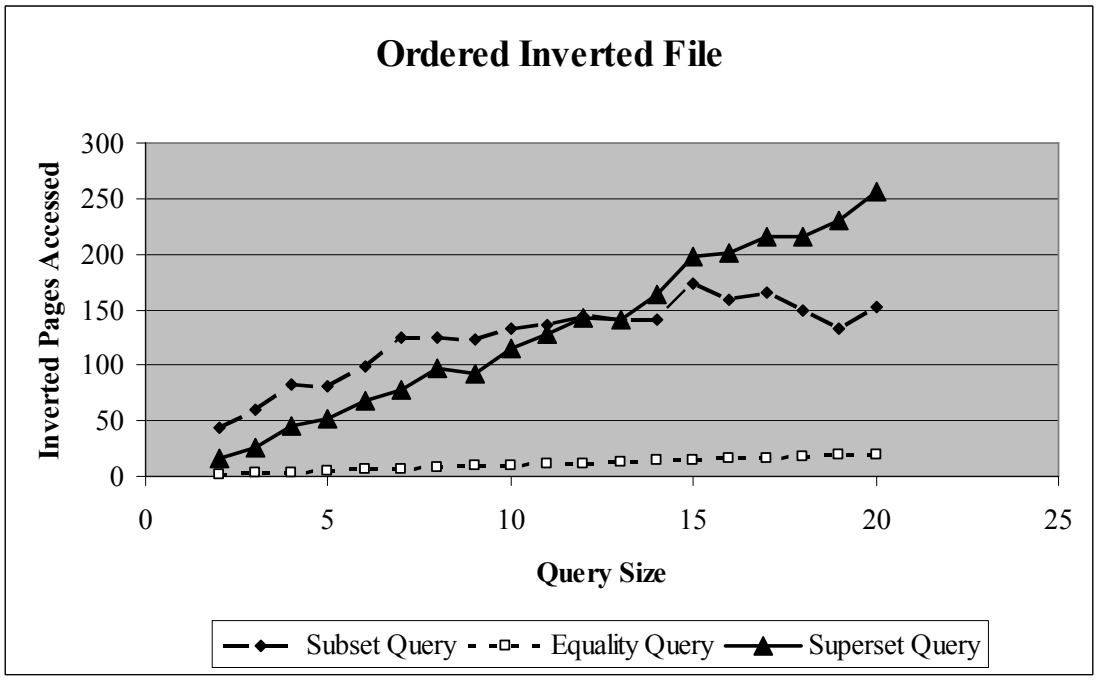
Για άλλη μια φορά η μεταβολή αυτή δεν επιδρά σημαντικά στην απόδοση των equality queries και αφήνει σχεδόν ανεπηρέαστο το πλήθος των σελίδων των ανεστραμμένων λιστών που πρέπει να προσπελάσουμε κατά την αποτίμησή τους. Αντίστοιχα, αυξάνεται το συνολικό πλήθος των σελίδων που προσπελαύνονται, αφού σύμφωνα με τα προηγούμενα μεγάλο Z θα ισοδυναμεί με αυξημένη πιθανότητα να χρειαστεί να διατρέξουμε μεγαλύτερα δέντρα. Αυτά παρουσιάζονται γραφικά στο σχήμα 5.9.



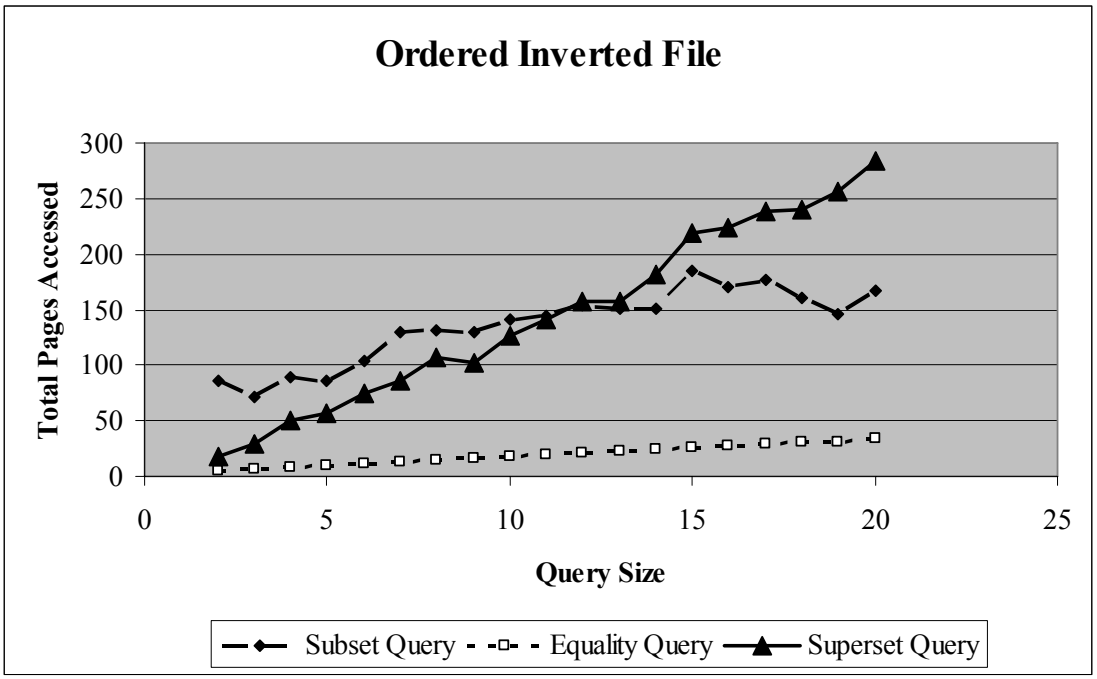
Σχήμα 5.9.: Επίδραση της διασποράς των αντικειμένων στην απόδοση των equality queries

5.2.1.4 Μεταβολή μήκους ερωτημάτων

Τέλος, εξετάζουμε με τη βοήθεια των σχημάτων 5.10 και 5.11 την επίδραση που έχει το μήκος των ερωτημάτων στην απόδοση του ευρετηρίου μας.



Σχήμα 5.10.: Επίδραση του μήκους των ερωτημάτων στις σελίδες των λιστών που προσπελαύνονται



Σχήμα 5.11.: Επίδραση του μήκους των ερωτημάτων στις συνολικές σελίδες που προσπελαύνονται

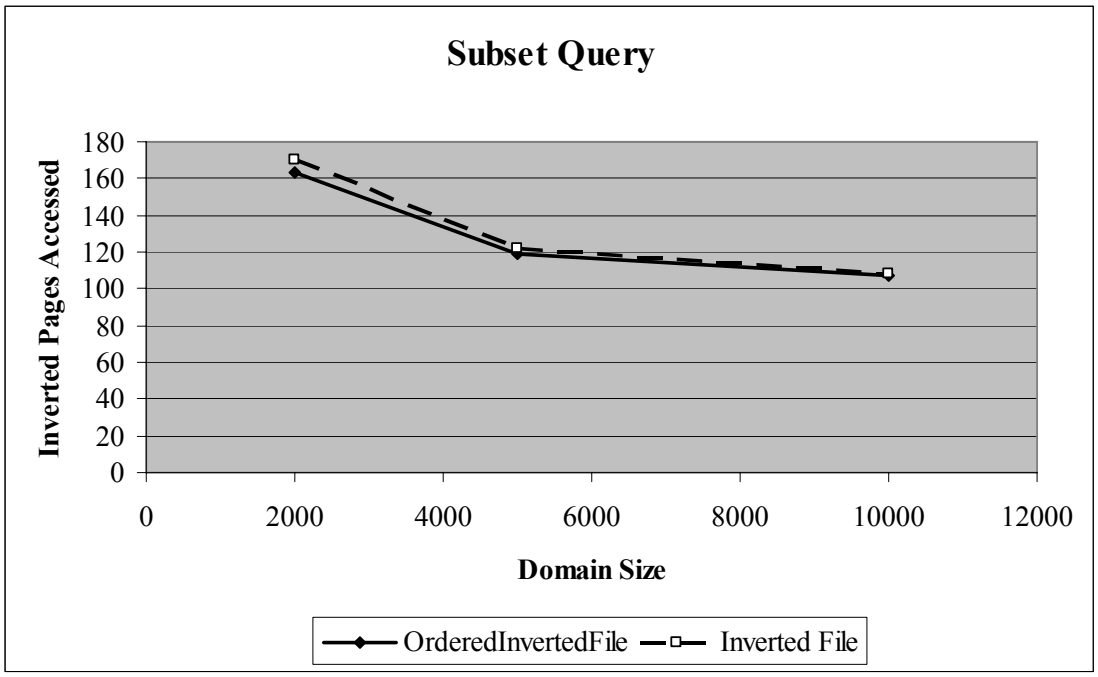
Όπως είναι αναμενόμενο, το πλήθος των σελίδων που προσπελαύνονται αυξάνει μαζί με το μήκος των ερωτημάτων, αφού έτσι αυξάνει και το πλήθος των λιστών που θα πρέπει να ερευνήσουμε για αποτελέσματα. Η μεταβολή αυτή όπως βλέπουμε επιδρά ακόμη και στα equality queries, αν και λιγότερο σε σχέση με τα άλλα ερωτήματα. Παρατηρούμε επίσης ότι για μικρά μήκη ερωτημάτων, οι αποτίμηση των superset queries απαιτεί την προσπέλαση λιγότερων σελίδων από αυτή των subset queries. Αυτό συμβαίνει γιατί στην περίπτωση των superset queries μικρά μήκη ερωτημάτων σημαίνουν όχι μόνο μικρότερες περιοχές πιθανών απαντήσεων, αλλά και λιγότερες σε κάθε λίστα, πράγμα που καθιστά την αποτίμηση αυτή αποδοτικότερη σε σχέση με αυτή των subset queries που σε κάθε περίπτωση ξεκινά από την αρχή των λιστών. Καθώς όμως το μήκος των ερωτημάτων αυξάνει το πλήθος των σελίδων που προσπελαύνονται από τα superset queries αυξάνει γρηγορότερα με αποτέλεσμα τελικά τα superset queries να απαιτούν την προσπέλαση περισσότερων σελίδων για μήκη ερωτημάτων μεγαλύτερα από 12.

5.2.2 Σύγκριση με το απλό Inverted File

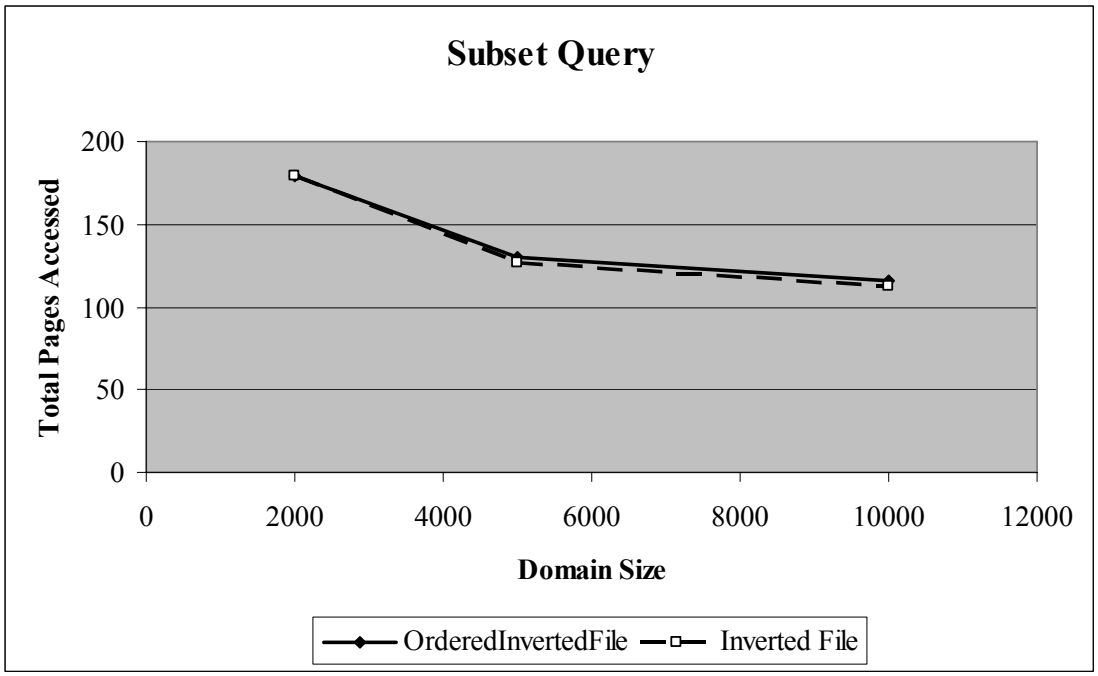
Συγκρίνουμε στη συνέχεια την απόδοση του ordered inverted file με αυτή του απλού inverted file για τα τρία βασικά είδη ερωτημάτων, και για τις μεταβολές που είδαμε παραπάνω κατά την εξέταση της συμπεριφοράς του ordered inverted file.

5.2.2.1 Subset queries

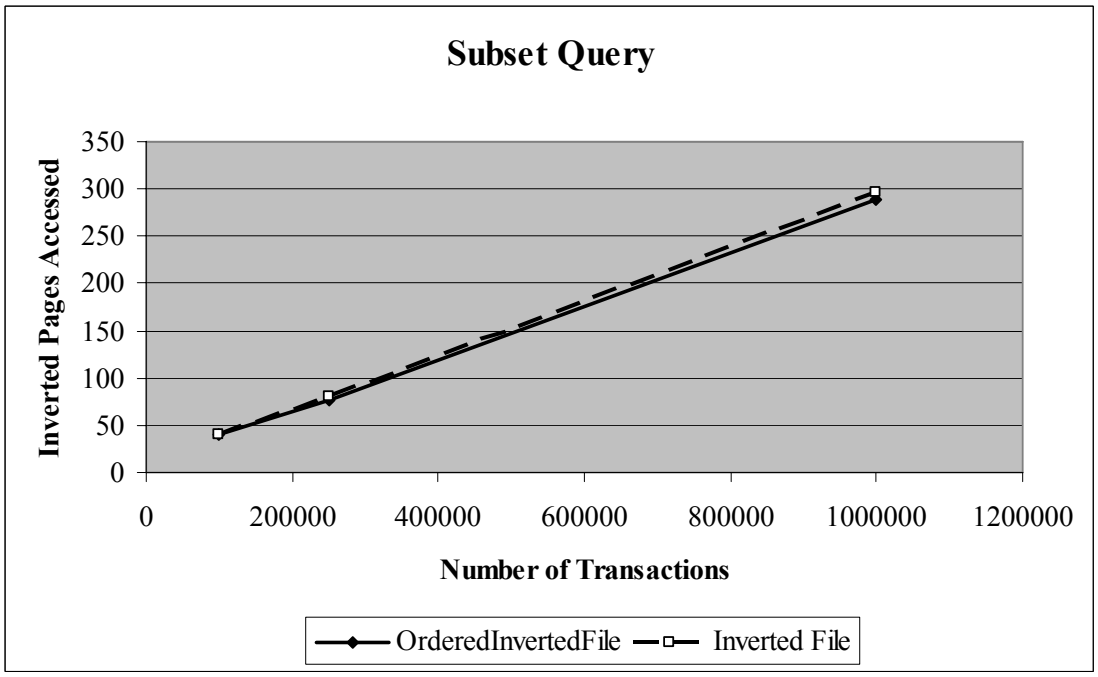
Στα σχήματα 5.12 έως 5.19 φαίνεται συγκριτικά η απόδοση των δύο ευρετηρίων καθώς μεταβάλλεται το μέγεθος του λεξιλογίου, το μέγεθος της βάσης, η τιμή του Z και το μέγεθος των ερωτημάτων. Οι συνολικές σελίδες που εμφανίζονται στα παρακάτω διαγράμματα, στην περίπτωση του ordered inverted file αναφέρονται σε σελίδες των λιστών, των B-Δέντρων και του ενδιάμεσου πίνακα, ενώ στην περίπτωση του inverted file μόνο σε σελίδες των λιστών και του ενδιάμεσου πίνακα. Η εμφάνισή τους είναι χρήσιμη καθώς οι σελίδες του ενδιάμεσου πίνακα που προσπελαύνει το απλό inverted file μπορεί να είναι περισσότερες από αυτές που προσπελαύνει το ordered inverted file καθώς οι δοσοληψίες σε αυτό δεν είναι ταξινομημένες και έτσι τα αποτελέσματα των ερωτημάτων δε βρίσκονται κατ' ανάγκην κοντά το ένα στο άλλο και μπορεί να εκτείνονται σε πολλές σελίδες του πίνακα.



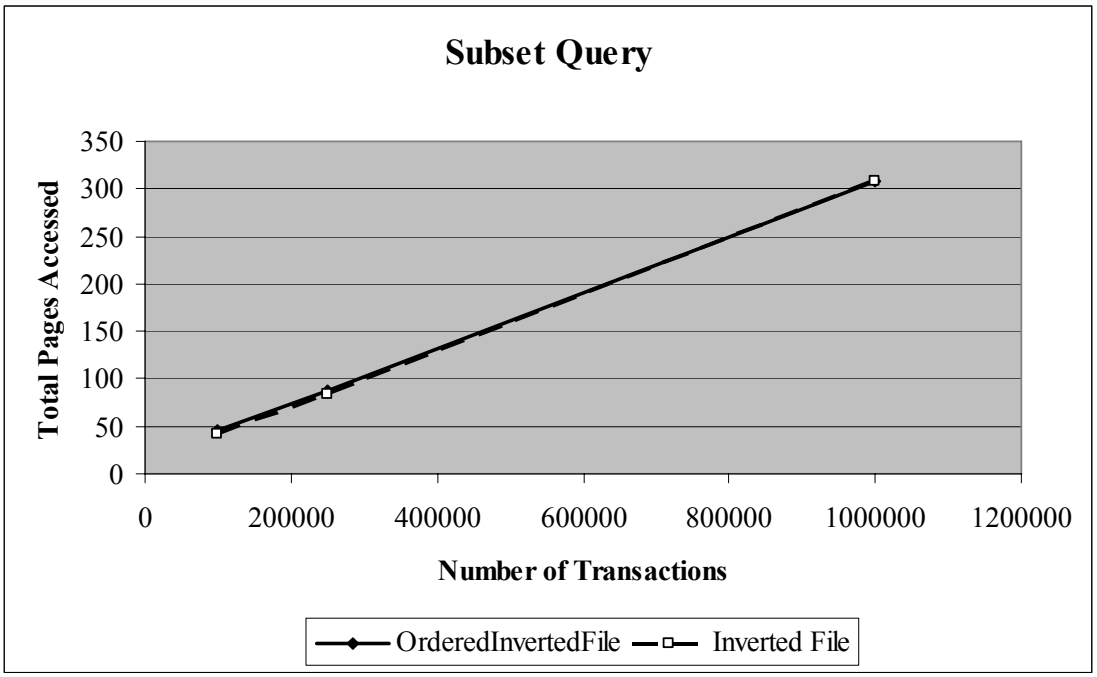
Σχήμα 5.12: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση subset queries συναρτήσει του μεγέθους του λεξιλογίου



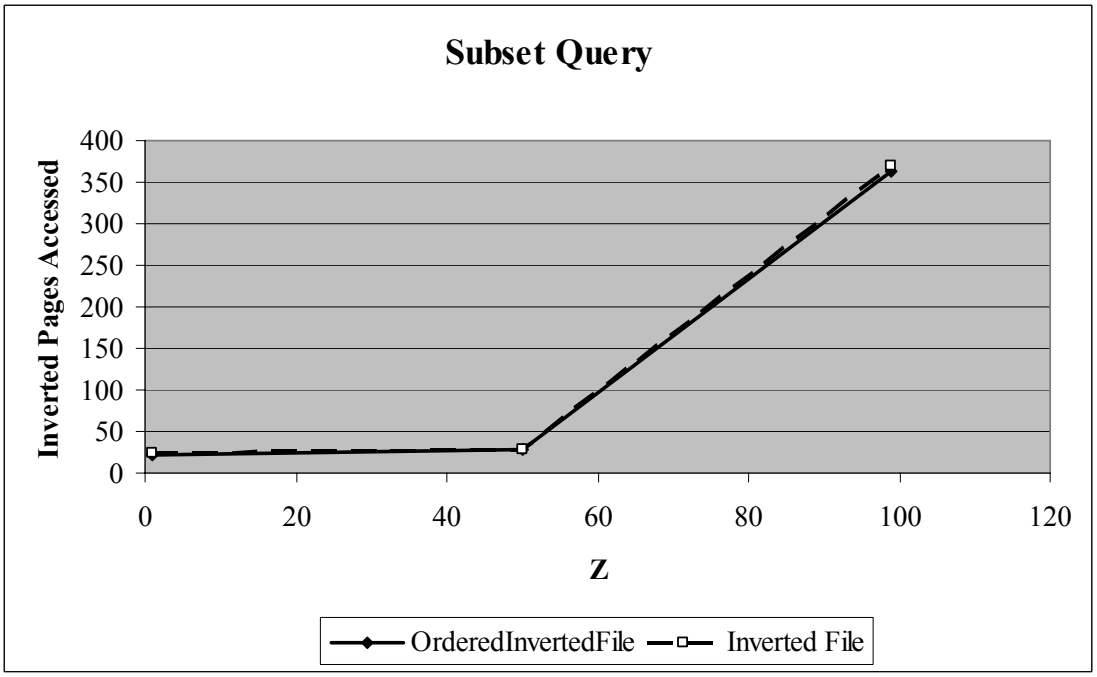
Σχήμα 5.13: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση subset queries συναρτήσει του μεγέθους του λεξιλογίου



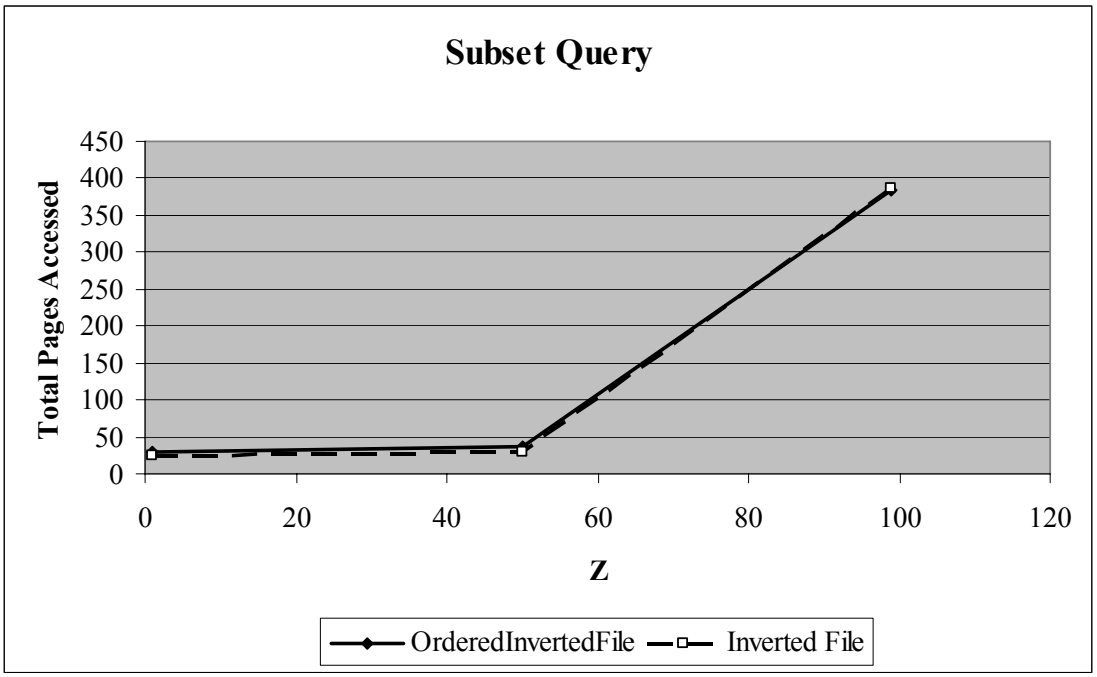
Σχήμα 5.14: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση subset queries συναρτήσει του μεγέθους της βάσης



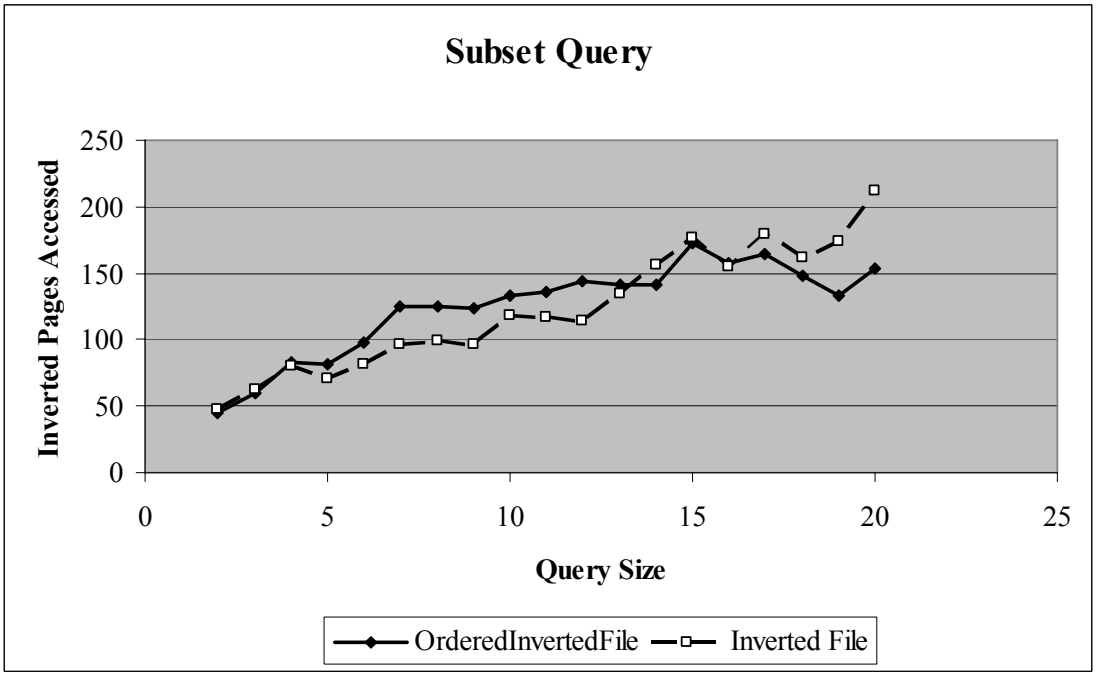
Σχήμα 5.15: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση subset queries συναρτήσει του μεγέθους της βάσης



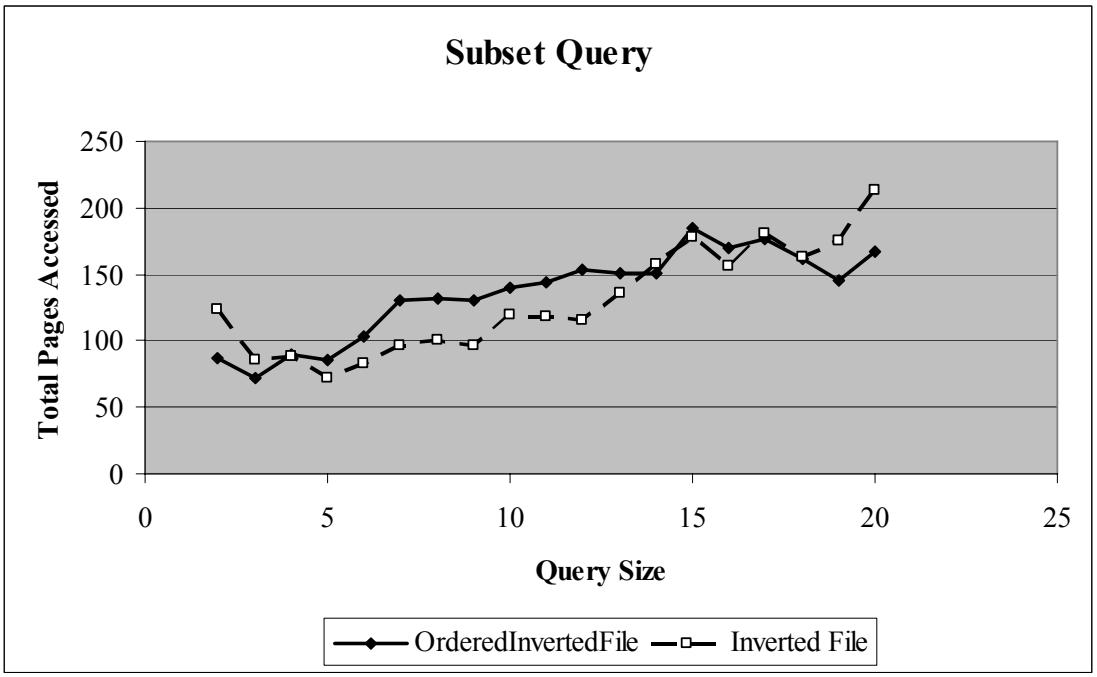
Σχήμα 5.16: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση subset queries συναρτήσει της διασποράς των αντικειμένων



Σχήμα 5.17: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση subset queries συναρτήσει της διασποράς των αντικειμένων



Σχήμα 5.18: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση subset queries συναρτήσει του μεγέθους των ερωτημάτων

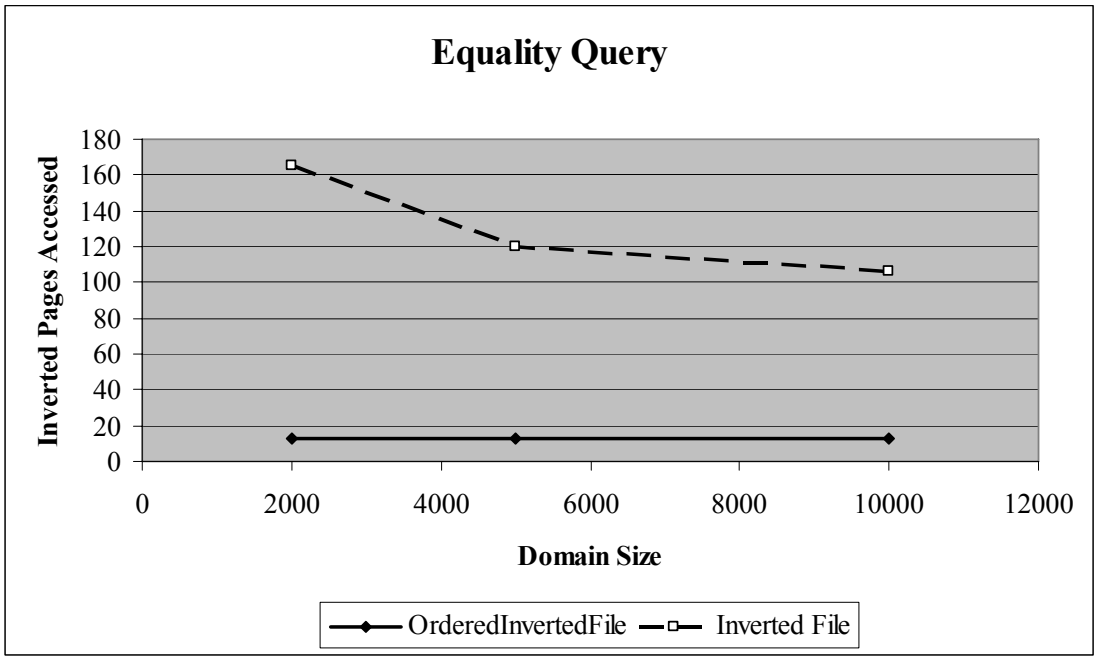


Σχήμα 5.19: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση subset queries συναρτήσει του μεγέθους των ερωτημάτων

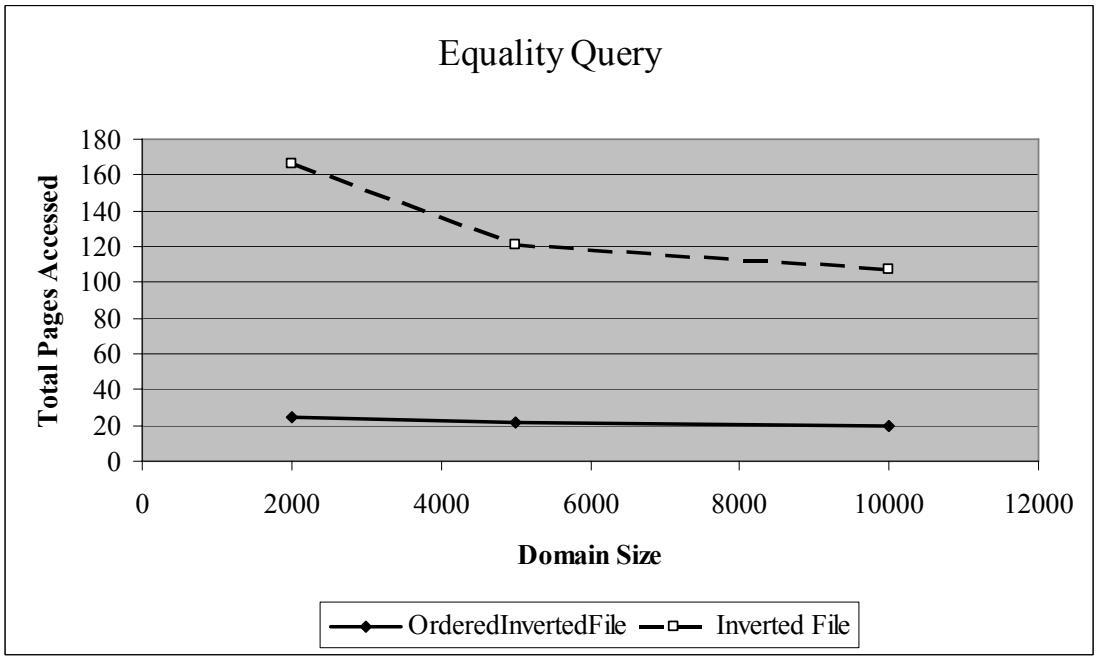
Στα παραπάνω διαγράμματα μπορούμε να δούμε ότι η απόδοση του ordered inverted file για την αποτίμηση των subset queries είναι γενικά καλύτερη από αυτή του απλού inverted file, αν και πάρα πολύ κοντά σε αυτή. Ειδικά στην περίπτωση που συγκρίνουμε τα δύο ευρετήρια συναρτήσει του μεγέθους των ερωτημάτων βλέπουμε ότι σε κάποιες περιπτώσεις το απλό inverted file μπορεί να υπερέχει ελαφρά του ordered inverted file. Η διαφορά αυτή οφείλεται πιθανότατα στο γεγονός ότι το απλό inverted file έχει κατασκευαστεί πάνω στις αταξινόμητες δοσοληψίες, με αποτέλεσμα σε κάποιες περιπτώσεις οι απαντήσεις στα ερωτήματα να βρίσκονται πιο κοντά στην αρχή των λιστών σε σχέση με το ordered inverted file, και έτσι η συνάρτηση MergeJoin που είδαμε στο προηγούμενο κεφάλαιο να προσπελαύνει τελικά λιγότερες σελίδες.

5.2.2.2 Equality queries

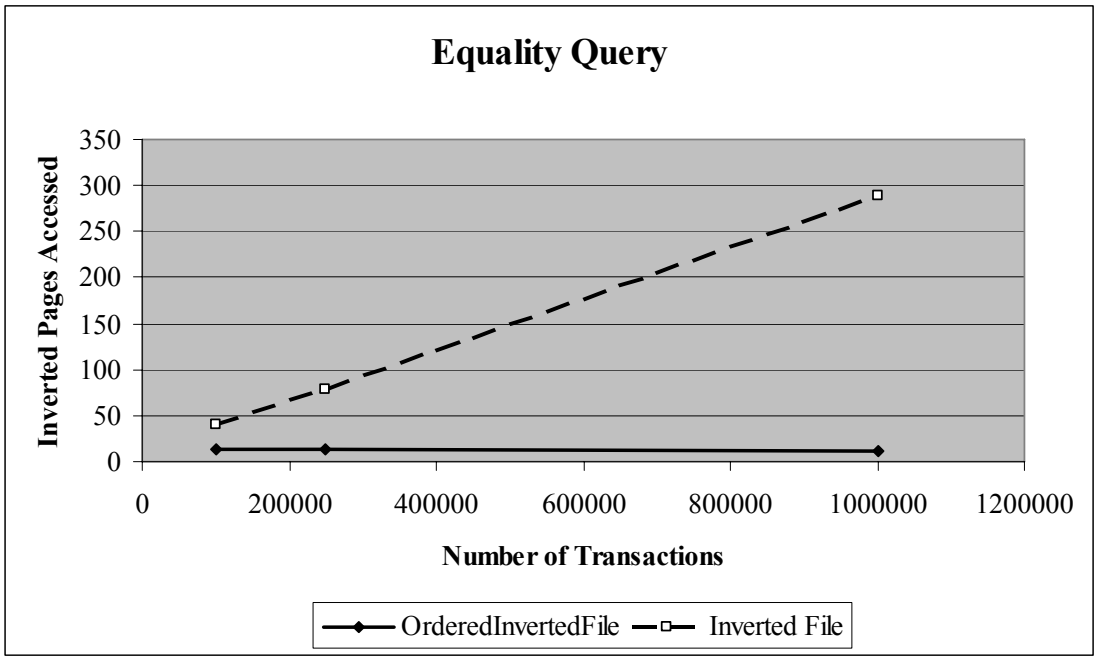
Αντίστοιχα, στα σχήματα 5.20 έως 5.27 συγκρίνονται οι επιδόσεις των δύο ευρετηρίων στην αποτίμηση των equality queries.



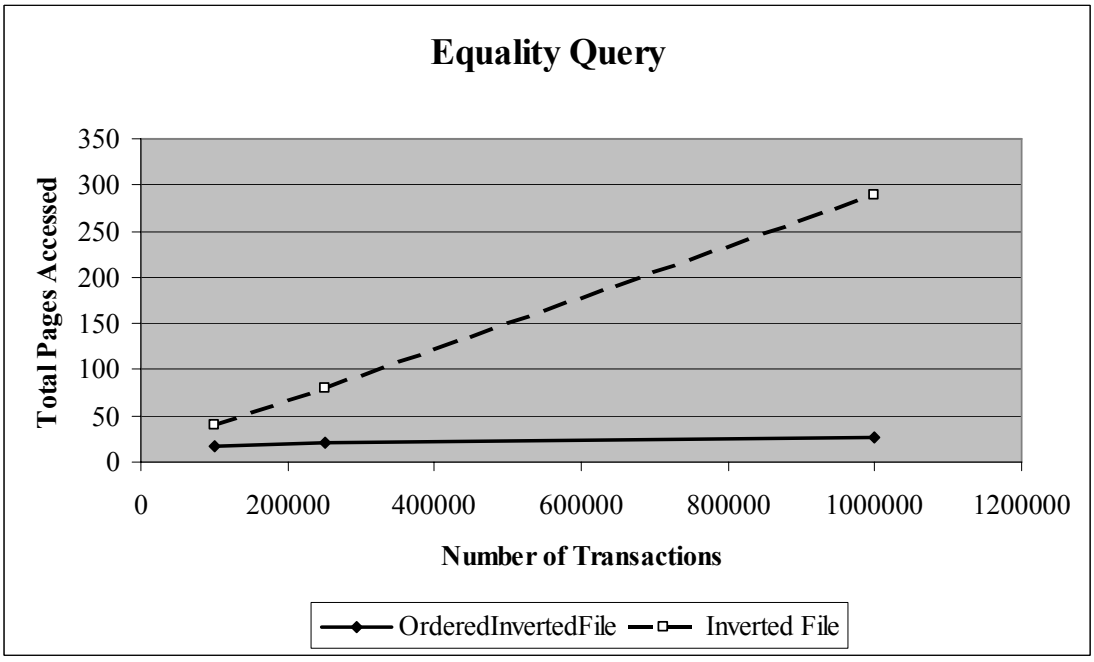
Σχήμα 5.20: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση equality queries συναρτήσει του μεγέθους του λεξιλογίου



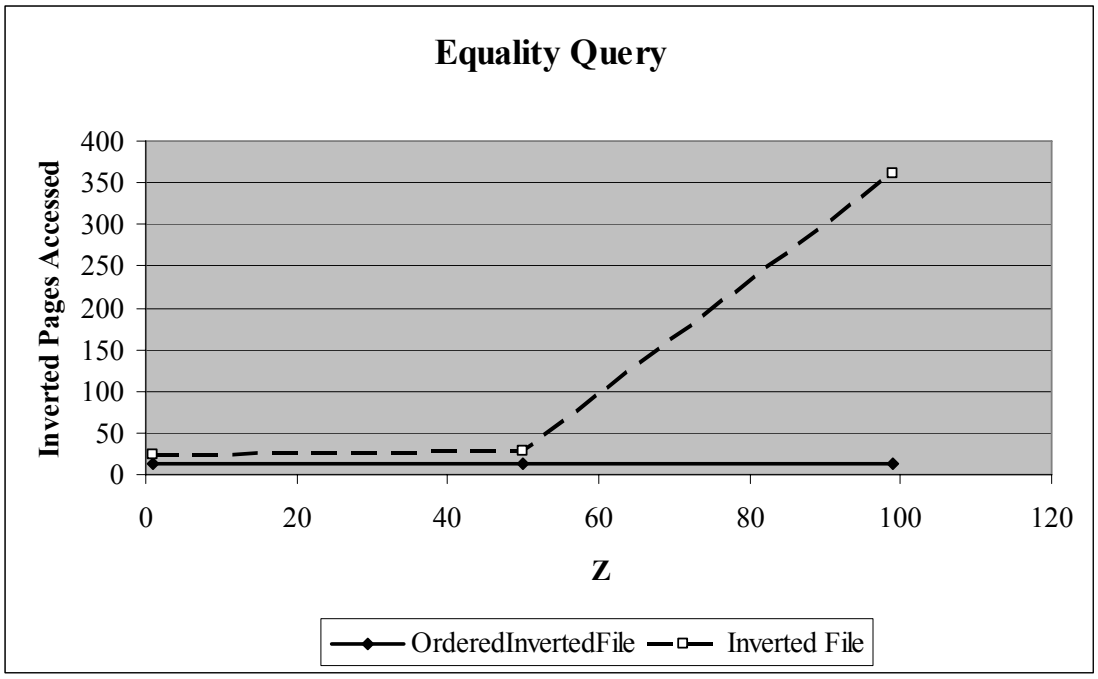
Σχήμα 5.21: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση equality queries συναρτήσει του μεγέθους του λεξιλογίου



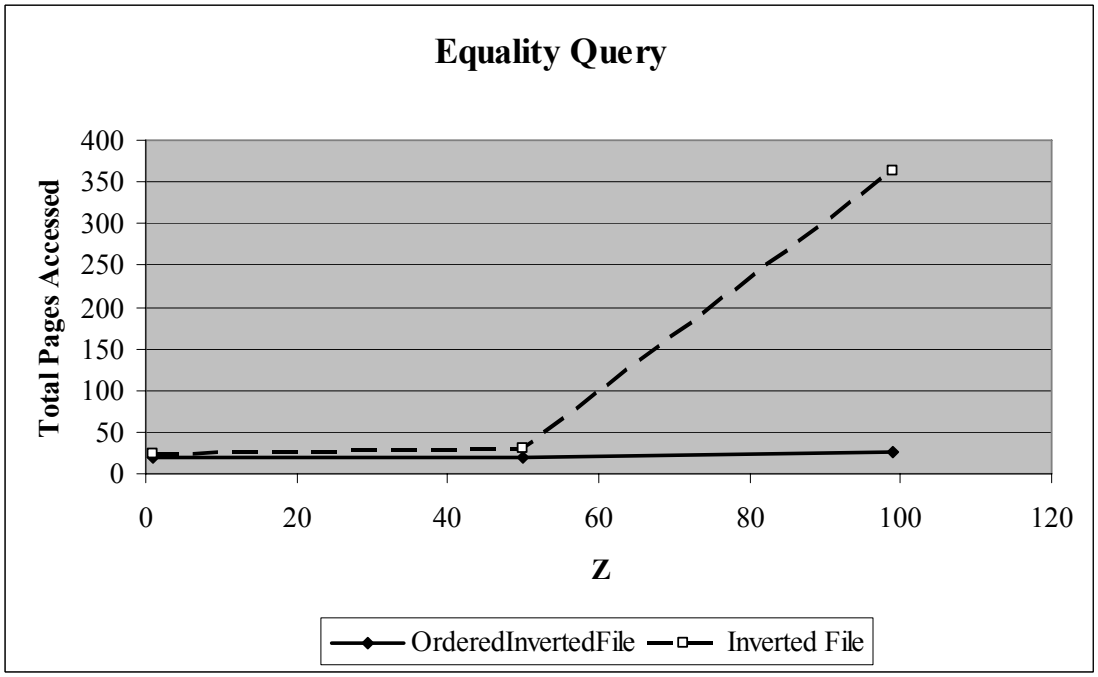
Σχήμα 5.22: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση equality queries συναρτήσει του μεγέθους της βάσης



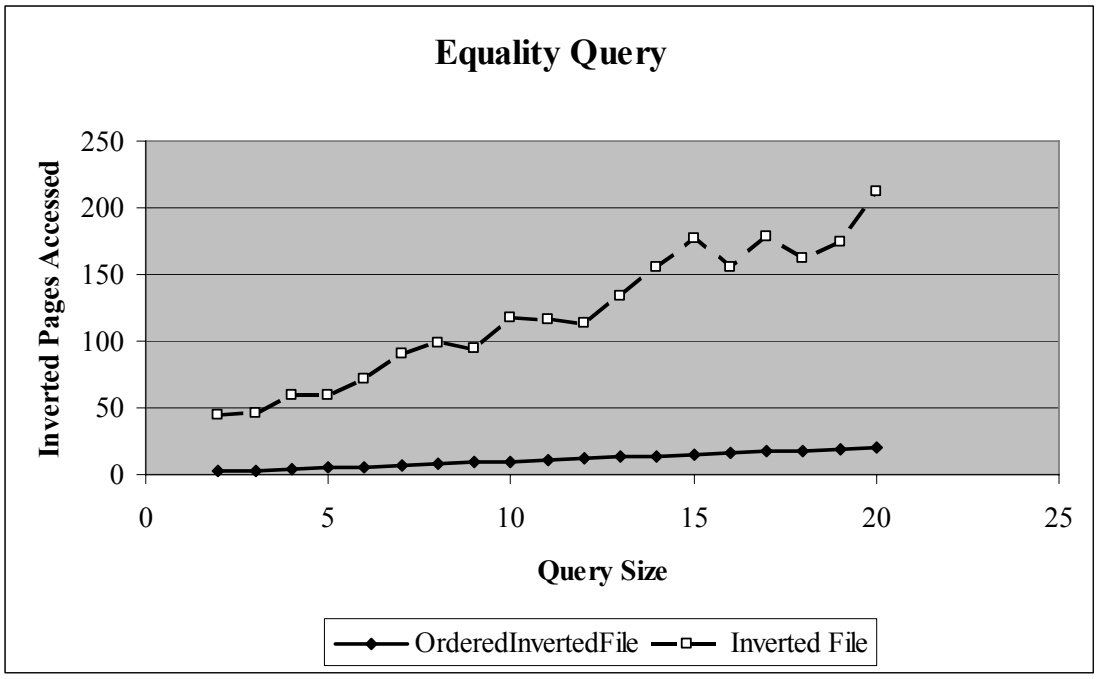
Σχήμα 5.23: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση equality queries συναρτήσει του μεγέθους της βάσης



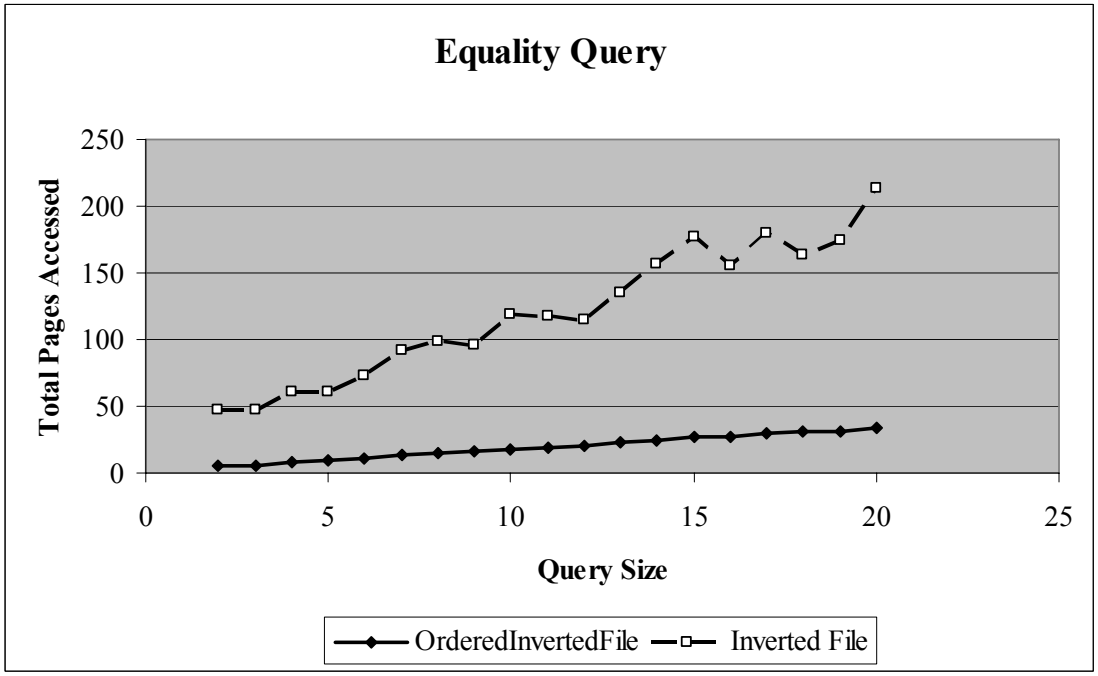
Σχήμα 5.24: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση equality queries συναρτήσει της διασποράς των αντικειμένων



Σχήμα 5.25: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση equality queries συναρτήσει της διασποράς των αντικειμένων



Σχήμα 5.26: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση equality queries συναρτήσει του μεγέθους των ερωτημάτων



Σχήμα 5.27: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση equality queries συναρτήσει του μεγέθους των ερωτημάτων

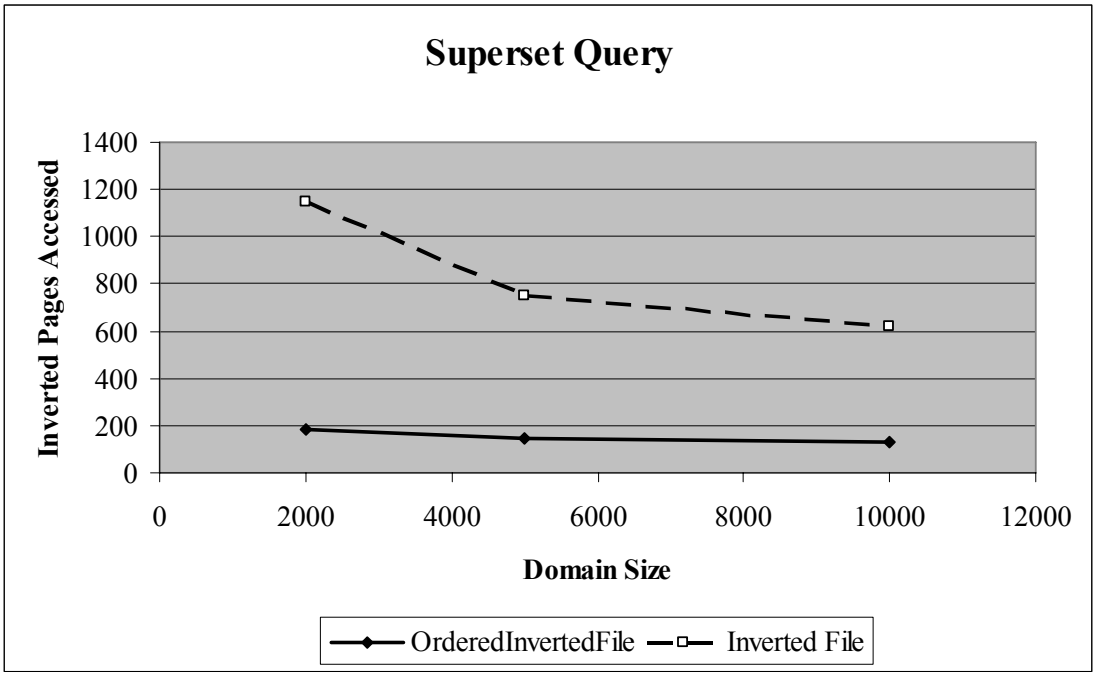
Στην περίπτωση αυτή βλέπουμε ότι το ordered inverted file υπερέχει σημαντικά του απλού inverted file, καθώς προσπελαύνει πολύ μικρά τμήματα των λιστών ενώ το inverted file τις προσπελαύνει σχεδόν ολόκληρες. Ακόμη και στις περιπτώσεις που συγκρίνουμε τη συνολική προσπέλαση σελίδων, οπότε συνυπολογίζονται και οι σελίδες των Β-Δέντρων που

προσπελαύνει το ordered inverted file, ο αριθμός των σελίδων που προσπελαύνει το ordered inverted file παραμένει έως και μία τάξη μεγέθους μικρότερος από τον αριθμό των σελίδων που προσπελαύνει το απλό inverted file.

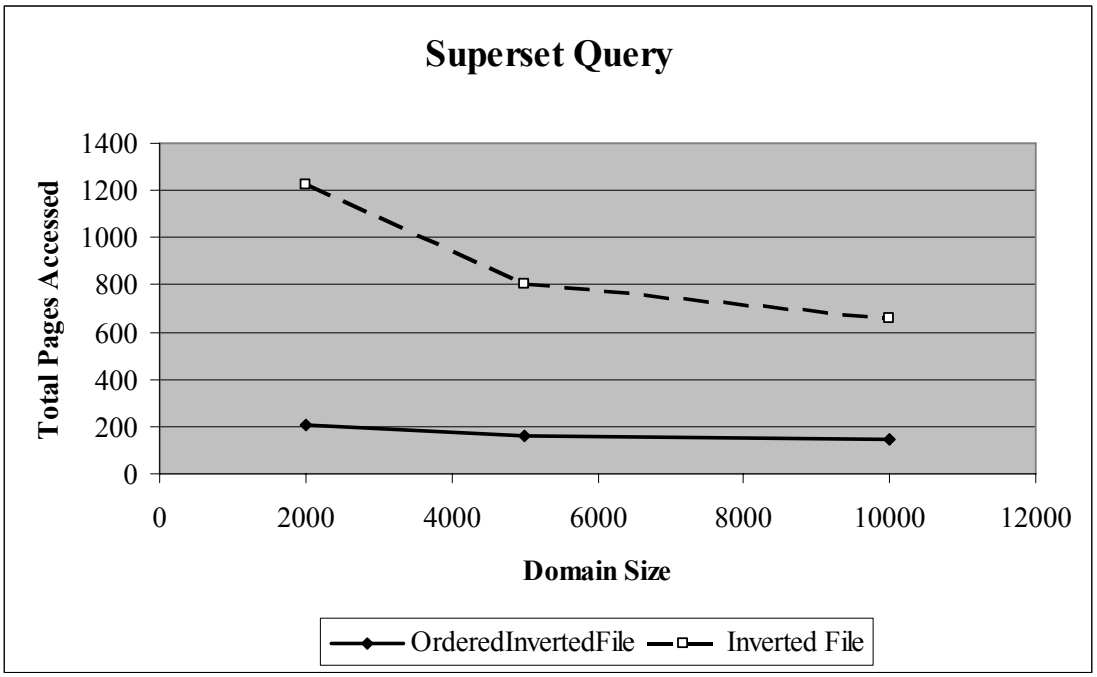
Η διαφορά ανάμεσα στα δύο ευρετήρια γίνεται εντονότερη στις περιπτώσεις που το μέγεθος των λιστών που πρέπει να προσπελαστούν αυξάνεται, δηλαδή για μικρά μεγέθη λεξικού, μεγάλα μεγέθη βάσης και ερωτημάτων και μεγάλες τιμές του Z. Αυτό συμβαίνει γιατί αν και η απόδοση του ordered inverted file παραμένει σχεδόν ανεπηρέαστη από το μέγεθος των λιστών όπως είδαμε και στην προηγούμενη παράγραφο, το ίδιο δε συμβαίνει με το απλό inverted file, η απόδοση του οποίου εξαρτάται σημαντικά από το μέγεθος των λιστών τις οποίες πρέπει να εξετάσει.

5.2.2.3 Superset queries

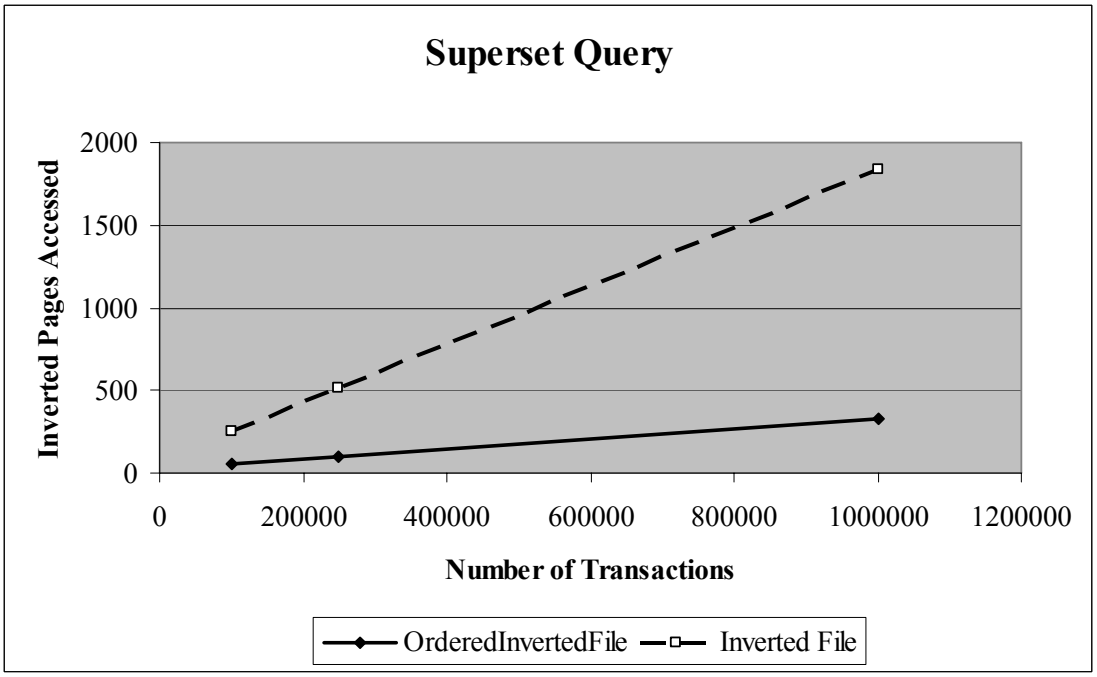
Στα σχήματα 5.28 έως 5.35 βλέπουμε αυτή τη φορά τη σύγκριση της απόδοσης των δύο ευρετηρίων κατά την αποτίμηση superset queries.



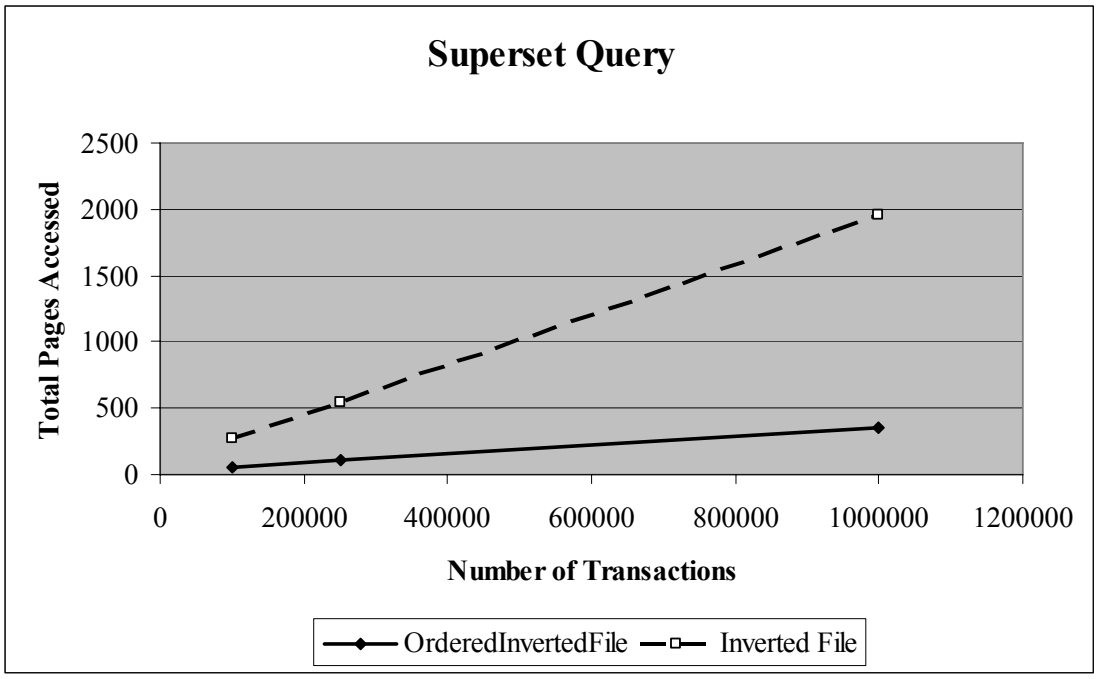
Σχήμα 5.28: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση superset queries συναρτήσει του μεγέθους του λεξιλογίου



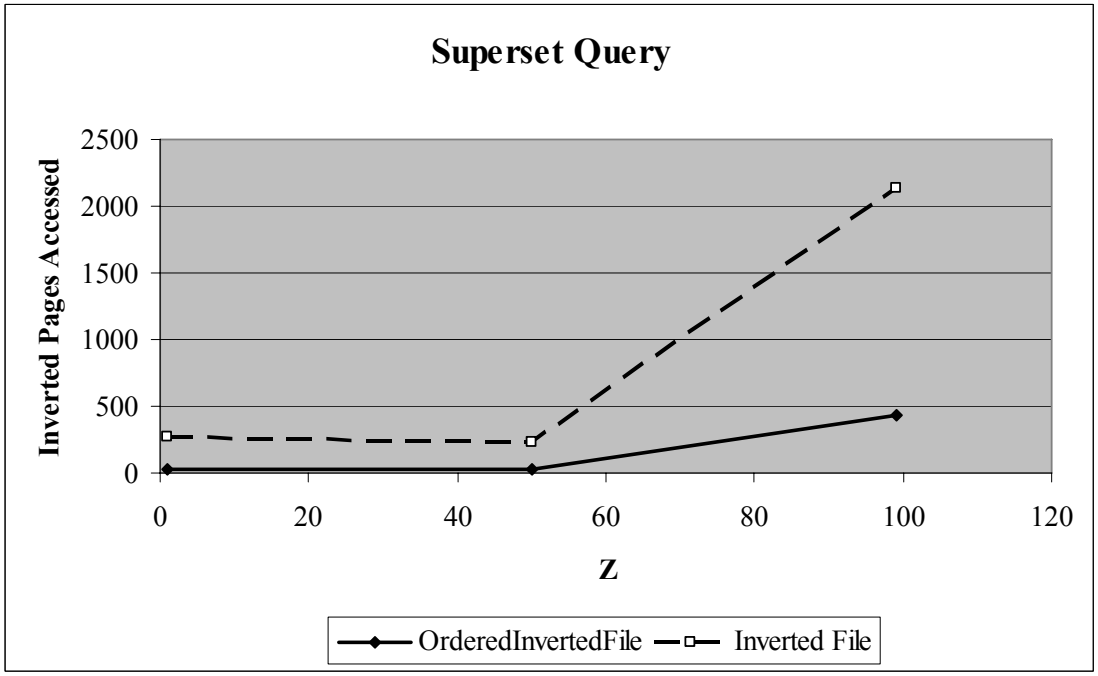
Σχήμα 5.29: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση superset queries συναρτήσει του μεγέθους του λεξιλογίου



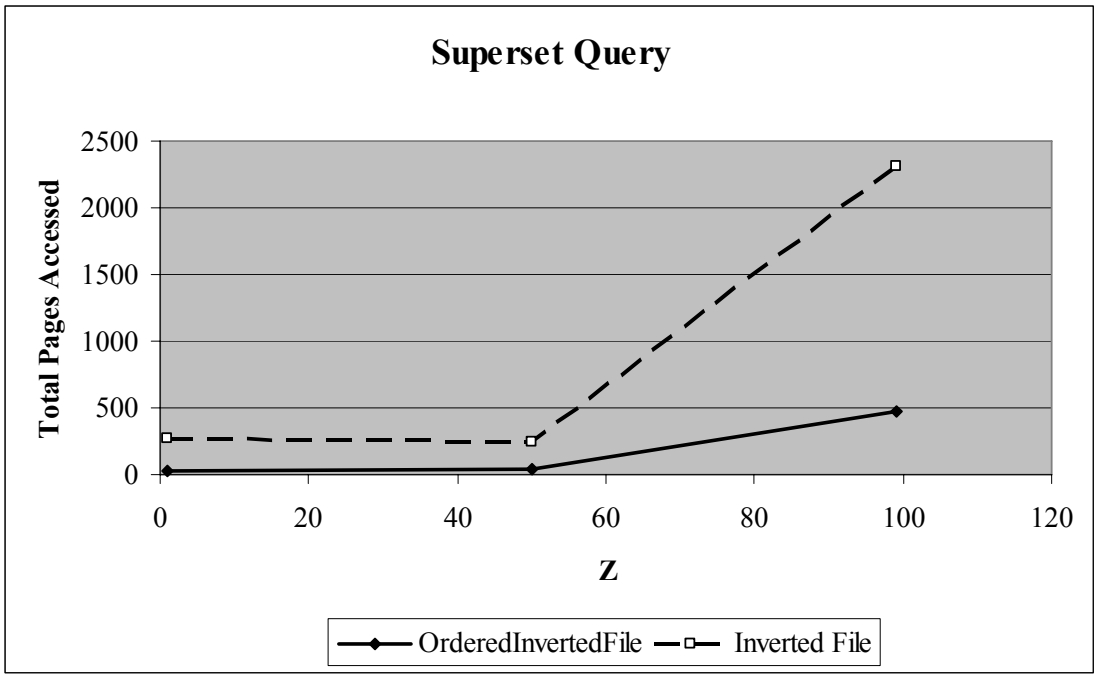
Σχήμα 5.30: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση superset queries συναρτήσει του μεγέθους της βάσης



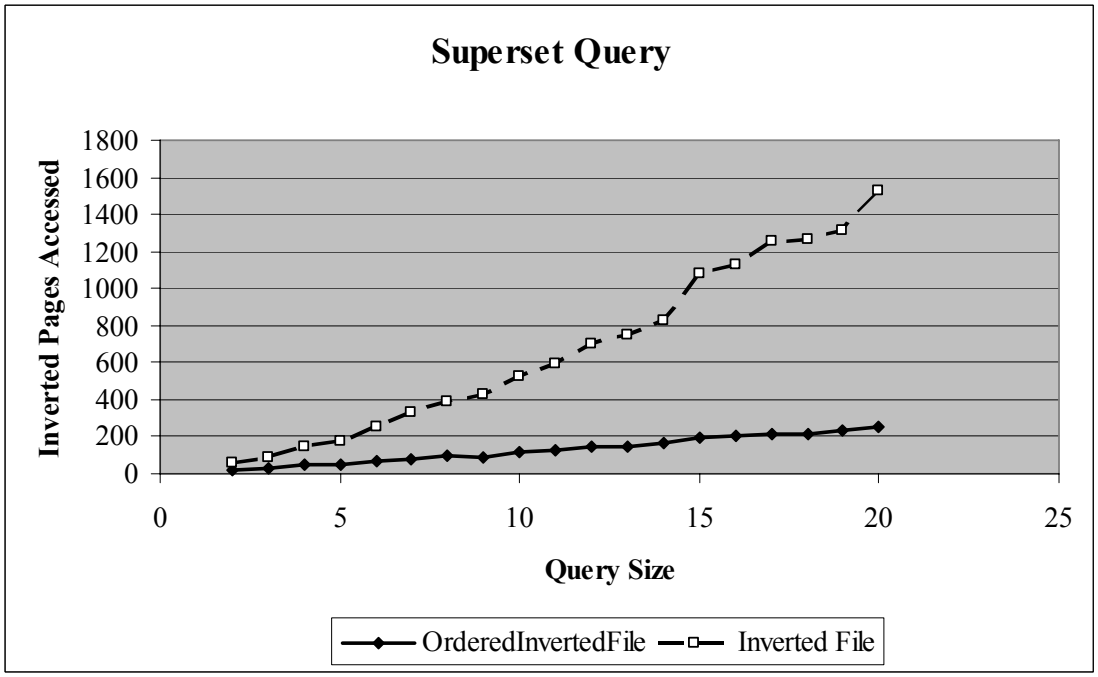
Σχήμα 5.31: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση superset queries συναρτήσει του μεγέθους της βάσης



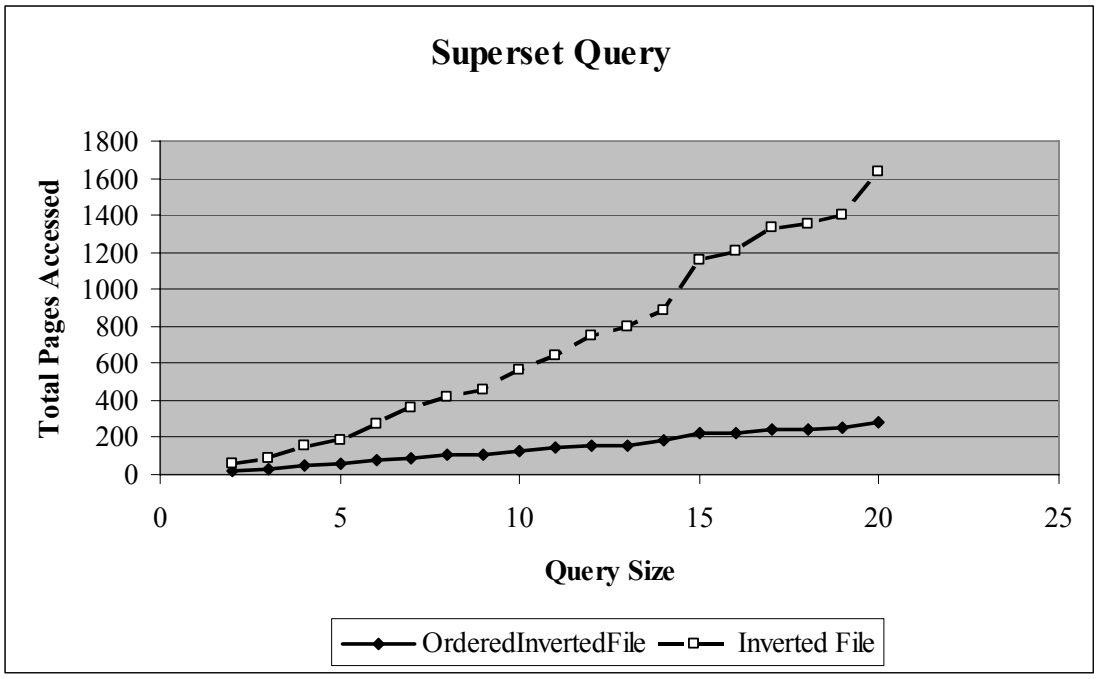
Σχήμα 5.32: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση superset queries συναρτήσει της διασποράς των αντικειμένων



Σχήμα 5.33: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση superset queries συναρτήσει της διασποράς των αντικειμένων



Σχήμα 5.34: Σύγκριση ordered inverted file και inverted file ως προς τις σελίδες των λιστών κατά την αποτίμηση superset queries συναρτήσει του μεγέθους των ερωτημάτων

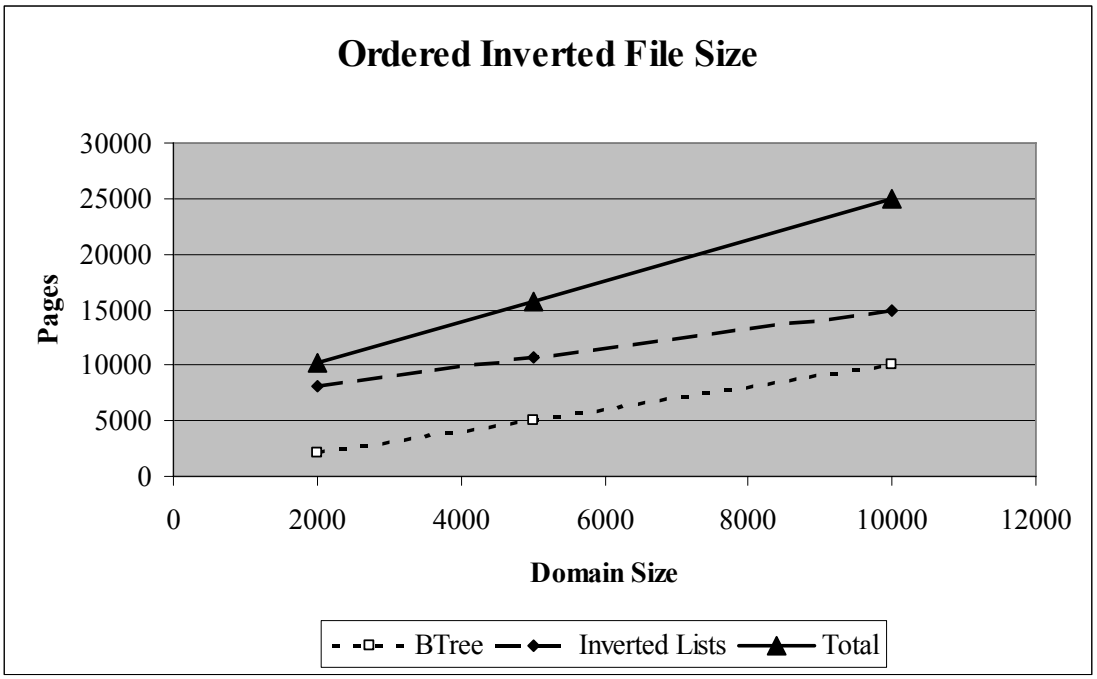


Σχήμα 5.35: Σύγκριση ordered inverted file και inverted file ως προς τις συνολικές σελίδες κατά την αποτίμηση superset queries συναρτήσει του μεγέθους των ερωτημάτων

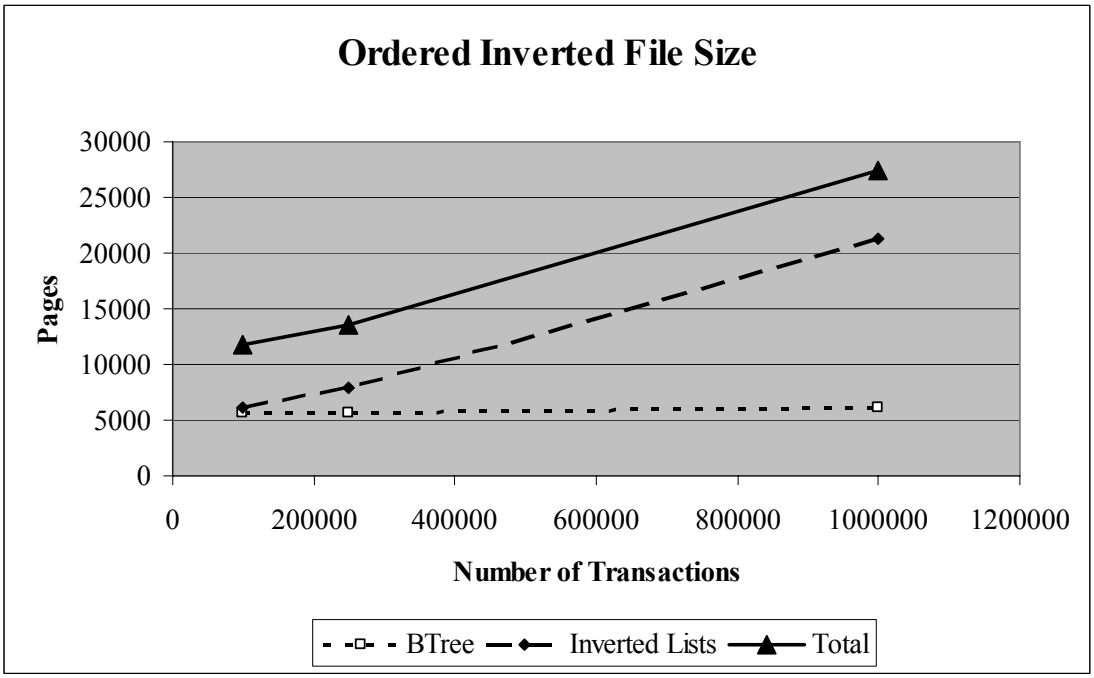
Παρατηρούμε ότι και σε αυτή την περίπτωση το ordered inverted file είναι αποδοτικότερο από το απλό inverted file, αφού στη χειρότερη περίπτωση το ordered inverted file θα προσπελάσει μία φορά την κάθε λίστα που εμπλέκεται στο ερώτημα, ενώ το απλό inverted file θα χρειαστεί να προσπελάσει την κάθε λίστα τόσες φορές όσα και τα βήματα της αναδρομής στα οποία αυτή εξετάζεται, αφού σε κάθε επανάληψη της αναδρομής διατρέχει σχεδόν ολόκληρες τις λίστες που εξετάζονται στο βήμα αυτό. Το γεγονός αυτό έχει σαν αποτέλεσμα το ordered inverted file να επηρεάζεται λιγότερο από τις μεταβολές στο μέγεθος των λιστών και των ερωτημάτων, με αποτέλεσμα η διαφορά της απόδοσης των δύο ευρετηρίων να μεγαλώνει σημαντικά με τις μεταβολές αυτές.

5.2.3 Μέγεθος Ordered Inverted File

Στην παράγραφο αυτή βλέπουμε τέλος το μέγεθος (σε σελίδες) που έχει γενικά το ordered inverted file και τον τρόπο που αυτό επηρεάζεται από τις μεταβολές της βάσης που εξετάσαμε παραπάνω. Το μέγεθος αυτό παρουσιάζεται στα σχήματα 5.36 έως 5.38 που ακολουθούν.



Σχήμα 5.36.: Επίδραση του μεγέθους του λεξιλογίου στο μέγεθος του ordered inverted file



Σχήμα 5.37.: Επίδραση του μεγέθους της βάσης στο μέγεθος του ordered inverted file

Σχήμα 5.36.: Επίδραση της διασποράς των αντικειμένων στο μέγεθος του ordered inverted file

Βλέπουμε ότι, όπως είναι αναμενόμενο, το μέγεθος των λιστών αυξάνεται καθώς αυξάνεται το μέγεθος της βάσης, χωρίς όμως να συμβαίνει το ίδιο και με το μέγεθος των Β-Δέντρων. Εφόσον στο κάθε Β-Δέντρο αποθηκεύεται ένα κλειδί για κάθε σελίδα της αντίστοιχης λίστας περιμένουμε τα Β-Δέντρα να μεγαλώνουν με πολύ μικρότερο ρυθμό από τις ανεστραμμένες λίστες, αλλά να παρουσιάζουν και αυτά γραμμική αύξηση. Αυτό δε συμβαίνει στην περίπτωσή μας γιατί τα δεδομένα τα οποία χρησιμοποιήσαμε στα πειράματά μας αποτελούνται από μικρό αριθμό δοσοληψιών, με αποτέλεσμα τα Β-Δέντρα που δημιουργούνται να περιέχουν πολλές μισοάδειες σελίδες. Έτσι, καθώς αυξάνεται το μέγεθος της βάσης, γεμίζουν οι σελίδες αυτές, χωρίς να χρειαστεί να προστεθούν καινούργιες σελίδες στο δέντρο. Η γραμμική αύξηση που περιμένουμε θα μπορούσε λοιπόν να παρατηρηθεί σε ακόμη μεγαλύτερες βάσεις, τις οποίες δε συμπεριλάβαμε στα πειράματά μας. Αντίθετα, το μέγεθος των Β-Δέντρων αυξάνεται γρηγορότερα με την αύξηση του λεξιλογίου, καθώς αυτό έχει σαν αποτέλεσμα τη δημιουργία περισσότερων λιστών και άρα περισσότερων Β-Δέντρων το καθένα από τα οποία βέβαια θα έχει μικρότερο μέγεθος. Τέλος παρατηρούμε ότι το μέγεθος του ευρετηρίου δεν επηρεάζεται σχεδόν καθόλου από τη διασπορά των αντικειμένων, αφού είτε έχουμε πολλές λίστες μετρίου μεγέθους, είτε λίγες πολύ μεγάλες και αρκετές μικρές το συνολικό μέγεθος του ευρετηρίου παραμένει το ίδιο.

Αξίζει εδώ να επισημάνουμε ότι στις περισσότερες περιπτώσεις το μέγεθος των Β-Δέντρων είναι αρκετά μικρότερο από αυτό των λιστών, γεγονός που μπορεί να κάνει δυνατή την αποθήκευσή τους στην κύρια μνήμη αυξάνοντας έτσι κι άλλο την απόδοση του ordered inverted file. Επιπλέον, το αυξημένο μέγεθος που το ευρετήριό μας έχει σε σχέση με το απλό

inverted file λόγω της παρουσίας των B-Δέντρων δεν είναι τέτοιο που να το καθιστά δύσχρηστο, δεδομένης ειδικά της μεγαλύτερης αποδοτικότητάς του σύμφωνα με τα όσα είδαμε στην ενότητα 5.2.2.

6

Επίλογος

6.1 Σύνοψη και συμπεράσματα

Στα κεφάλαια που προηγήθηκαν περιγράψαμε την ανάπτυξη και λειτουργία ενός ευρετηρίου για σύνθετα δεδομένα και συγκεκριμένα για τιμές σύνολα, το οποίο θέλαμε να είναι αποδοτικό στην αποτίμηση subset, equality και superset queries. Το ευρετήριο αυτό βασίστηκε στο ήδη υπάρχον inverted file το οποίο συνδυάσαμε με το B-Δέντρο, με σκοπό να ξεπεράσουμε ορισμένα από τα μειονεκτήματά του, και ονομάστηκε ordered inverted file.

Μετά από μία περιγραφή του τρόπου κατασκευής και λειτουργίας του ordered inverted file, καθώς και την παρουσίαση του κώδικα με τον οποίο υλοποιήσαμε το ευρετήριο, έγιναν πειράματα που συνέκριναν το ευρετήριό μας με το inverted file και τα οποία έδειξαν ότι υπερτερεί αυτού κατά πολύ κατά την αποτίμηση equality και superset queries, ενώ τα δύο ευρετήρια παρουσιάζουν πολύ μικρές διαφορές κατά την αποτίμηση subset queries, με το ordered inverted file να εμφανίζεται συνήθως λίγο αποδοτικότερο.

Επιπλέον, το ordered inverted file, αν και μεγαλύτερο από το απλό inverted file, δεν έχει μέγεθος τέτοιο που να δυσχεραίνει τη χρήση του. Έτσι, δεδομένης της απόδοσής του σε σχέση με το απλό inverted file μπορούμε τελικά να πούμε ότι το ordered inverted file ανταποκρίνεται στις απαιτήσεις μας και επιτυγχάνει τους στόχους που θέσαμε στην αρχή της εργασίας.

6.2 Μελλοντικές επεκτάσεις

Δεδομένου του γεγονότος ότι αυτή η πρώτη υλοποίηση του ordered inverted file αποδείχθηκε αρκετά αποδοτική μέσα από τα πειράματα που κάναμε, είναι σκόπιμο να προτείνουμε κάποιες δυνατές προσθήκες και βελτιώσεις.

Μία τέτοια προσθήκη είναι η υλοποίηση της ανανέωσης του ordered inverted file. Η ανανέωση αυτή εξετάστηκε θεωρητικά στο κεφάλαιο 3, δεν υλοποιήθηκε όμως στον κώδικα που παρουσιάσαμε στο κεφάλαιο 4. Έτσι, μία μελλοντική υλοποίηση της ανανέωσης και έρευνα πάνω στο πόσο αποδοτική είναι αυτή είναι απαραίτητη για την ολοκληρωμένη χρήση του ευρετηρίου.

Επιπλέον, στο κεφάλαιο 5 παρατηρήσαμε ότι αν και το ordered inverted file προσπελαύνει σημαντικά λιγότερες σελίδες κατά την αποτίμηση equality και superset queries από το απλό inverted file, κάτι τέτοιο δε συμβαίνει και με τα subset queries. Είναι έτσι ενδιαφέρον να υλοποιηθεί κάποια νέα και πιο αποδοτική μέθοδος αποτίμησης των ερωτημάτων αυτών. Μία τέτοια μέθοδος θα μπορούσε να παίρνει την τομή των περιοχών πιθανών απαντήσεων μόνο των i μικρότερων λιστών και στη συνέχεια να χρησιμοποιεί τα B-Δέντρα των υπολοίπων για να ψάξει μόνο για δοσοληψίες που εμφανίζονται στην τομή αυτή, χωρίς έτσι να χρειαστεί να διατρέξει ολόκληρες τις περιοχές πιθανών απαντήσεων των μεγαλύτερων λιστών.

Τέλος, θα μπορούσε να ερευνηθεί ακόμη ο τρόπος αποτίμησης και άλλων ειδών ερωτημάτων με χρήση του ordered inverted file και η απόδοση αυτού σε σχέση με το απλό inverted file, όπως έγινε στην εργασία αυτή για τα subset, equality και superset queries.

7

Βιβλιογραφία

[BR99] Baeza-Yates, R. and Ribeiro-Neto, B., Modern Information Retrieval, ACM Press/Addison-Wesley, 1999

[Car03] Carlson, D., Retrieval in B-Trees, from http://cis.stvincent.edu/swd/btree/retrieve.html, 2003

[Car05] Carlson, D., B-Trees, from http://cis.stvincent.edu/swd/btree/btree.html, 2005

[HM99] Helmer, S., Moerkotte, G., A Study of Four Index Structures for Set-Valued Attributes of LowCardinality, Reihe Informatik 2, 1999

[HPY+04] Han, J., Pei, J., Yin, Y.and Mao, R., Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Aproach, Data Mining and Knowledge Discovery, 8(1):53-87, 2004

[Neu98] Neumann, M., 433 Examples in 132 (or 162*) programming languages – C++: MergeSort, in http://www.ntecs.de/old-hp/uu9r/lang/html/cplusplus.en.html, 1998

[SKS01] Silberschatz, A., Korth, H.F. and Sudarshan, S., Database System Concepts, 4th Edition, McGraw-Hill, 2001

[TPV+06] Terrovitis, M., Passas, S., Vassiliadis, P.and Sellis. T., A Combination of Trie-Trees and Inverted Files for the indexing of set-valued attributes,

Technical Report, 2006

[WMB99] Witten, I.H., Moffat, A. and Bell, T.C., Managing Gigabytes: Compressing and Indexing Documents and Images, Morgan Kaufmann, 2nd Edition, 1999

[ZMR98] Zobel, J., Moffat, A. and Ramamohanarao, K., Inverted Files versus Signature Files for Text Indexing, ACM Transactions on Database Systems, 23(4):453-490, 1998

[ZMS92] Zobel, J., Moffat, A. and Sacks-Davis, R., An Efficient Indexing Technique for Full-Text Database Systems, Proceedings of the 18th VLDB Conference, 1992

[Πασ06] Πασσάς, Σ., Βελτιστοποίηση Μεθόδων Εξόρυξης Γνώσης, Διπλωματική Εργασία, 2006