Τ.Ε.Ι. ΜΕΣΟΛΟΓΓΙΟΥ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ & ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΓΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΣΤΗ ΔΙΟΙΚΗΣΗ & ΣΤΗΝ ΟΙΚΟΝΟΜΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΤΕΧΝΙΚΕΣ ΔΕΙΚΤΟΔΟΤΗΣΗΣ ΓΙΑ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ»

ΕΙΣΗΓΗΤΗΣ:

κ. ΜΑΚΡΗΣ

ΣΠΟΥΔΑΣΤΡΙΕΣ :

ΧΙΩΤΗ ΚΩΝ/ΝΑ

ΧΑΪΝΑ ΑΝΤΩΝΙΑ

ΜΕΣΟΛΟΓΓΙ 2005

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Η συγκεκριμένη εργασία έχει ως στόχο να παρουσιάσει τις διαφορετικές *τεχνικές δεικτοδότησης* που αφορούν την υποστήριξη της αποδοτικής πρόσβασης σε *χρονικά δεδομένα* Η σύγκριση των τεχνικών είναι βασισμένη σε μία συλλογή μετρικών που αποτελείται από τα σημαντικότερα κριτήρια απόδοσης, συμπεριλαμβανομένου του χώρου που καταναλώνεται, της επεξεργασίας πράξεων ενημέρωσης και του χρόνου ερώτησης για τα αντιπροσωπευτικά ερωτήματα. Πρόσθετα κριτήρια που εξετάζονται είναι οι σελιδοποιήσεις ενός δείκτη, η δυνατότητα να συγκεντρωθούν τα σχετικά στοιχεία μαζί και η δυνατότητα να χωριστούν αποτελεσματικά τα τρέχοντα από τα παλαιά στοιχεία Η σύγκριση είναι βασισμένη στην ανάλυση χειρότερης περίπτωσης και ως εκ τούτου δεν γίνεται καμία υπόθεση για την κατανομή δεδομένων και για την συχνότητα εμφάνισης κάθε είδους ερώτησης. Στη συνέχεια παρουσιάζεται η σύγκριση από κάποια αντιπροσωπευτικά είδη ερωτήσεων και προσδιορίζονται τα δύσκολα προβλήματα στην πρόσβαση των χρονικών δεδομένων και περιγράφεται το πώς οι διαφορετικές μέθοδοι προσπαθούν να λύσουν αυτά τα προβλήματα

Στην εργασία επίσης πραγματοποιείται μια μελέτη των *διαχρονικών* δομών δεδομένων. Οι συνηθισμένες δομές δεδομένων είναι εφήμερες υπό την έννοια ότι μια αλλαγή στη δομή καταστρέφει την παλαιά έκδοση, αφήνοντας μόνο τη νέα έκδοση διαθέσιμη για τη χρήση. Αντίθετα, μια διαχρονική δομή επιτρέπει την πρόσβαση σε οποιαδήποτε έκδοση, παλαιά ή νέα, οποιαδήποτε στιγμή. Αναπτύσσονται οι απλές, συστηματικές, και αποδοτικές τεχνικές για τις συνδεμένες διαχρονικές δομές δεδομένων. Χρησιμοποιούμε τις τεχνικές για να σχεδιάσουμε διαχρονικά δυαδικά δέντρα αναζήτησης με λογαριθμικούς χρόνους προσπέλασης, εισαγωγής, διαγραφής και O (1) απαιτήσεις χώρου για την εισαγωγή και τη διαγραφή.

Οι λειτουργίες αναζήτησης σε βάσεις δεδομένων, τόσο συμβατικές όσο και βάσεις που περιέχουν δεδομένα στο χώρο, απαιτούν ιδιαίτερη υποστήριξη στο φυσικό επίπεδο, και περιλαμβάνουν ερωτήσεις για σημεία («Βρες όλα τα αντικείμενα που περιέχουν δοσμένο σημείο») κι ερωτήσεις για περιοχές («Βρες όλα τα αντικείμενα που επικαλύπτουν μια δοσμένη περιοχή»). Στη συγκεκριμένη εργασία παρουσιάζεται επίσης μια σύνοψη των ερευνητικών αποτελεσμάτων σχετικά με μεθόδους προσπέλασης σε πολλές διαστάσεις για την υποστήριξη λειτουργιών αναζήτησης, εισαγωγής και διαγραφής σε βάσεις δεδομένων που περιέχουν δεδομένα στο χώρο (από εδώ και στο εξής θα καλούμε τέτοιες βάσεις σαν βάσεις σύνθετων δεδομένων).

1ο Κεφάλαιο

"Είδη Χρονικών Βάσεων"

1ο Κεφάλαιο

1.1 ΕΙΣΑΓΩΓΗ

Οι Χρονικές Βάσεις Δεδομένων δεν είναι πολύ διαδεδομένες σε σχέση με τις κλασικές Βάσεις Δεδομένων. Στα συνηθισμένα συστήματα βάσεων δεδομένων λαμβάνεται μόνο μία ενιαία λογική κατάσταση της πραγματικότητας. Κατά τη διάρκεια των συναλλαγών η κλασική Βάση Δεδομένων εξελίσσεται από μία συνεπή κατάσταση στην επόμενη καθώς η προηγούμενη κατάσταση χάνεται και αντικαθίσταται από την επόμενη όταν μία αλλαγή πραγματοποιείται. Γίνεται σαφές το γεγονός ότι τέτοιες Βάσεις Δεδομένων δεν έχουν την δυνατότητα να κρατούν πληροφορίες που αφορούν το παρελθόν, αλλά κρατούν ένα στιγμιότυπο (snapshot) της πραγματικότητας. Το χαρακτηριστικό αυτό θεωρείται μειονέκτημα για τις εφαρμογές που απαιτούν στοιχεία από το παρελθόν, το παρόν και το μέλλον. Στο σημείο αυτό δημιουργείται η ανάγκη εισαγωγής ενός νέου είδους βάσης δεδομένων, το χρονικό σύστημα βάσης δεδομένων.

*«Ο όρος **Χρονική Βάση Δεδομένων** (Temporal Database) αναφέρεται γενικά σε ένα σύστημα βάσεων δεδομένων που υποστηρίζει κάποια χρονική περιοχή (time domain) και είναι ικανή να διαχειριστεί τα χρονικώς μεταβαλλόμενα στοιχεία(time-varying data).»*

1.2 Κατηγορίες Βάσεων Δεδομένων

Μία ταξινόμηση του χρόνου στις βάσεις δεδομένων αναπτύχθηκε από ερευνητές. Συγκεκριμένα, ο χρόνος μπορεί να ταξινομηθεί σε δύο κατηγορίες, σε *transaction time*: ορίζεται ως ο χρόνος της διάρκειας αποθήκευσης ενός γεγονότος στη βάση δεδομένων και σε *valid time:* ορίζεται ως ο χρόνος που ένα γεγονός γίνεται ενεργό.

Παρακάτω ορίζονται τα τρία είδη χρονικών Β.Δ. κατά ορισμένους ερευνητές. Ανάλογα με ποιες διαστάσεις χρόνου υποστηρίζουν από τη Β.Δ.. οι κατηγορίες είναι τα εξής:

1.2.1 Transaction time Β.Δ. (Χρόνου Συναλλαγής Β.Δ.)

Στην περίπτωση αυτών των Β.Δ. χαρακτηριστικό γνώρισμα είναι η ανικανότητα να αλλαχθεί το παρελθόν, διατηρείται όμως η ιστορία όλων των προηγούμενων συναλλαγών.

1.2.2 Valid time Β.Δ. (Έγκυρου Χρόνου Β.Δ.)

Αυτές οι Β.Δ. συγκρατούν στοιχεία για το παρελθόν, το παρόν και το μέλλον. Στην περίπτωση αλλαγής ή διόρθωσης κάποιων στοιχείων, τα παλιά στοιχεία χάνονται και αντικαθιστούνται από τα νέα.

1.2.3 Bitemporal Β.Δ. (Διχρονικές Βάσεις)

Συνδυάζει τα χαρακτηριστικά των δύο παραπάνω τύπων. Αναπαριστά την πραγματικότητα και επιτρέπει αναδρομικές και εκ των υστέρων αλλαγές.

1.3 Το Tuple-Versioning Χρονικό Μοντέλο

Το συγκεκριμένο χρονικό μοντέλο χρησιμοποιείται σε αυτή την εργασία.. Σύμφωνα με αυτό το μοντέλο, η βάση δεδομένων είναι ένα σύνολο εγγραφών (ή tuples) που αποθηκεύουν τις εκδόσεις των αντικειμένων της πραγματικής ζωής. Κάθε τέτοια εγγραφή έχει ένα χρονικά σταθερό κλειδί (key) και ένα αριθμό από χρονικά μεταβαλλόμενα χαρακτηριστικά (time – varying attributes). Επιπλέον, έχει ένα ή δύο χρονικά διαστήματα (intervals), ανάλογα με το είδος χρόνου που υποστηρίζεται. Κάθε χρονικό διάστημα χαρακτηρίζεται από δύο τιμές: χρονική στιγμή έναρξης (start_time) και χρονική στιγμή τέλους (end_time), δηλαδή είναι της μορφής (start_time, end_time).

1. 4 Καθορισμός Προβλήματος

Στη συνέχεια θα παρουσιαστούν τα τρία είδη χρονικών Βάσεων Δεδομένων πιο αναλυτικά.

1.4.1 Transaction_Time DataBase (Χρόνου Συναλλαγής Β.Δ.)

Θεωρείται ένα άδειο αρχικά σύνολο αντικειμένων, το οποίο εξελίσσεται στο χρόνο. Ο χρόνος είναι διακριτός και περιγράφεται από μια σειρά συνεχόμενων, μη αρνητικών ακεραίων. Κάθε αλλαγή που συμβαίνει σε μια χρονική στιγμή, η χρονική αυτή στιγμή αντιστοιχεί σε κάποιους από τους ακεραίους. Μια αλλαγή αντιστοιχεί στην προσθήκη ή στην αφαίρεση ενός αντικειμένου ή στην τροποποίηση της τιμής κάποιου χαρακτηριστικού ενός αντικειμένου. Θεωρώντας ότι η τροποποίηση της τιμής ενός χαρακτηριστικού ισούται με την διαγραφή ενός αντικειμένου και την εισαγωγή ενός νέου, το οποίο είναι όμοιο με αυτό που διαγράφεται αλλά με αλλαγμένη την τιμή του συγκεκριμένου χαρακτηριστικού, θα μπορούσε να πει κάποιος ότι υπάρχουν μόνο διαγραφές και προσθήκες.

Ένα αντικείμενο θεωρείται "ζωντανό" από την στιγμή που εισάγεται στο σύνολο των αντικειμένων της βάσης μέχρι την στιγμή που θα διαγραφεί. Η "κατάσταση" ενός εξελισσόμενου συνόλου τη χρονική στιγμή t συμβολίζεται με S(t) και είναι το σύνολο των αντικειμένων που είναι ζωντανά αυτή την χρονική στιγμή. Οι παλιές καταστάσεις δεν μπορούν να αλλάξουν, ενώ αλλαγές γίνονται μόνο στην τρέχουσα κατάσταση S(t).

Παράδειγμα

Ένα παράδειγμα που εφαρμόζεται στην πραγματική ζωή είναι η εξέλιξη των εργαζομένων σε μια εταιρία. Κάθε εργαζόμενος έχει ένα αναγνωριστικό (ssn) και ένα χαρακτηριστικό (μισθός). Οι αλλαγές περιλαμβάνουν τις προσθήκες νέων εργαζομένων (προσλήψεις, επαναπροσλήψεις), αλλαγές μισθού, ή διαγραφές εργαζομένων (παραιτήσεις ή απολύσεις).

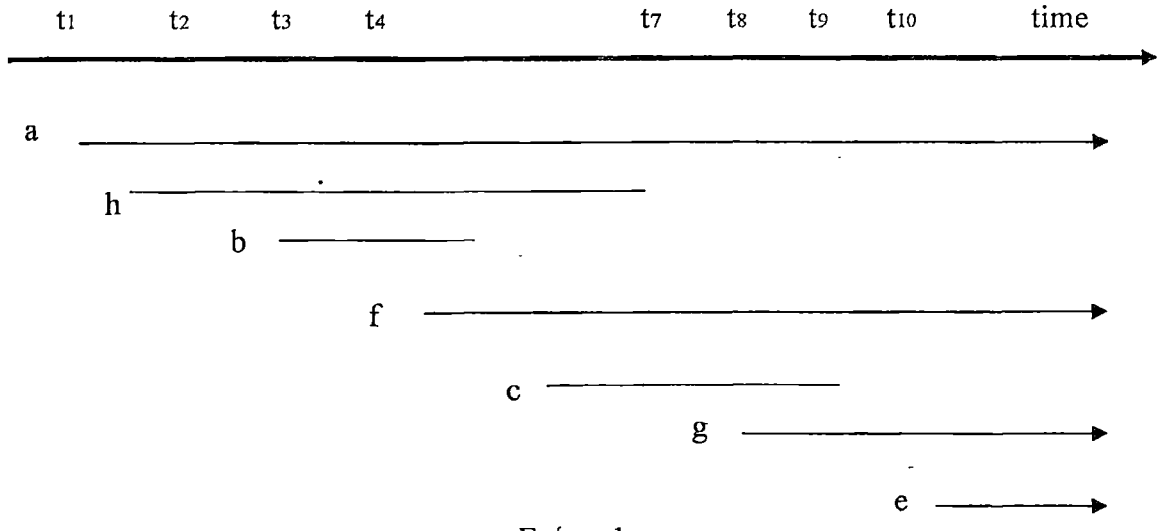
Αν υποτεθεί ότι η κατάσταση του παραδείγματος χρειάζεται να αποθηκευτεί σε μια βάση δεδομένων και εφόσον ο χρόνος πάντα αυξάνεται και το παρελθόν δεν αλλάζει, μπορεί να χρησιμοποιηθεί μια transaction_time database, με την ακόλουθη προϋπόθεση στην διαδικασία ενημέρωσης,.

Όταν ένα νέο αντικείμενο προστίθεται στο εξελισσόμενο σύνολο την χρονική στιγμή t, η εγγραφή που αντιπροσωπεύει αυτό το αντικείμενο, αποθηκεύεται στη βάση δεδομένων συνοδευόμενη από ένα χρονικό διάστημα χρόνου συναλλαγής της μορφής [t,now). Το now είναι μια μεταβλητή που περιέχει την τρέχουσα χρονική στιγμή συναλλαγής και αυτό γίνεται γιατί όταν ένα αντικείμενο γεννιέται δεν είναι ακόμα γνωστός ο χρόνος διαγραφής του. Αν το αντικείμενο σβηστεί στη χρονική στιγμή t΄, το διάστημα χρόνου συναλλαγής γίνεται [t,t΄). Συνεπώς, σε τέτοιου είδους βάσεις δεδομένων υποστηρίζονται οι "λογικές διαγραφές" (logical deletions), δηλαδή η εγγραφή του διαγραμμένου αντικειμένου παραμένει στην βάση, αλλά με διαφορετικό transaction end_time.
Για να γίνει πιο κατανοητή η έννοια της βάσης δεδομένων χρόνου συναλλαγής, παρουσιάζεται και σε σύγκριση με μια κλασική βάση δεδομένων. Μια κλασική βάση έχει μία και μόνη λογική κατάσταση (logical database state), την παρούσα. Μια Βάση Δεδομένων χρόνου συναλλαγής έχει περισσότερες από μια λογικές καταστάσεις.

Λογική κατάσταση μιας βάσης δεδομένων τη χρονική στιγμή t είναι το σύνολο των εγγραφών, των οποίων το χρονικό διάστημα χρόνου συναλλαγής περιέχει το t.

Μια Βάση Δεδομένων χρόνου συναλλαγής θα πρέπει να αποθηκεύει τις προηγούμενες λογικές καταστάσεις, να υποστηρίζει την πρόσθεση, την αφαίρεση και την τροποποίηση των αντικειμένων που βρίσκονται στην παρούσα κατάσταση και να έχει την ικανότητα προσπέλασης και ανάκτησης (μέσω ερωτημάτων) των αντικειμένων που ανήκουν σε κάθε μια από τις καταστάσεις της βάσης.

Το Σχήμα 1 παρουσιάζει την εξέλιξη ενός πεδίο χρόνου συναλλαγής στο χρόνο, όπου οι αλλαγές γίνονται σε αύξουσα χρονική σειρά.



Σχήμα 1

Οι γραμμές που καταλήγουν σε ▸ αντιστοιχούν σε αντικείμενα τα οποία δεν έχουν ακόμα διαγραφεί. Την στιγμή t_{10}, η κατάσταση $s(t_9) = \{a, f, g\}$ ενημερώνεται με την προσθήκη ενός αντικειμένου e και συνεπώς δημιουργείται μια νέα κατάσταση η $s(t_{10}) = \{a, f, g, e\}$.

1.4.2 Valid_Time Database (Έγκυρου Χρόνου Β.Δ.)

Μία Βάση Δεδομένων αποτελείται από αντικείμενα διαστημάτων (interval-objects). Λέγοντας έγκυρου χρόνου εννοούμε ότι το αντικείμενο μεταφέρει ένα χρονικό διάστημα, το οποίο δείχνει την έγκυρη περίοδο κάποιας ιδιότητας του αντικειμένου.

Σε μια Βάση Δεδομένων έγκυρου χρόνου, οι αλλαγές που ισχύουν για ένα αντικείμενο διαστήματος είναι η προσθήκη, η διαγραφή και η τροποποίηση του, ενώ όλες οι προηγούμενες καταστάσεις του διαγράφονται.

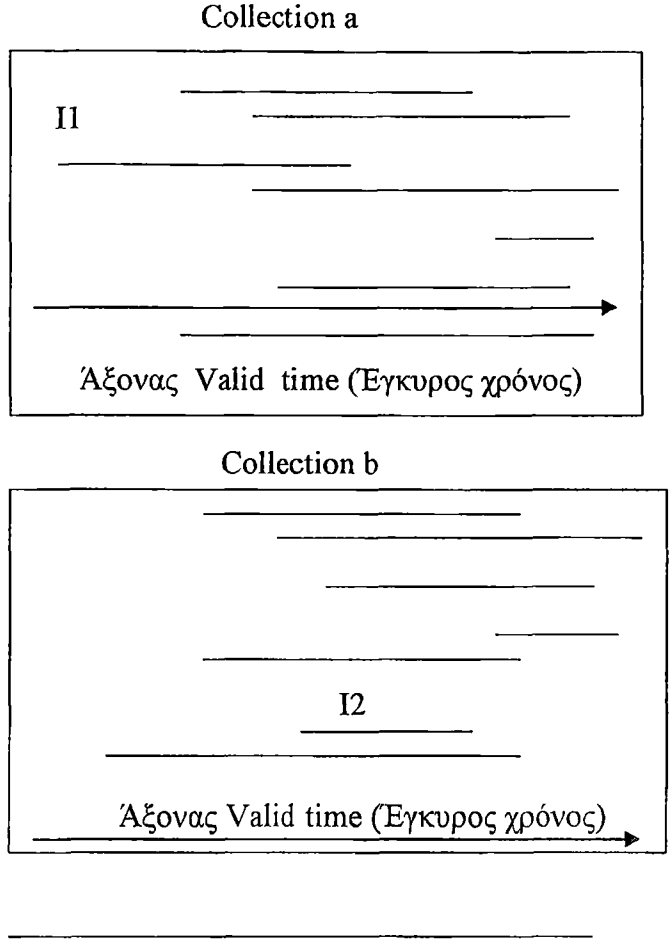
Με βάση ένα έγκυρου χρόνου σημείο t, τα αντικείμενα διαστήματος μπορούν να χωριστούν σε παρελθοντικά, ζωντανά (τρέχοντα) και μελλοντικά ως προς το t, ανάλογα με το διάστημα έγκυρου χρόνου που βρίσκονται. Γι' αυτό το λόγο ο χρόνος έχει σχέση με τον άξονα έγκυρου χρόνου. Η εγγραφή ενός αντικειμένου διαστήματος μπορεί να αλλάξει ανεξάρτητα από την θέση του στον άξονα έγκυρου χρόνου. Συνεπώς η συγκεκριμένη βάση έχει την δυνατότητα διόρθωσης λαθών σε οποιοδήποτε σημείο του πεδίου έγκυρου χρόνου.

Παράδειγμα

Ένα παράδειγμα που εφαρμόζεται στην πραγματική ζωή είναι η συλλογή συμβολαίων μιας εταιρίας. Κάθε συμβόλαιο έχει ένα αναγνωριστικό (contract_no), ένα χαρακτηριστικό (amount), και ένα χρονικό διάστημα το οποίο δίνει την εγκυρότητα του συμβολαίου. Όταν συμβεί μια διόρθωση, κρατιέται το διορθωμένο συμβόλαιο. Για ένα τέτοιο παράδειγμα χρησιμοποιούμε μια βάση δεδομένων έγκυρου χρόνου. Όταν ένα αντικείμενο προστεθεί στην συλλογή, τότε αποθηκεύεται μια εγγραφή στη βάση με τα χαρακτηριστικά του, συμπεριλαμβανομένου και του χρονικού διαστήματος έγκυρου χρόνου. Η χρονική στιγμή της εισαγωγής δεν κρατιέται. Όταν ένα αντικείμενο διαγράφεται από την βάση, τότε και η αντίστοιχη εγγραφή διαγράφεται. Αν τροποποιηθεί ένα χαρακτηριστικό του αντικειμένου, αλλάζει και η τιμή του και η προηγούμενη τιμή δεν διατηρείται. Μια βάση έγκυρου χρόνου κρατά μόνο την τελευταία "snapshot" της συλλογής των αντικειμένων και καμία πληροφορία για παλιές καταστάσεις ή για το πώς εξελίχθηκε. Μια τέτοια βάση μπορεί να αποθηκεύει εγγραφές με το ίδιο αναγνωριστικό, αλλά με μη επικαλυπτόμενα χρονικά διαστήματα έγκυρου χρόνου.

Για την επιτυχή προσπέλαση μιας τέτοιας βάσης θα πρέπει να υποστηρίζεται πρόσθεση, αφαίρεση και τροποποίηση. Επίσης θα πρέπει να αποθηκεύεται η τελευταία συλλογή των αντικειμένων και να μπορεί να απαντά αποδοτικά στις ερωτήσεις.
Στο σχήμα 2 φαίνονται οι δύο καταστάσεις μια συλλογής από αντικείμενα διαστήματος.

Η νέα συλλογή b δημιουργείται από την συλλογή a, μετά την διαγραφή του αντικειμένου I_1 και την προσθήκη του I_2. Η συλλογή b θα διατηρηθεί στην βάση δεδομένων.



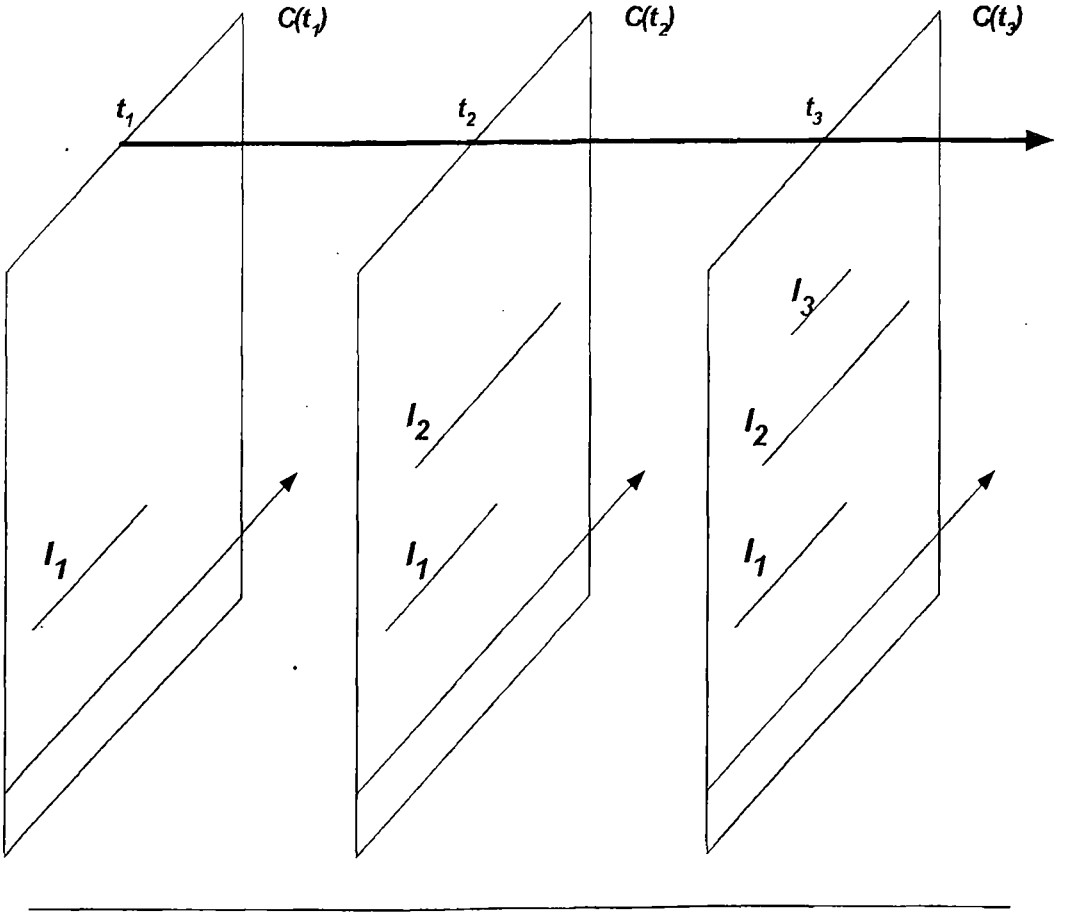
Σχήμα 2

1.4.3 Biteporal DataBase (Διχρονικές Β.Δ.)

Μια διχρονική βάση δεδομένων μπορεί να χαρακτηριστεί ως ο συνδυασμός των δυο προαναφερθέντων. Δηλαδή, σαν η εξέλιξη στον χρονικό άξονα συναλλαγών μιας δυναμικής συλλογής αντικειμένων διαστημάτων. Έχουμε μια ακολουθία από συλλογές αντικειμένων διαστημάτων που δεικτοδοτούνται στον χρονικό άξονα συναλλαγών. Όταν σε μια βάση δεδομένων εισαχθεί ένα αντικείμενο διαστήματος τη χρονική στιγμή t, τότε δημιουργείται μια εγγραφή που περιέχει το αναγνωριστικό (contract_no) του αντικειμένου, ένα χαρακτηριστικό (contract_amount), ένα valid_time χρονικό διάστημα (duration of contract) και ένα αρχικό transaction_time χρονικό διάστημα [t,now). Αν γίνει μια ενημέρωση τότε το now, μπορεί να αλλάξει μελλοντικά.

Το Σχήμα 3 δείχνει την αναπαράσταση μιας διχρονικής βάσης δεδομένων , όπου το διάστημα χρόνου συναλλαγής του I_2 αντικειμένου διαστήματος είναι $[t_2,t_3)$.

Για την επιτυχή προσπέλαση μιας διχρονικής βάσης θα πρέπει να υποστηρίζεται η αποθήκευση των προηγούμενων λογικών καταστάσεων στα αντικείμενα διαστήματος, να υποστηρίζεται πρόσθεση, αφαίρεση και τροποποίηση της τρέχουσας λογικής κατάστασης, και τέλος οι απαντήσεις και οι προσπελάσεις των ερωτήσεων όσον αφορά τα αντικείμενα διαστήματος που ανήκουν στις καταστάσεις της βάσης να είναι αποδοτικές.



Σχήμα 3

1.5 Διαφορές μεταξύ των τριών Βάσεων Δεδομένων

Οι διαφορές που προκύπτουν είναι οι ακόλουθες:

Χρόνος Συναλλαγής –Έγκυρος Χρόνος

Μια Βάση Δεδομένων χρόνου συναλλαγή διατηρεί την ιστορία της δραστηριότητάς της. Κάθε δοσοληψία με τη Βάση Δεδομένων προσδιορίζεται με μία χρονική ένδειξη. Είναι εφικτή η προσπέλαση διαφορετικών χρονικών στιγμιότυπων της Βάσης Δεδομένων. Αντίθετα, μια Βάση Δεδομένων έγκυρου χρόνου διατηρεί αντικείμενα που μία συνιστώσα τους υποδηλώνει χρονικά στιγμιότυπα. Το σύνολο των τιμών αυτών αποθηκεύουν τις τωρινές γνώσεις για το παρόν, το παρελθόν ή ακόμα και το μέλλον των αποθηκευμένων αντικειμένων.

Χρόνου Συναλλαγής – Διχρονικές Β.Δ.
Μια Βάση Δεδομένων χρόνου συναλλαγής διατηρεί την ιστορία ενός εξελισσόμενου συνόλου από αντικείμενα, ενώ μία Διχρονική Βάση Δεδομένων κρατά την ιστορία ενός εξελισσόμενου συνόλου από αντικείμενα διαστήματος.

Έγκυρου Χρόνου - Διχρονικές
Μια έγκυρου χρόνου Βάση Δεδομένων κρατά μόνο την τελευταία συλλογή από τα αντικείμενα διαστήματος. Δηλαδή, κάθε συλλογή $C(t_i)$ μιας Βάσης Δεδομένων μπορεί να θεωρηθεί σαν μια ξεχωριστή Βάση Δεδομένων έγκυρου χρόνου.

1.6 Σχολιασμός

Σε μία συμβατική Βάση Δεδομένων, τα πάντα μπορούν να ενημερωθούν. Έάν τα ιστορικά στοιχεία μπορούν να ενημερωθούν, βρισκόμαστε αντιμέτωποι με τη πιθανότητα ότι, προφανώς, η ιστορία μπορεί να αλλαχτεί. Τι γίνεται με αυτή την πιθανότητα;

Είναι σημαντικό να γίνει κατανοητό ότι η Βάση Δεδομένων δεν περιέχει "τον πραγματικό κόσμο," περιέχει μόνο τη *γνώση ή τις πεποιθήσεις* μας για τον πραγματικό κόσμο. Και ενώ είναι τέλεια λογικό να βεβαιωθεί ότι το παρελθόν είναι αμετάβλητο και η ιστορία δεν μπορεί υπό αυτήν τη μορφή ποτέ να αλλάξει, είναι εξίσου λογικό να βεβαιωθεί ότι οι πεποιθήσεις μας για αυτήν μπορούν να αλλάξουν. Σε ένα πλαίσιο βάσεων δεδομένων, επομένως, όταν μιλάμε για την "ενημέρωση ιστορίας" σημαίνει ότι πραγματικά ενημερώνει τις πεποιθήσεις μας για την ιστορία, όχι ότι ενημερώνει την ιστορία .

Παράδειγμα
Έστω το p να είναι η πρόταση "Ο προμηθευτής ήταν σε σύμβαση" υποθέτει ότι είναι η ισχύων κατανόηση μας ότι αυτή η παρούσα κατάσταση λαμβάνεται από την 1η Ιουλίου του 1999, μέχρι την 1η Μαΐου του 2000, και επομένως παρεμβάλλουμε την ακόλουθη πλειάδα στη Βάση Δεδομένων:

S#	ΑΠΟ	ΕΩΣ
S1	1η Ιουλίου 1999	1η Μαΐου 2000

Σημειώνεται πολύ προσεκτικά ότι αυτή η πλειάδα δεν αντιστοιχεί στην πρόταση p. Μάλλον, αντιστοιχεί σε αυτό που μπορεί να κληθεί η timestamp επέκταση της πρότασης p που μπορεί να δηλωθεί έτσι: "Ο Προμηθευτής S_1 ήταν στο πλαίσιο της σύμβασης από την 1η Ιουλίου 1999, έως την 1η Μαΐου 2000". Κατά συνέπεια, ο έγκυρος χρόνος για το p σε αυτό το παράδειγμα είναι το διάστημα του χρόνου κατά τη διάρκεια του οποίου (σύμφωνα με τις παρούσες πεποιθήσεις μας) το p ήταν στην πραγματικότητα αληθινό. Σημείωση: υποθέτουμε για την απλότητα ότι το προσδιορισμένο διάστημα (από την 1η Ιουλίου του 1999, μέχρι την 1η Μαΐου του 2000) είναι ο μόνος έγκυρος χρόνος για το p. Γενικότερα, ο έγκυρος χρόνος για μια δεδομένη πρόταση είναι ένα σύνολο διαστημάτων, όχι ένα μοναδικό διάστημα. Παραδείγματος χάριν, εάν η κατανόησή μας ότι ο προμηθευτής S1 ήταν επίσης στο πλαίσιο της σύμβασης προηγουμένως από την 1η Ιανουαρίου το 1998, έως την 1η Απριλίου το 1998, κατόπιν ο σχετικός έγκυρος χρόνος θα περιελάμβανε σαφώς δύο διαστήματα, όχι μόνο ένα.
Υποθέστε ότι τώρα μαθαίνουμε ότι η σύμβαση του προμηθευτή S_1 άρχισε στην πραγματικότητα στις 1 Ιουνίου, όχι 1 Ιουλίου, και επομένως αντικαθιστάμε το αρχικό tuple από ένα άλλο που μοιάζει με αυτό:

S#	ΑΠΟ	ΕΩΣ
S1	1η Ιουνίου 1999	1η Μάιου 2000

Αυτή η αλλαγή δεν έχει καμία επίδραση στην πρόταση p υπό αυτήν τη μορφή, φυσικά αλλάζει μόνο το σχετικό timestamp (π.χ. απεικονίζει την αναθεωρημένη κατανόησή μας ότι ο προμηθευτής S_1 ήταν στην πραγματικότητα στο πλαίσιο της σύμβασης από την 1η Ιουνίου του 1999, μέχρι την 1η Μάιου του 2000). Κατά συνέπεια, ο έγκυρος χρόνος για την πρόταση p είναι τώρα το διάστημα από την 1η Ιουνίου το 1999, έως την 1η Μαΐου το 2000. Με άλλα λόγια, μόλις "ενημερώσαμε την πεποίθησή μας για την ιστορία." . Τι βεβαίως δεν έχουμε κάνει, εντούτοις, είναι η ενημέρωση στην ιστορία καθώς η ενημέρωση που αποδώσαμε δεν κάνει και δεν μπορεί να αλλάξει το ιστορικό γεγονός όπου η Βάση Δεδομένων προηγουμένως έδειξε των προμηθευτή S_1 κάτω από τη σύμβαση από 1η Ιουλίου (όχι Ιουνίου) το 1999.

Τέλος, υποθέστε ότι ανακαλύπτουμε ότι ένα λάθος έχει γίνει και ο προμηθευτής S_1 δεν ήταν ποτέ στο πλαίσιο της σύμβασης .Επομένως διαγράφουμε την πλειάδα για τον προμηθευτή S_1 εξ ολοκλήρου. Η πρόταση p είναι τώρα γνωστή για να είναι ψεύτικη Κατά συνέπεια, δεν υπάρχει τώρα κανένας έγκυρος χρόνος που συνδέεται με αυτό καθόλου. (Ισοδύναμα, μπορέσαμε να πούμε ότι ο έγκυρος χρόνος είναι τώρα ένα κενό σύνολο διαστημάτων.)

Τώρα υποθέστε ότι η αρχική πλειάδα παρεμβλήθηκε στο χρόνο t_1 και αντικαταστάθηκε στο χρόνο t_2, και εκείνη η αντικατεστημένη πλειάδα διαγράφηκε έπειτα στο χρόνο t_3. Κατόπιν η λογοτεχνία θα έλεγε ότι το διάστημα από το t_1 στο t_2 ήταν ο χρόνος συναλλαγής, όχι για την πρόταση p υπό αυτήν τη μορφή, αλλά μάλλον για την timestamp επέκταση του p με "έγκυρο-χρόνο timestamp" 1η Ιουλίου το 1999, έως την 1η Μαΐου το 2000. Δηλαδή το διάστημα από το t1 στο t2 είναι ο χρόνος κατά

τη διάρκεια του οποίου η Βάση Δεδομένων βεβαίωσε ότι αυτή η ιδιαίτερη επέκταση του p ήταν αληθινή. Επιπλέον, η λογοτεχνία επίσης θα έλεγε ότι το διάστημα από το t_2 στο t_3 ήταν ο χρόνος συναλλαγής για την timestamp επέκταση του p με έγκυρο-χρόνο timestamp 1η Ιουνίου το 1999, έως την 1η Μαΐου το 2000 δηλαδή το διάστημα από το t2 στο t3 είναι ο χρόνος κατά τη διάρκεια του οποίου η Βάση Δεδομένων βεβαίωσε ότι εκείνη η ιδιαίτερη επέκταση του p ήταν αληθινή.

ΣΗΜΕΙΩΣΗ: Γενικά, οι χρόνοι συναλλαγής (όπως στους έγκυρους χρόνους) είναι σύνολα διαστημάτων, όχι μόνο μεμονωμένα διαστήματα αυτό καθ' εαυτό. Παραδείγματος χάριν, εάν η Βάση Δεδομένων εμφάνισε πρόσθετα ότι η πρώτη από τις timestamp επεκτάσεις του p είναι αληθινή κατά τη διάρκεια του διαστήματος από t_4 σε t_5, τότε ο σχετικός χρόνος συναλλαγής θα περιελάμβανε σαφώς δύο διαστήματα, όχι μόνο ένα.

Μέχρι τότε, είναι ικανοποιητικό να τονιστεί το σημείο ότι-όπως τα παραδείγματά μας έχουν προτείνει ότι οι έγκυροι χρόνοι μπορούν να ενημερωθούν, αλλά οι χρόνοι συναλλαγής δεν μπορούν (οι έγκυροι χρόνοι απεικονίζουν τις πεποιθήσεις μας για την ιστορία, και εκείνες οι πεποιθήσεις μπορούν να αλλάξουν οι χρόνοι συναλλαγής, σε αντίθεση, απεικονίζουν την ιστορία υπό αυτήν τη μορφή, και η ιστορία δεν μπορεί να αλλάξει).

2ο Κεφάλαιο

"Οργάνωση σύνθετων δεδομένων"

2ο Κεφάλαιο

2.1 Εισαγωγή

Ο ολοένα αυξανόμενος αριθμός εφαρμογών που βασίζονται σε μεγάλο βαθμό σε δεδομένα πολλαπλών διαστάσεων έχει – εδώ και καιρό – καταστήσει αναγκαία τη μελέτη τρόπων διαχείρισης σύνθετων δεδομένων. Τέτοιου είδους δεδομένα συναντώνται σε πολλές επιστημονικές περιοχές όπως γεω-επιστήμες, μηχανολογία (σχέδιο), σχεδιασμός ηλεκτρονικών μικροκυκλωμάτων (VLSI), ρομποτική, τεχνητή όραση, αυτόνομη πλοήγηση, περιβαλλοντική προστασία, ανάλυση ιατρικών εικόνων κ.α.. Στην περίπτωση του σχεδιασμού κυκλωμάτων VLSI, τα δεδομένα είναι δισδιάστατα. Στην περίπτωση του μηχανολογικού σχεδίου τα δεδομένα εκτείνονται στο χώρο – και κατά συνέπεια έχουν τρεις διαστάσεις. Για την αναπαράστασή τους χρησιμοποιούνται :

- μέθοδοι διάσπασης σε μικρότερα τμήματα
- κατασκευαστική γεωμετρία στερεών (Constructive Solid Geometry)
- αναπαραστάσεις των ορίων τους

Αξίζει να σημειωθεί η διαφορά ανάμεσα σε βάσεις σύνθετων αντικειμένων και σε βάσεις που περιέχουν εικόνες. Οι μεν πρώτες, περιέχουν δεδομένα πολλαπλών διαστάσεων με πλήρη στοιχεία που αφορούν τα αντικείμενα αυτά, όπως ρητές πληροφορίες για το μέγεθος και θέση τους σε κάποιο σχήμα. Η απεικόνιση γίνεται συνήθως με τη μορφή διανυσμάτων. Οι δε δεύτερες, περιέχουν απλά αναπαραστάσεις αντικειμένων, χωρίς περαιτέρω στοιχεία σχετικά με το είδος, τα χαρακτηριστικά και τη ιδιαίτερη σημασία τους. Δεν αποδίδεται σημασία για την ανάλυση των προς αποθήκευση δεδομένων. Οι βάσεις εικόνων αποτελούν απλά αποθηκευτικά μέσα για μη επεξεργασμένα δεδομένα, π.χ. εικόνες, σήματα από αισθητήρες, ηχητικά φάσματα, κ.τ.λ. Δεδομένα όπως ακτινογραφίες ή δορυφορικές εικόνες αποθηκεύονται όπως είναι σαν εικόνες σε βάσεις δεδομένων που περιέχουν εικόνες, χωρίς παραπάνω ανάλυση.

2.1.2 Βασικές Ιδιότητες Δεδομένων Σε Βάσεις Σύνθετων Αντικειμένων

Τα δεδομένα που αποθηκεύονται σε βάσεις σύνθετων δεδομένων εμφανίζουν κάποια βασικά χαρακτηριστικά, τα οποία συνοψίζονται ως εξής :

- *Διαθέτουν πολύπλοκη δομή.* Μπορεί να γίνεται λόγος για ένα απλό σημείο ή για μερικές χιλιάδες πολύγωνα διασπαρμένα στο χώρο.
- *Μεταβάλλονται, συνήθως, δυναμικά με το χρόνο,* λόγω εισαγωγών και διαγραφών σημείων ή και ολόκληρων περιοχών.
- *Έχουν μεγάλο μέγεθος.* Γεωγραφικοί χάρτες, για παράδειγμα, ενδεχομένως καταλαμβάνουν πολλά Gigabytes αποθηκευτικού χώρου.
- *Δεν υπάρχει καθορισμένο σύνολο λειτουργιών* που μπορούν να επιτελεστούν σε αυτά. Εντούτοις, υπάρχουν λειτουργίες που συναντώνται πιο συχνά σε σύγκριση με άλλες, π.χ. τομή.

- *Δεν ισχύει η κλειστότητα* ως προς τις πράξεις που επιτελούνται σε βάσεις σύνθετων δεδομένων. Για παράδειγμα, η τομή δύο πολυγώνων μπορεί να είναι σύνολο διασκορπισμένων σημείων, ακμές ή ακόμα και ξένα μεταξύ τους πολύγωνα.
- Η διατήρηση και διαχείριση βάσεων σύνθετων δεδομένων έχει *αυξημένο κόστος* συγκριτικά με τις απλές βάσεις δεδομένων.

Η αναζήτηση σε βάσεις σύνθετων δεδομένων προϋποθέτει εύρεση και ενημέρωση σύνθετων δεδομένων που ισοδυναμούν με ανάγκη ταχύτατης εκτέλεσης διαδικασιών αναζήτησης / εύρεσης σε γεωγραφικό χώρο. Η λύση δίνεται από την ανάπτυξη ειδικών μεθόδων προσπέλασης σε σύνθετα δεδομένα, ενώ το βασικό πρόβλημα που εμφανίζεται οφείλεται στη μη ύπαρξη διάταξης στα αποθηκευμένα δεδομένα.

Οι ειδικές αυτές μέθοδοι προσπέλασης ενδέχεται να χειρίζονται μία μόνο διάσταση και να υλοποιούνται από αντίστοιχες δομές (π.χ. B-trees). Μέθοδοι αναζήτησης για πολλές διαστάσεις μπορούν να προκύψουν από επαναληπτική εφαρμογή των δομών αναζήτησης μιας διάστασης τόσες φορές όσες είναι οι διαστάσεις. Ο τρόπος όμως αυτός δεν παρουσιάζει καλή απόδοση μιας και δεν είναι προφανές το πώς μπορεί να γενικευτεί με ορθότητα η ιδέα της επαναληπτικής εφαρμογής.

2.2 Βασικές Ιδιότητες Αναζήτησης Σε Βάσεις Σύνθετων Δεδομένων

Οι μέθοδοι προσπέλασης για πολλές διαστάσεις θα πρέπει να διαθέτουν κάποιες σημαντικές ιδιότητες, ικανοποιώντας έτσι συγκεκριμένες απαιτήσεις. Αυτές είναι οι παρακάτω :

- Δυναμικότητα: να κρατιέται αρχείο με τις πραγματοποιήσιμες αλλαγές, έστω κι αν αυτές γίνονται χωρίς συστηματικό τρόπο.
- Διαχείριση δευτερογενούς / τριτογενούς αποθηκευτικού χώρου : αναγκαιότητα διαφανούς ολοκλήρωσης (αποφυγή μεγάλων καθυστερήσεων και πολλών προσπελάσεων στις θέσεις μνήμης) της κύριας μνήμης και των λοιπών αποθηκευτικών χώρων.
- Μεγάλο εύρος υποστηριζόμενων λειτουργιών / πράξεων : δεν πρέπει η σωστή και αποδοτική πραγματοποίηση μιας λειτουργίας (π.χ. εύρεσης) να γίνεται σε βάρος άλλων λειτουργιών.
- Ανεξαρτησία των δεδομένων εισόδου από τη σειρά καταχώρησής τους: οι μέθοδοι πρέπει να λειτουργούν ορθά και αποδοτικά ακόμα και στην περίπτωση διασκορπισμένων κι ελάχιστα δομημένων δεδομένων.
- Απλότητα: η απλή υλοποίηση συντελεί καθοριστικά στην ομαλή ενσωμάτωση και χρησιμοποίηση τέτοιων μεθόδων ακόμα και σε ένα πολύπλοκο περιβάλλον.
- Διαβαθμισιμότητα: οι μέθοδοι προσπέλασης θα πρέπει να προσαρμόζονται στην σταδιακή επέκταση της βάσης δεδομένων.
- Αποδοτικότητα σε χρόνο: να εκτελούνται με υψηλή ταχύτητα.
- . Αποδοτικότητα σε χώρο: να εκτελούνται με τις ελάχιστες απαιτήσεις και την ελάχιστη επιβάρυνση σε αποθηκευτικό χώρο.
- Υποστήριξη ταυτόχρονων λειτουργιών με αξιοπιστία και ορθή ανάκαμψη μετά από πιθανή βλάβη.
- Ελάχιστη δυνατή επιβάρυνση για τα υπάρχοντα μέρη του συστήματος στο οποίο πρόκειται να ενσωματωθούν .

Για την εφαρμογή μεθόδων προσπέλασης πολλών διαστάσεων σε βάσεις σύνθετων αντικειμένων χρησιμοποιείται η έννοια των πολυτόπων, γεωμετρικών κατασκευασμάτων δηλαδή, που προκύπτουν από την τομή ημιχώρων .Για τα πολύτοπα υπάρχουν οι έννοιες των κορυφών και των ακμών ελαφρώς τροποποιημένες από τη συνήθη ερμηνεία τους. Το αποτέλεσμα της ένωσης πολυτόπων είναι ένα πολύεδρο. Χαρακτηριστικό είναι ότι επιτρέπονται πολύεδρα με εσωτερικές ασυνέχειες (δηλ. τρύπες), ενώ για κάθε πολύεδρο ορίζεται το εξωτερικό μέρος, το εσωτερικό μέρος και τα όριά του. Δεδομένου ότι τα πολύεδρα που μπορούν να προκύψουν κατά τη διάρκεια μιας τέτοιας διαδικασίας ενδεχομένως έχουν εξαιρετικά πολύπλοκη δομή, συνήθως προσεγγίζονται με κάποια απλούστερα, βασικά πολύεδρα χωρίς σημαντικές απώλειες σε ουσιαστική πληροφορία. Η τεχνική αυτή αναφέρεται και σαν τεχνική του Minimun Bounding Box (ελάχιστου περιβάλλοντος ορθοκανονικού πολυτόπου).

2.3 Τρόποι Επεξεργασίας Ερωτήσεων σε Βάσεις Σύνθετων Δεδομένων

Δύο είναι οι βασικές προσεγγίσεις στην αντιμετώπιση ερωτήσεων σε βάσεις σύνθετων δεδομένων .
Σύμφωνα με την πρώτη, αρχικά προσδιορίζεται προσεγγιστικά το αντικείμενο προς αναζήτηση. Στη συνέχεια, αφού έχει περιοριστεί σημαντικά ο χώρος των πιθανών απαντήσεων, ολοκληρώνεται η αναζήτηση και επιστρέφεται η απάντηση.
Σύμφωνα με τη δεύτερη προσέγγιση, το προς αναζήτηση αντικείμενο αποσυντίθεται και εμφανίζεται σαν ένωση άλλων απλούστερων αντικειμένων των οποίων η εύρεση είναι απλούστερη και ταχύτερη. Έτσι, επιτυγχάνεται αποδοτικότερη επεξεργασία σχετικά τόσο με το χρόνο που χρειάστηκε προκειμένου να απαντηθεί η ερώτηση όσο και με τις συνθήκες υπό τις οποίες έγινε αυτό. Ο όρος «συνθήκες» αφορά τόσο στον απαιτούμενο αποθηκευτικό χώρο, όσο και στον απαιτούμενο χρόνο που μπορεί να εκφραστεί είτε σαν πλήθος προσπελάσεων στο δίσκο είτε σαν απαιτούμενοι κύκλοι λειτουργίας της Κεντρικής Μονάδας Επεξεργασίας.

2.4 Τρόποι Διατύπωσης Ερωτήσεων Σε Μια Βάση Σύνθετων Δεδομένων

Οι ερωτήσεις προς μια βάση σύνθετων δεδομένων γίνεται συνήθως με χρήση μιας εκτεταμένης SQL γλώσσας ενώ τα αποτελέσματα /απαντήσεις έχουν τη μορφή σύνθετων αντικειμένων .

Στη συνέχεια παραθέτουμε κάποιους βασικούς τύπους ερωτήσεων προς βάσεις σύνθετων αντικειμένων .

- Δεδομένου ενός αντικειμένου ερώτησης, εντόπισε αντικείμενα:
  - ✓ που έχουν ίδια έκταση με το δοσμένο αντικείμενο.
  - ✓ που έχουν τουλάχιστον ένα κοινό σημείο με το δοσμένο αντικείμενο.
  - ✓ που περιέχουν το δοσμένο αντικείμενο.
  - ✓ που περιέχονται στο δοσμένο αντικείμενο.
  - ✓ γειτονικά προς το δοσμένο αντικείμενο.
  - ✓ σε ελάχιστη απόσταση από το δοσμένο αντικείμενο.
- Δεδομένου ενός σημείου, εντόπισε αντικείμενα που το περιέχουν .
- Δεδομένης μιας περιοχής στο χώρο, εντόπισε αντικείμενα που την περιέχουν .

Για την εφαρμογή μεθόδων προσπέλασης πολλών διαστάσεων σε βάσεις σύνθετων αντικειμένων χρησιμοποιείται η έννοια των πολυτόπων, γεωμετρικών κατασκευασμάτων δηλαδή, που προκύπτουν από την τομή ημιχώρων .Για τα πολύτοπα υπάρχουν οι έννοιες των κορυφών και των ακμών ελαφρώς τροποποιημένες από τη συνήθη ερμηνεία τους. Το αποτέλεσμα της ένωσης πολυτόπων είναι ένα πολύεδρο. Χαρακτηριστικό είναι ότι επιτρέπονται πολύεδρα με εσωτερικές ασυνέχειες (δηλ. τρύπες), ενώ για κάθε πολύεδρο ορίζεται το εξωτερικό μέρος, το εσωτερικό μέρος και τα όριά του. Δεδομένου ότι τα πολύεδρα που μπορούν να προκύψουν κατά τη διάρκεια μιας τέτοιας διαδικασίας ενδεχομένως έχουν εξαιρετικά πολύπλοκη δομή, συνήθως προσεγγίζονται με κάποια απλούστερα, βασικά πολύεδρα χωρίς σημαντικές απώλειες σε ουσιαστική πληροφορία. Η τεχνική αυτή αναφέρεται και σαν τεχνική του Minimun Bounding Box (ελάχιστου περιβάλλοντος ορθοκανονικού πολυτόπου).

2.3 Τρόποι Επεξεργασίας Ερωτήσεων σε Βάσεις Σύνθετων Δεδομένων

Δύο είναι οι βασικές προσεγγίσεις στην αντιμετώπιση ερωτήσεων σε βάσεις σύνθετων δεδομένων .
Σύμφωνα με την πρώτη, αρχικά προσδιορίζεται προσεγγιστικά το αντικείμενο προς αναζήτηση. Στη συνέχεια, αφού έχει περιοριστεί σημαντικά ο χώρος των πιθανών απαντήσεων, ολοκληρώνεται η αναζήτηση και επιστρέφεται η απάντηση.
Σύμφωνα με τη δεύτερη προσέγγιση, το προς αναζήτηση αντικείμενο αποσυντίθεται και εμφανίζεται σαν ένωση άλλων απλούστερων αντικειμένων των οποίων η εύρεση είναι απλούστερη και ταχύτερη. Έτσι, επιτυγχάνεται αποδοτικότερη επεξεργασία σχετικά τόσο με το χρόνο που χρειάστηκε προκειμένου να απαντηθεί η ερώτηση όσο και με τις συνθήκες υπό τις οποίες έγινε αυτό. Ο όρος «συνθήκες» αφορά τόσο στον απαιτούμενο αποθηκευτικό χώρο, όσο και στον απαιτούμενο χρόνο που μπορεί να εκφραστεί είτε σαν πλήθος προσπελάσεων στο δίσκο είτε σαν απαιτούμενοι κύκλοι λειτουργίας της Κεντρικής Μονάδας Επεξεργασίας.

2.4 Τρόποι Διατύπωσης Ερωτήσεων Σε Μια Βάση Σύνθετων Δεδομένων

Οι ερωτήσεις προς μια βάση σύνθετων δεδομένων γίνεται συνήθως με χρήση μιας εκτεταμένης SQL γλώσσας ενώ τα αποτελέσματα /απαντήσεις έχουν τη μορφή σύνθετων αντικειμένων .

Στη συνέχεια παραθέτουμε κάποιους βασικούς τύπους ερωτήσεων προς βάσεις σύνθετων αντικειμένων .

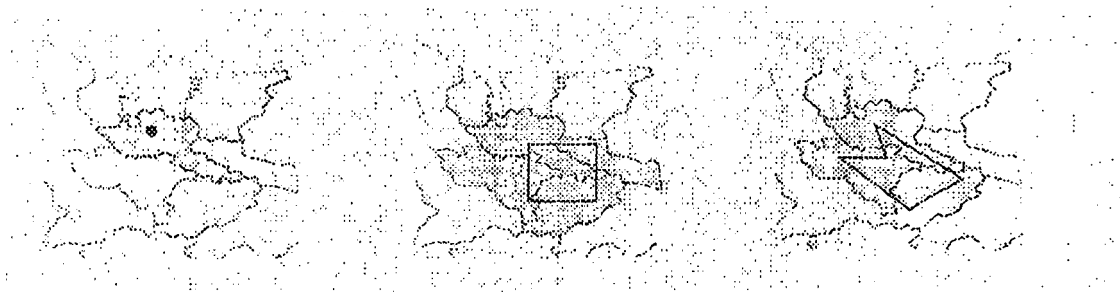
- Δεδομένου ενός αντικειμένου ερώτησης, εντόπισε αντικείμενα:
  - ✓ που έχουν ίδια έκταση με το δοσμένο αντικείμενο.
  - ✓ που έχουν τουλάχιστον ένα κοινό σημείο με το δοσμένο αντικείμενο.
  - ✓ που περιέχουν το δοσμένο αντικείμενο.
  - ✓ που περιέχονται στο δοσμένο αντικείμενο.
  - ✓ γειτονικά προς το δοσμένο αντικείμενο.
  - ✓ σε ελάχιστη απόσταση από το δοσμένο αντικείμενο.
- Δεδομένου ενός σημείου, εντόπισε αντικείμενα που το περιέχουν .
- Δεδομένης μιας περιοχής στο χώρο, εντόπισε αντικείμενα που την περιέχουν .

➢ Δεδομένης μιας πρότασης σχετικά με τις θέσεις αντικειμένων, διαπίστωσε αν είναι αληθής ή ψευδής.

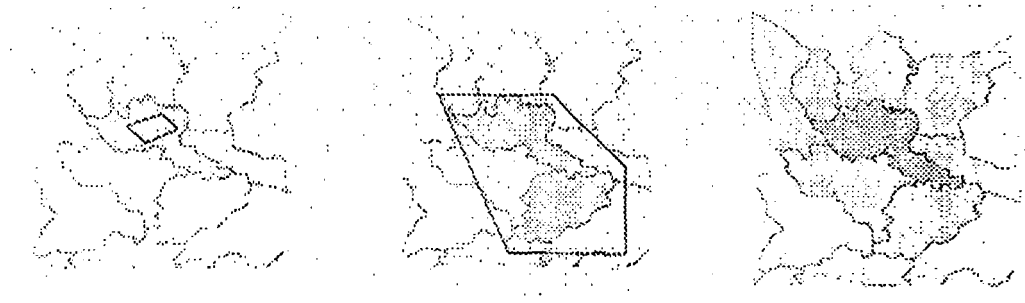
Τέλος, αναφέρουμε κάποιες από τις βασικές πράξεις που συναντώνται συχνότερα στα πλαίσια επεξεργασίας σύνθετων δεδομένων .

- ✓ τέμνει (intersects)
- ✓ περιέχει contains)
- ✓ εμπεριέχεται (is_enclosed_by)
- ✓ βρίσκεται σε δοσμένη απόσταση (distance Θq)
- ✓ βορειοδυτικά (northwest)
- ✓ γειτονεύει (adjacent)
- ✓ έχει ένα κοινό σημείο /ακμή (meets)

Ακολουθούν ενδεικτικά παραδείγματα απλών και συνδυαστικών ερωτήσεων προς βάσεις σύνθετων αντικειμένων.



Ερώτηση για σημείο Ερώτηση για περιοχή Ερώτηση τομής



Ερώτηση «περιέχεται» Ερώτηση «περιέχει» Ερώτηση γειτνίασης

2.5 Βασικές Δομές Δεδομένων

Οι κλασσικές μέθοδοι προσπέλασης σε μια διάσταση αποτελούν θεμελιώδες στοιχείο για όλες σχεδόν τις μεθόδους προσπέλασης για πολλαπλές διαστάσεις. Οι πιο κοινές μέθοδοι προσπέλασης σε μια διάσταση είναι ο Γραμμικός και Εκτεταμένος Κατακερματισμός (hashing) και τα B-Trees. Οι ιεραρχικές μέθοδοι προσπέλασης

όπως τα B-Trees διαθέτουν διαβαθμισιμότητα, έχουν καλή συμπεριφορά για διασκορπισμένα δεδομένα εισόδου και είναι σχεδόν ανεξάρτητες από την κατανομή της εισόδου. Αντίθετα, η συμπεριφορά μεθόδων που βασίζονται στο κατακερματισμό επηρεάζεται σε μεγάλο βαθμό από τα δεδομένα εισόδου και από τη συνάρτηση . Ο δεύτερος παράγοντας περιορίζεται σημαντικά με τη χρήση συναρτήσεων κατακερματισμού που διατηρούν τις σχέσεις διάταξης και γειτονίας των δεδομένων εισόδου με αποτέλεσμα ακόμα και στην περίπτωση πολύ διασκορπισμένων δεδομένων, αυτά να συγκεντρώνονται σε ένα σχετικά συνεκτικό χώρο.

2.5.1 Μέθοδοι Προσπέλασης Σε Μια Διάσταση

2.5.1.1 Γραμμικός Κατακερματισμός

Ο χώρος διασπάται σε περιοχές προκαθορισμένου μεγέθους που καλούνται buckets (σελίδες δίσκου/κάδος) και καταλαμβάνουν μία σελίδα στο δίσκο. Όταν γεμίζει ένα bucket διασπάται το ίδιο σε δύο νέα ισομεγέθη.

Το πρόβλημα με την τεχνική αυτή είναι ότι οι διασπάσεις ενδεχομένως οδηγούν σε υπερχείλιση των διαθέσιμων σελίδων του δίσκου με αποτέλεσμα να γίνεται υπερβολική σπατάλη αναγκαίου αποθηκευτικού χώρου. Το πρόβλημα αντιμετωπίζεται με το Εκτεταμένο κατακερματισμό.

2.5.1.2 Εκτεταμένος Κατακερματισμός

Ο χώρος διασπάται σε κελιά ενώ χρησιμοποιείται ένας κεντρικός κατάλογος κάθε στοιχείο του οποίου αποτελεί δείκτη σε ένα κελί. Υπερχείλιση σε κάποιο κελί έχει σαν αποτέλεσμα τη διάσπαση όλων των κελιών σε δύο νέα ισομεγέθη. Με τον τρόπο αυτό γίνεται προσπάθεια αντιμετώπισης της υπερχείλισης των διαθέσιμων σελίδων του δίσκου.

Το βασικά προβλήματα που εμφανίζει η μέθοδος είναι αφενός η κακή (ελάχιστη) χρησιμοποίηση του καταλόγου και αφετέρου η ανεξέλεγκτη αύξηση του καταλόγου λόγω των βημάτων διπλασιασμού. Η αντιμετώπιση γίνεται με το φραγμένου καταλόγου εκτεταμένο κατακερματισμό σύμφωνα με το οποίο ο κατάλογος επιτρέπεται να αυξάνεται μέχρι ένα προκαθορισμένο όριο. Όταν ξεπεραστεί το όριο αυτό, ο κατάλογος δεν τροποποιείται ενώ επέρχεται αύξηση στο χώρο δεδομένων .

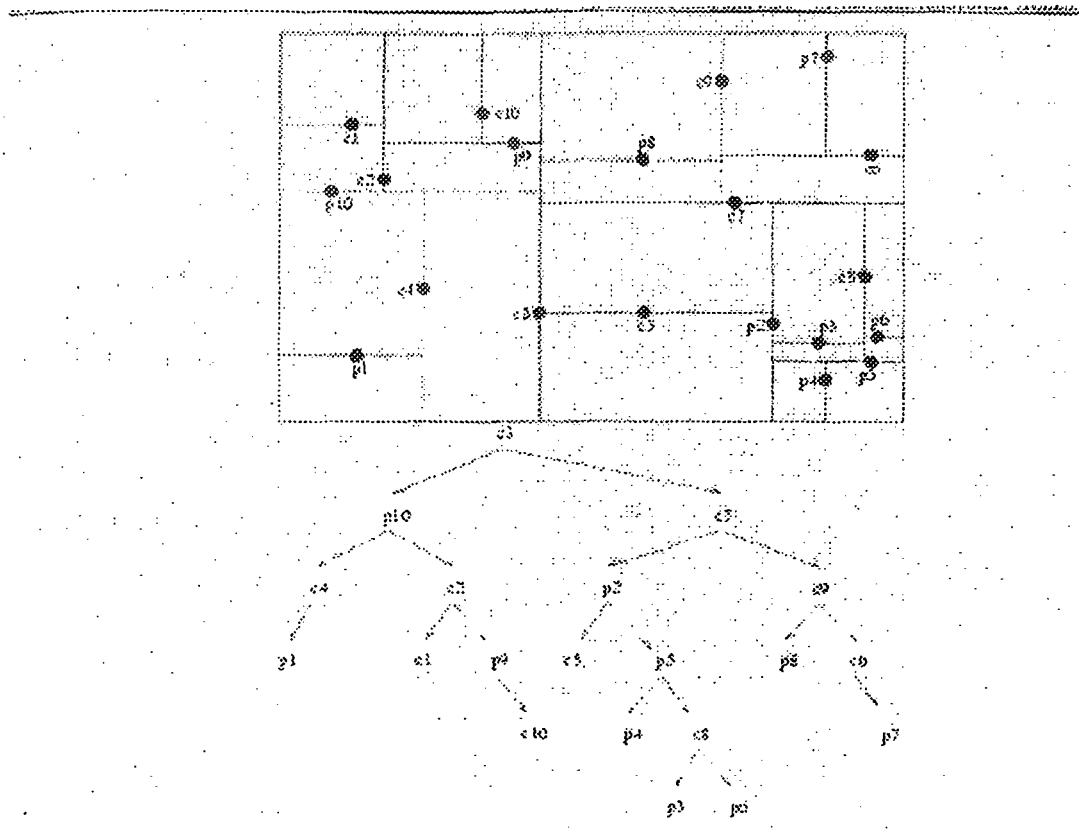
2.5.1.3. Το B-Tree

Πρόκειται για ζυγισμένο δυαδικό δένδρο αναζήτησης. Τα δεδομένα οργανώνονται με ιεραρχικό τρόπο. Τα φύλλα και ενδεχομένως οι εσωτερικοί κόμβοι περιέχουν δείκτες. Κάθε κόμβος αντιστοιχεί σε μία σελίδα του δίσκου και έχει περιορισμένο αριθμό παιδιών, με αποτέλεσμα να γίνεται αποδοτική χρησιμοποίηση του διαθέσιμου αποθηκευτικού χώρου. Υπερχείλιση σε έναν κόμβο συνεπάγεται διπλασιασμό του. Παρά τα καλά χαρακτηριστικά του δυαδικού δένδρου αναζήτησης, για ομοιόμορφα κατανεμημένα δεδομένα ο κατακερματισμός (hashing) έχει κατά μέσο όρο καλύτερη απόδοση.

2.6 Δομές Για Την Κύρια Μνήμη

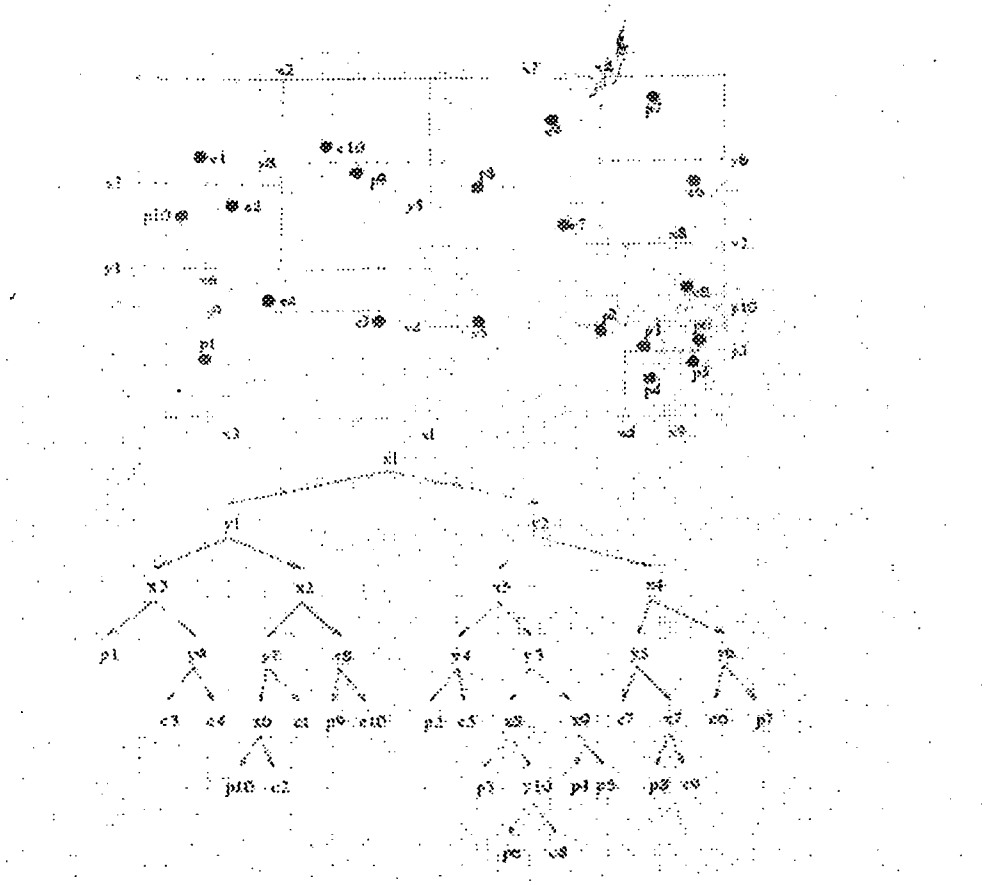
2.6.1 Το K-D-Tree

Πρόκειται για d-διάστατο, δυαδικό δένδρο εύρεσης που χρησιμοποιεί σημεία σα δεδομένα. Η διαχείριση του προς επεξεργασία χώρου γίνεται με την αναδρομική διάσπασή του σε υποχώρους μέσω υπερεπιπέδων d-l διαστάσεων .Με τον τρόπο αυτό λειτουργίες όπως εισαγωγή και εύρεση πραγματοποιούνται άμεσα χωρίς επιπλέον πολυπλοκότητα, γεγονός που ενδεχομένως δεν ισχύει για τη διαγραφή όπου και απαιτείται αναδιοργάνωση του υποδένδρου κάτω από σημείο που διαγράφεται. Στη συνέχεια παρουσιάζεται ένας χώρος που θέλουμε να επεξεργαστούμε. Στο χώρο αυτό έχουν σημειωθεί τα σημεία που μας ενδιαφέρουν. Πιο κάτω παραθέτουμε το k-d-tree για το στιγμιότυπο αυτό.



K-D-Tree

Ένα βασικό μειονέκτημα της συγκεκριμένης δομής είναι η ευαισθησία της στη σειρά εισαγωγής των δεδομένων. Επίσης μειονέκτημα αποτελεί το γεγονός ότι τα σημεία είναι διασκορπισμένα σε όλο το δένδρο. Αντιμετώπιση σε αυτά τα προβλήματα προτείνεται κατ' αρχήν μέσω του προσαρμοστικού k-d-tree το οποίο προσπαθεί να είναι κατά το μέγιστο δυνατό ισοζυγισμένο επιλέγοντας πιο ευέλικτη στρατηγική για τις διασπάσεις. Η τεχνική λειτουργεί καλά όταν τα σημεία (δεδομένα) είναι γνωστά εξ' αρχής και σπάνια γίνονται ενημερώσεις και αλλαγές στα δεδομένα.



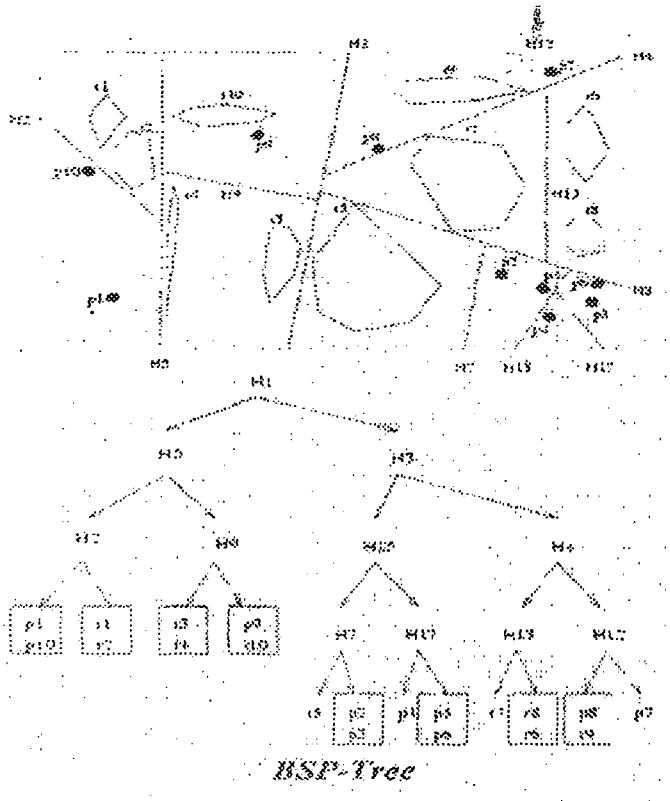
Adaptive K-D-Tree

Μια εναλλακτική προσέγγιση αποτελεί το bintree που είναι ουσιαστικά παραλλαγή του k-d-tree. Τώρα ο χώρος χωρίζεται σε ισομεγέθη κουτιά d διαστάσεων έτσι ώστε καθένα από αυτά να περιέχει φραγμένο αριθμό σημείων . Γενικά, πάντως, το βασικό πρόβλημα όλων των k-d-trees είναι ότι για συγκεκριμένες κατανομές των δεδομένων εισόδου δεν υπάρχει διάσπαση που να ισοκατανέμει τα σημεία.

2.6.2 Το BSP-Tree

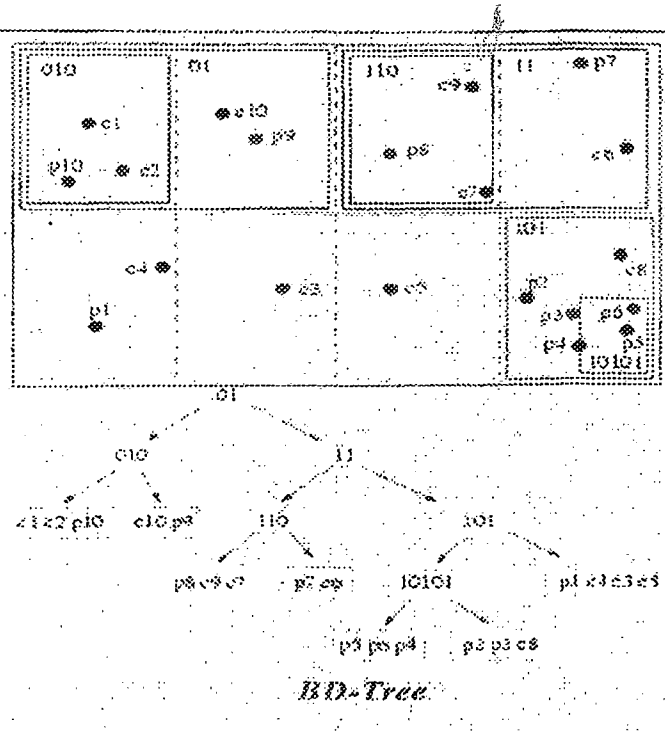
Τα BSP είναι δυαδικά δένδρα και αποτελούν προσπάθεια αντιμετώπισης του βασικού προβλήματος των k-d trees που αναφέρθηκε παραπάνω. Ο χώρος διασπάται σε όμοια επίπεδα ομοιόμορφα κατανεμημένα ενώ η χρησιμοποίηση τυχαιότητας αυξάνει την πιθανότητα να βρεθεί μια καλή διάσπαση. Η αναδρομική διάσπαση συνεχίζεται μέχρι τα αντικείμενα σε όλους τους υποχώρους να μην ξεπερνούν έναν προκαθορισμένο αριθμό.

Τα BSP- Trees προσαρμόζονται καλά σε διαφορετικές κατανομές των δεδομένων εισόδου. Παρολαυτά, επειδή είναι μη ζυγισμένα δένδρα, ενδέχεται να έχουν πολύ βαθιά υποδένδρα γεγονός που συνεπάγεται κακή επίδραση στην απόδοσή τους και πιθανώς μεγάλες απαιτήσεις σε αποθηκευτικό χώρο.



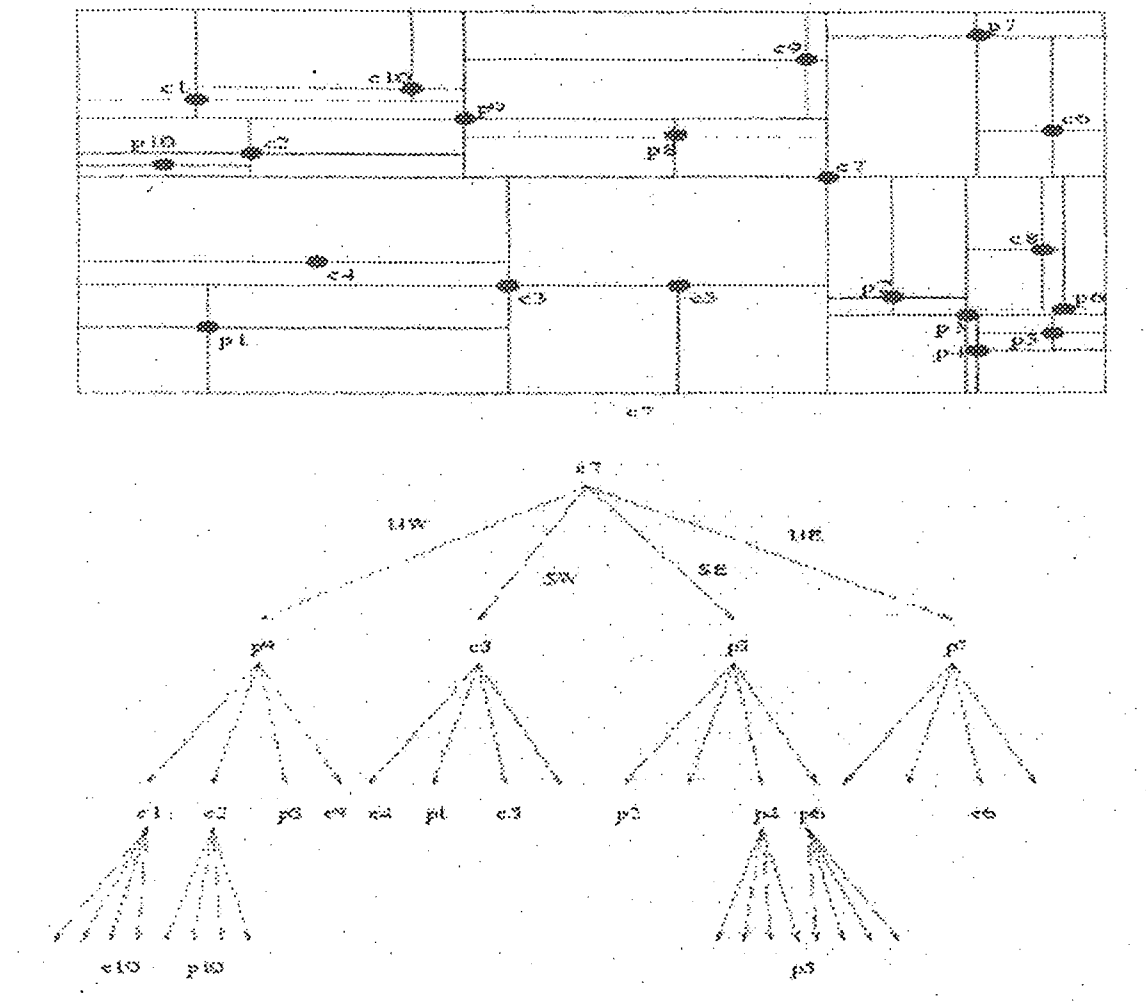
2.6.3 Το BD-Tree

Πρόκειται για δενδρική δομή επεξεργασίας σύνθετων δεδομένων. Χρησιμοποιούνται κώδικες Peano για την κωδικοποίηση των υποχώρων του προς επεξεργασία χώρου σε συμβολοσειρές που αποτελούνται από 0 και 1. Για το λόγο αυτό τα BD-Trees αναφέρονται και σαν δένδρα που χρησιμοποιούν Bit-Interleaving: Στις συμβολοσειρές που κωδικοποιούν τους υποχώρους, οι περιττές θέσεις αντιστοιχούν σε κάθετες γραμμές ενώ οι άρτιες θέσεις σε οριζόντιες γραμμές. Τα BD- Trees υλοποιούν μια ευέλικτη πολιτική όσον αφορά στον τρόπο που πραγματοποιούνται οι αναγκαίες διασπάσεις αποφεύγοντας έτσι προβλήματα υπό- χρησιμοποίησης του διαθέσιμου αποθηκευτικού χώρου.

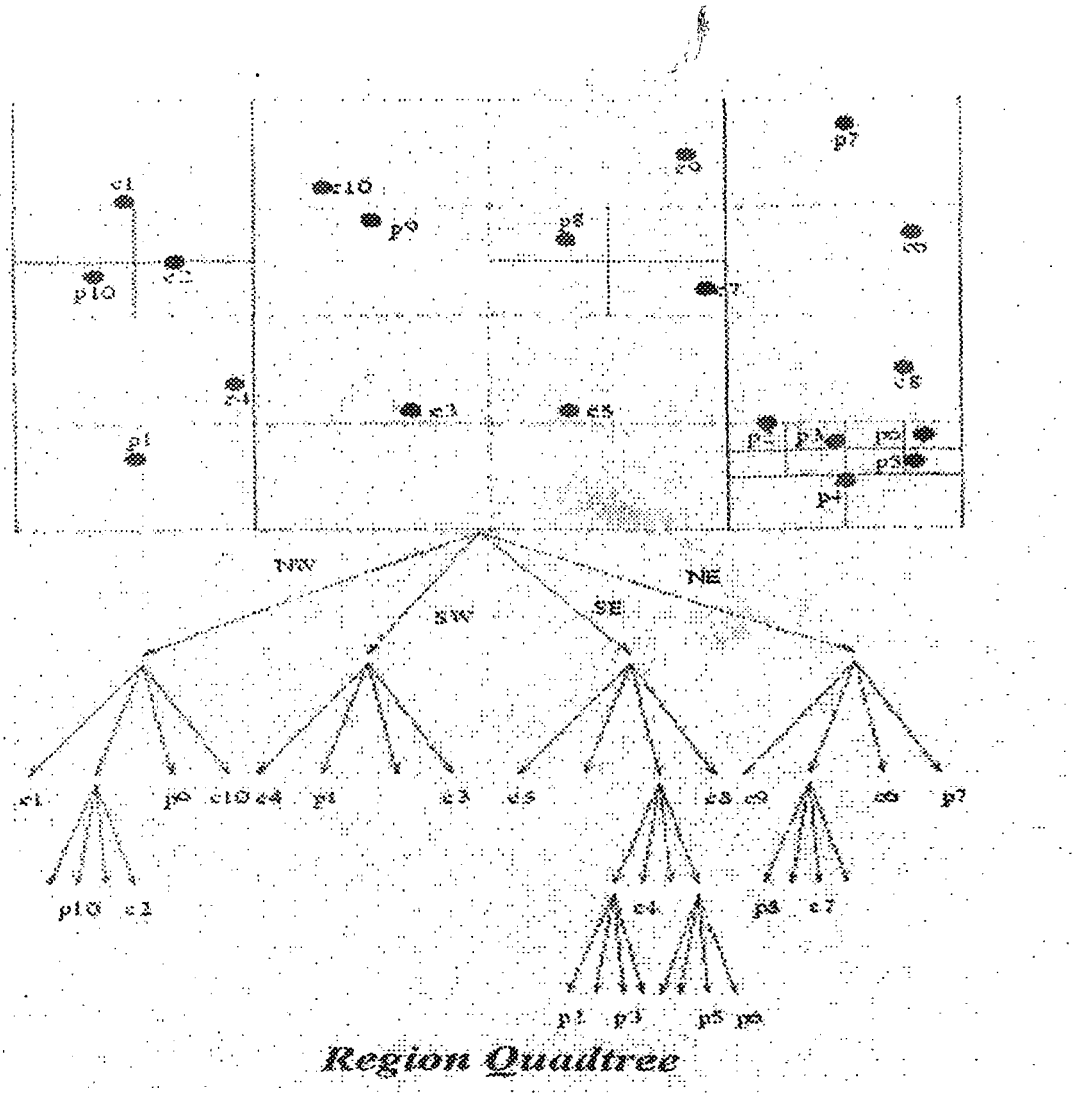


2.6.4 Το QuadTree

Δενδρική δομή παρόμοια με τα k-d- Trees με τη βασική διαφορά ότι δεν πρόκειται πλέον για δυαδικά δένδρα. Κάθε εσωτερικός κόμβος έχει 2^d παιδιά όχι απαραίτητα ισομεγέθη. Επιπλέον, τα Quadtrees δεν είναι κατ' ανάγκη ζυγισμένα δένδρα. Δύο από τις βασικότερες παραλλαγές των Quatrees είναι τα point quadtrees, που ουσιαστικά είναι πολλαπλών διαστάσεων δυαδικά δένδρα αναζήτησης και τα region quadtrees που υιοθετούν μια κανονικοποιημένη διάσπαση του προς επεξεργασία χώρου.



Point Quadtree



2.7 Μέθοδοι Προσπέλασης Σημείου

Οι μέθοδοι που παρουσιάστηκαν μέχρι τώρα είναι σχεδιασμένες για τη διαχείριση της κύριας μνήμης και θεωρούν δεδομένο ότι όλη η προς επεξεργασία πληροφορία είναι αποθηκευμένη στην κύρια μνήμη. Ως εκ τούτου δε λαμβάνουν υπόψη τους το δευτερεύοντα αποθηκευτικό χώρο που υφίσταται στην πραγματικότητα. Φυσικά, οι μέθοδοι προσπέλασης της κύριας μνήμης μπορούν να χρησιμοποιηθούν και για τη δευτερεύουσα έχοντας όμως πολύ κακή απόδοση. Για το λόγο αυτό έχουν σχεδιαστεί μέθοδοι προσπέλασης που λαμβάνουν υπόψη και δευτερεύουσα μνήμη, οι οποίες λειτουργούν σε συντονισμό με το λειτουργικό σύστημα και βελτιστοποιούν την απόδοση. Στη συνέχεια, παρουσιάζονται τέτοιες μέθοδοι οι οποίες επιπλέον είναι προσανατολισμένες στην ειδική κατηγορία ερωτήσεων που λέγονται ερωτήσεις προσπέλασης σημείου (point access methods). Η βάση δεδομένων περιέχει σημεία οργανωμένα σε ομάδες που καλούνται buckets (σελίδες δίσκου), καθένα από τα οποία αντιστοιχεί σε μία σελίδα του δίσκου και απεικονίζει ένα τμήμα του προς επεξεργασία χώρου.

2.7.1 Κατακερματισμός Πολλαπλών Διαστάσεων

Βασικός στόχος όλων των μεθόδων που εμπίπτουν στην κατηγορία αυτή είναι η διατήρηση των σχέσεων γειτονίας και γενικότερα των σχετικών θέσεων των αντικειμένων στον προς επεξεργασία χώρο όταν αυτά μεταφέρονται σε μια βάση σύνθετων δεδομένων .Με άλλα λόγια, αντικείμενο που είναι κοντά στο χώρο πρέπει να αποθηκεύονται και κοντά. Η υιοθέτηση αυτής της πολιτικής έχει σαν αποτέλεσμα την ελαχιστοποίηση των προσπελάσεων στο δίσκο κι επομένως την επιτάχυνση της επεξεργασίας.

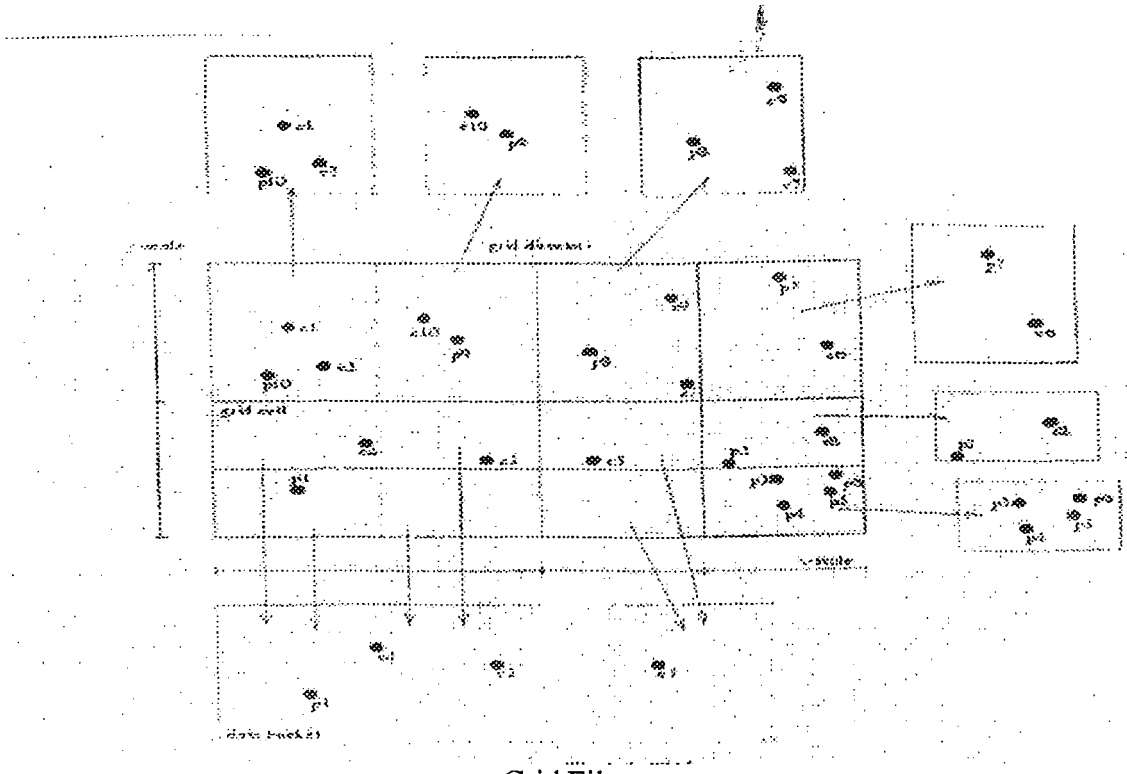
Με βάση την παραπάνω ιδέα έχουν αναπτυχθεί μέθοδοι που βασίζονται στο Εκτεταμένο Κατακερματισμό (Hashing), όπως είναι το Grid File, το EXCELL, το Grid File 2 επιπέδων και το Twin Grid File.

Μια οικογένεια μεθόδων που βασίζεται στο γραμμικό κατακερματισμό είναι ο καλούμενος πολυδιάστατος γραμμικός κατακερματισμός. Τέλος, έχουν αναπτυχθεί και υβριδικές μέθοδοι προσπέλασης σημείου με στόχο το συνδυασμό και την εκμετάλλευση των καλών χαρακτηριστικών των δύο παραπάνω κλάσεων μεθόδων, όπως το BANG File και το BUDDY Tree.

2.7.2. Το Grid File

Χρησιμοποιείται κατάλογος και ο χώρος που θέλουμε να επεξεργαστούμε χωρίζεται με την μορφή πλέγματος όχι απαραίτητα σε ισομεγέθη κελιά. Η μέθοδος βασίζεται στον κατακερματισμό με την έννοια ότι τα κελιά ομαδοποιούνται σε buckets που αντιστοιχούν σε μία σελίδα του δίσκου. Ο κατάλογος αποθηκεύεται στην δευτερεύουσα μνήμη, ενώ το πλέγμα στην κύρια μνήμη, γεγονός που έχει σαν αποτέλεσμα τα δεδομένα να προσδιορίζονται με δύο το πολύ προσπελάσεις στον δίσκο.

Το βασικό πρόβλήμα της μεθόδου είναι η ενδεχόμενη υπερεπέκταση του καταλόγου ακόμα και όταν τα δεδομένα είναι ομοιόμορφα κατανεμημένα.



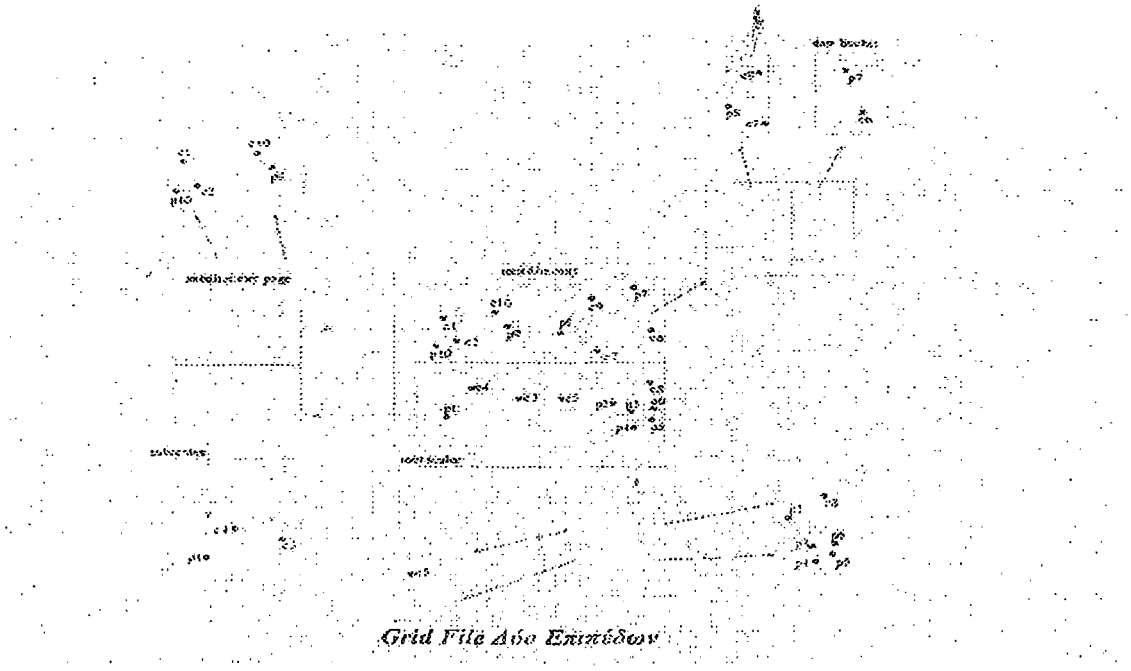
Grid File

2.7.3 EXCELL

Η μέθοδος είναι πολύ σχετική με την προηγούμενη με τη διαφορά ότι ο χώρος που Θέλουμε να επεξεργαστούμε χωρίζεται τώρα σε ομοιόμορφα τμήματα αποσκοπώντας σε περιορισμένη αύξηση του καταλόγου με μικρή ίσως αύξηση του αριθμού των αχρησιμοποίητων buckets.

2.7.4 Grid File Δύο Επιπέδων

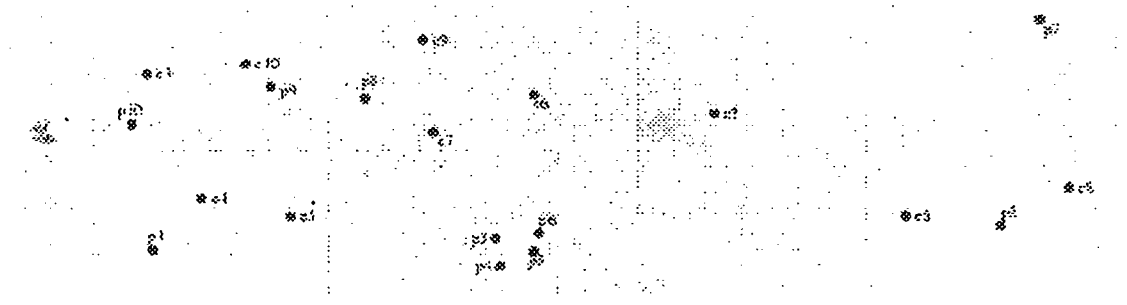
Η βασική ιδέα είναι η χρησιμοποίηση ενός δεύτερου grid file για την διαχείριση του καταλόγου που χωρίζεται σε δύο επίπεδα. Οι είσοδοι στο πρώτο επίπεδο καταλόγου αποτελούν δείκτες στα στοιχεία του δευτέρου επιπέδου τα οποία με την σειρά τους αποτελούν δείκτες σε δεδομένα. Στόχο αποτελεί πάλι η περιορισμένη αύξηση του καταλόγου. Το δεύτερο επίπεδο του grid file κρατιέται στην κύρια μνήμη.



Grid File Δύο Επιπέδων

2.7.5 Δίδυμα Grid Files

Στόχος της μεθόδου είναι η αύξηση του ποσοστού χρησιμοποίησης του διαθέσιμου αποθηκευτικού χώρου. Τα δύο grid files που χρησιμοποιούνται «βλέπουν» όλο τον προς επεξεργασία χώρο ενώ η κατανομή των δεδομένων μεταξύ τους γίνεται με δυναμικό τρόπο στοχεύοντας στην ελαχιστοποίηση του συνολικού απαραίτητου αριθμού σελίδων δίσκου. Η μέθοδος εμφανίζει πολύ καλή απόδοση όταν ο χώρος για επεξεργασία είναι μεγάλος.



Δεν είναι απαραίτητη η χρησιμοποίηση καταλόγου και στην περίπτωση που χρησιμοποιείται έχει πολύ μικρό μέγεθος. Η μέθοδος έχει πολύ μικρές απαιτήσεις σε αποθηκευτικό χώρο και κρατάει όλη τη σχετική πληροφορία στην κύρια μνήμη. Έχει αναπτυχθεί πληθώρα ντετερμινιστικών και πιθανοτικών αλγορίθμων για τον αποδοτικό υπολογισμό των διευθύνσεων για την προσπέλαση των δεδομένων. Πάντως η μέθοδος δεν παρουσιάζει εξαιρετική απόδοση ανεξάρτητα από την κατανομή των δεδομένων εισόδου.

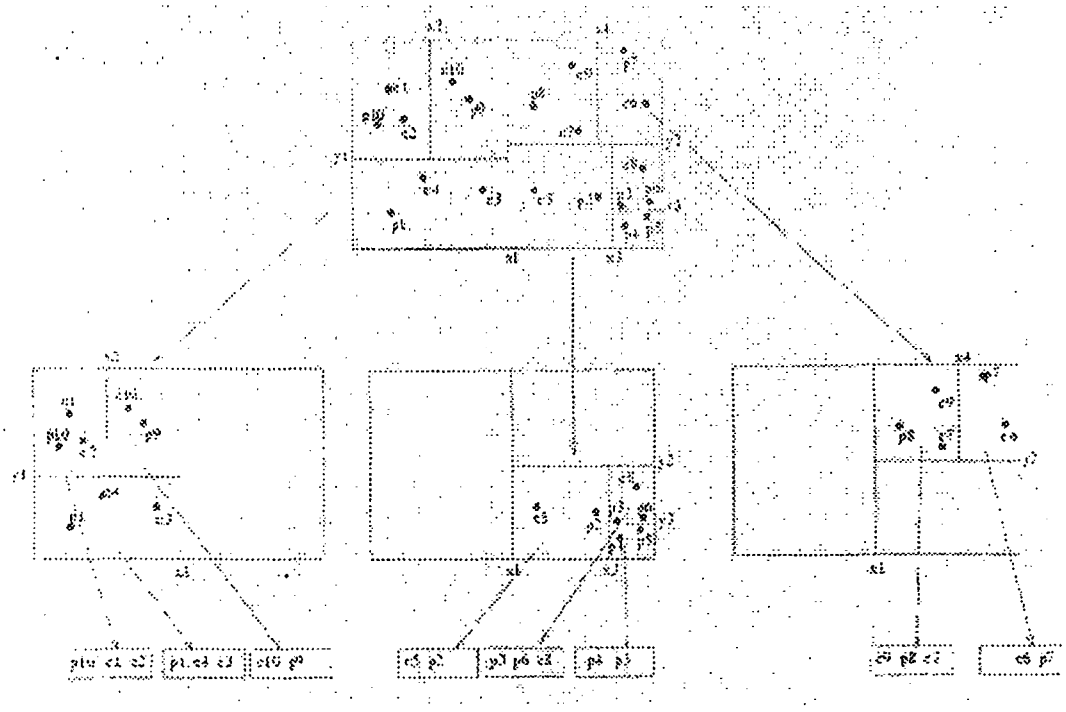
2.8 Ιεραρχικές Μέθοδοι Προσπέλασης

Παρουσιάζονται μέθοδοι προσπέλασης σημείου που βασίζονται σε δενδρικές δομές, όχι απαραίτητα δυαδικές. Τα σημεία οργανώνονται σε buckets, καθένα από τα οποία αντιστοιχεί σε ένα φύλλο του χρησιμοποιούμενου δένδρου και μία σελίδα του δίσκου. Εσωτερικοί κόμβοι λειτουργούν σαν καθοδηγητές για την αναζήτηση αντιστοιχώντας σε ένα μέρος του συνολικού χώρου με ιεραρχική δόμηση.

2.8.1. Το k-d-B-Tree

Συνδυάζει τις ιδιότητες του προσαρμοστικού k-d-tree και του B-Tree για τη διαχείριση σημείων πολλαπλών διαστάσεων .Περιοχές που αντιστοιχούν σε κόμβους ίδιου επιπέδου δεν έχουν κοινά σημεία μεταξύ τους ενώ η ένωσή τους δίνει το συνολικό χώρο που θέλουμε να επεξεργαστούμε.

Πρόκειται για πλήρως ζυγισμένο δένδρο που προσαρμόζεται καλά σε διαφορετικές κατανομές των δεδομένων εισόδου. Παρολαυτά δεν υπάρχει εγγύηση για βέλτιστη χρησιμοποίηση του διαθέσιμου χώρου.

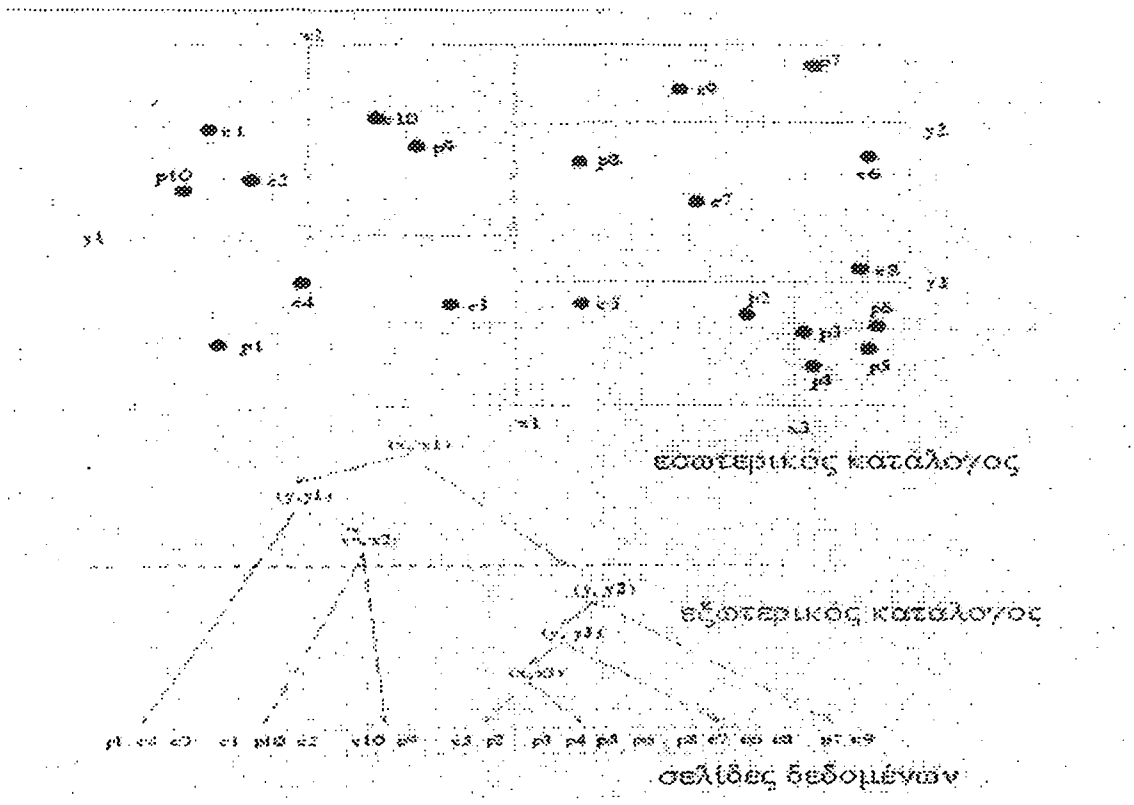


k-d-B-Tree

2.8.2 Το LSD-Tree

Παρουσιάζει καλή απόδοση ακόμα και σε περιπτώσεις μη ομοιόμορφης κατανομής των δεδομένων εισόδου. Περιέχει κατάλογο που ακολουθεί την οργάνωση ενός k-d-tree ενώ ο χώρος διασπάται σε ξένα μεταξύ τους κελιά διαφόρων μεγεθών .Διαθέτει ένα είδος εξωτερικής ζύγισης με την έννοια ότι τα εξωτερικά του υποδένδρα διαφέρουν το πολύ κατά ένα. Στην κύρια μνήμη κρατιέται το άκρως απαραίτητο

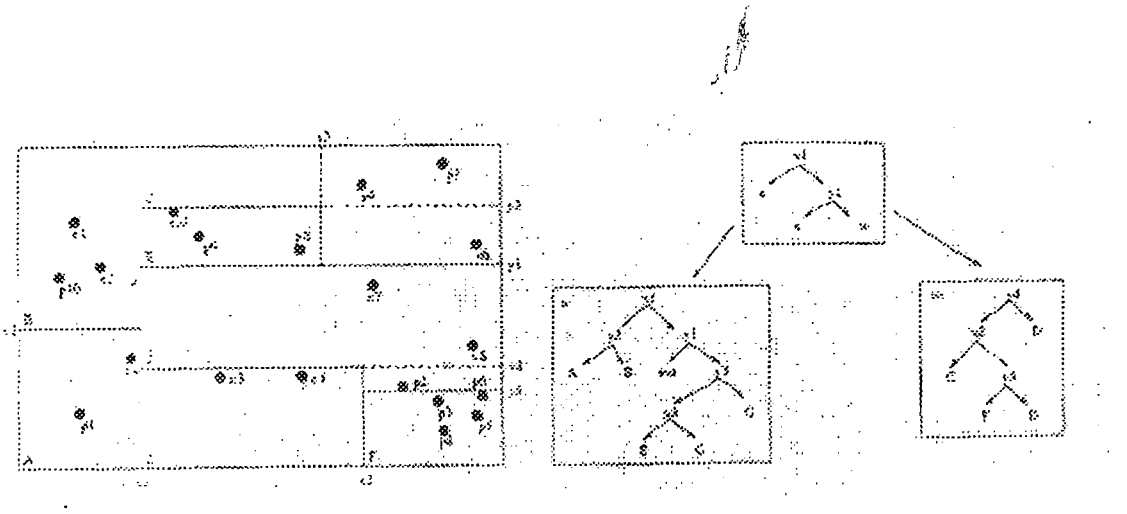
τμήμα της δομής ενώ κάποια άλλα κομμάτια της αποθηκεύονται στη δευτερεύουσα μνήμη γεγονός που κάνει δύσκολη και μη αποδοτική την ενσωμάτωση της δομής αυτής σε μια βάση δεδομένων γενικού σκοπού.



LSD-Tree

2.8.3. Το Buddy Tree

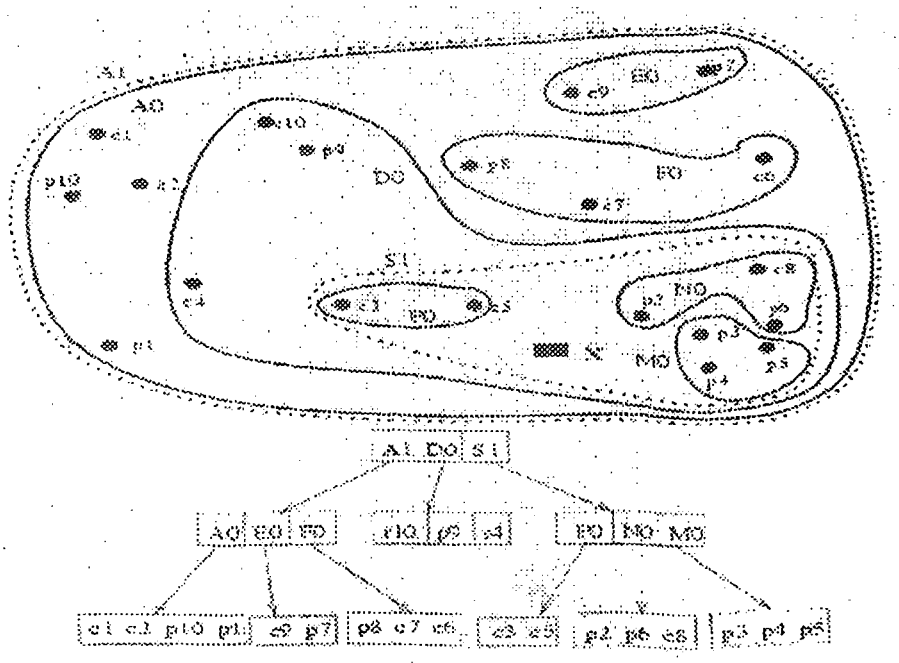
Βασίζεται στο δυναμικό κατακερματισμό και διατηρεί έναν κατάλογο δομημένο με τη μορφή δένδρου. Ο χώρος χωρίζεται με χρήση k-d-trees αναδρομικά σε δύο ανισομεγέθη τμήματα και κάθε εσωτερικός κόμβος αντιστοιχεί σε ένα d-διάστατο τμήμα του συνολικού χώρου. Κόμβοι στο ίδιο επίπεδο του δέντρου αντιστοιχούν σε περιοχές ξένες μεταξύ τους ενώ τα φύλλα του καταλόγου είναι δείκτες σε σελίδες δεδομένων. Το δέντρο δεν είναι απαραίτητα ζυγισμένο (τα φύλλα βρίσκονται ενδεχομένως σε διαφορετικά επίπεδα). Αποφεύγεται η υπερβολική αύξηση του καταλόγου ο οποίος αποτελεί καλό οδηγό για τον προσδιορισμό των δεδομένων . Η δομή είναι ιδιαίτερα ευέλικτη σε περιπτώσεις αλλαγής των καταχωρημένων στοιχείων.



hB-Tree

2.8.6 Το BV-Tree

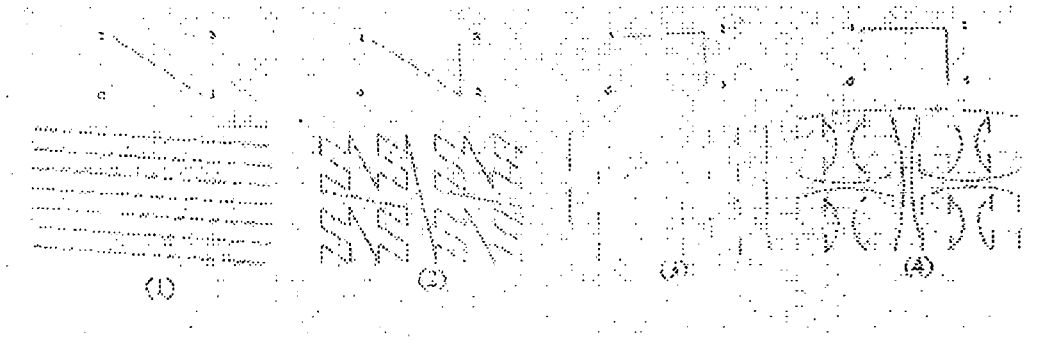
Αποτελεί προσπάθεια για την γενίκευση του B-tree σε πολλές διαστάσεις χαλαρώνοντας τις απαιτήσεις ζύγισης και βέλτιστης χρησιμοποίησης του αποθηκευτικού χώρου. Μπορεί να ενσωματωθεί σε άλλες μεθόδους προσπέλασης. Για την απαίτηση σε μια ερώτηση προσδιορίζονται όλες οι πιθανές απαντήσεις και επιλέγεται αυτή που προσεγγίζει περισσότερο την σωστή.



BV-Tree

2.9 Space Filling Curves Για Δεδομένα Με Τη Μορφή Σημείων

Οι μέθοδοι προσπέλασης σε πολλές διαστάσεις σχεδιάζονται προκειμένου να αντιμετωπιστεί το πρόβλημα της μη διάταξης των δεδομένων και της μη διατήρησης των σχέσεών γειτνίασής τους. Μια ιδέα είναι η χρησιμοποίηση ευρετικών μεθόδων διάταξης των δεδομένων εισόδου κατά τέτοιον τρόπο ώστε στοιχεία που βρίσκονται κοντά στον αρχικό χώρο να βρίσκονται κοντά και στην απεικόνισή τους στη δομή που χρησιμοποιείται για την επεξεργασία. Ο χώρος χωρίζεται με τη μορφή πλέγματος του οποίου τα κελιά έχουν ετικέτες (αριθμούς) που προσδιορίζουν τη θέση τους με μοναδικό τρόπο στο συνολικό χώρο αποδίδοντας έτσι μια συνολική διάταξη στα σημεία του χώρου που θέλουμε να επεξεργαστούμε. Προκύπτουν έτσι καμπύλες που καλύπτουν όλο το χώρο και γι ' αυτό το λόγο ονομάζονται και space filling curves. Ένα βασικό πλεονέκτημα των καμπυλών κάλυψης του χώρου είναι η ανεξαρτησία τους από το πλήθος των χρησιμοποιούμενων διαστάσεων αφού τα πάντα απεικονίζονται κωδικοποιημένα στο επίπεδο. Εντούτοις. ενδέχεται να είναι αναγκαίος ο επανυπολογισμός ίδιων πραγμάτων σε περιπτώσεις πολύπλοκης μορφής του υπό επεξεργασία χώρου. Μερικές γνωστές καμπύλες κάλυψης του χώρου παρουσιάζονται στα ακόλουθα σχήματα.



2.9.1 Σύνθετες Μέθοδοι Προσπέλασης

Καμία από τις μεθόδους προσπέλασης με πολλαπλές διαστάσεις που ήδη παρουσιάστηκαν δεν είναι δυνατόν να εφαρμοστεί απευθείας σε βάσεις δεδομένων που περιέχουν σύνθετα αντικείμενα. Για αυτές τις περιπτώσεις χρησιμοποιούνται οι μέθοδοι προσπέλασης σημείου τροποποιημένες με μία από τις ακόλουθες τεχνικές:

1. Μετασχηματισμός
2. Επικαλυπτόμενες περιοχές
3. Αντιγραφή
4. Πολλαπλά Επίπεδα

2.9.2 Μετασχηματισμός

Δύο είναι οι δυνατές επιλογές:

1. μετατροπή κάθε αντικειμένου σε ένα σημείο πολλαπλών διαστάσεων ,

2. μετατροπή κάθε αντικειμένου σε μια κωδικοποιημένη ισοδύναμη μορφή του στο επίπεδο μέσω καμπυλών κάλυψης του χώρου.

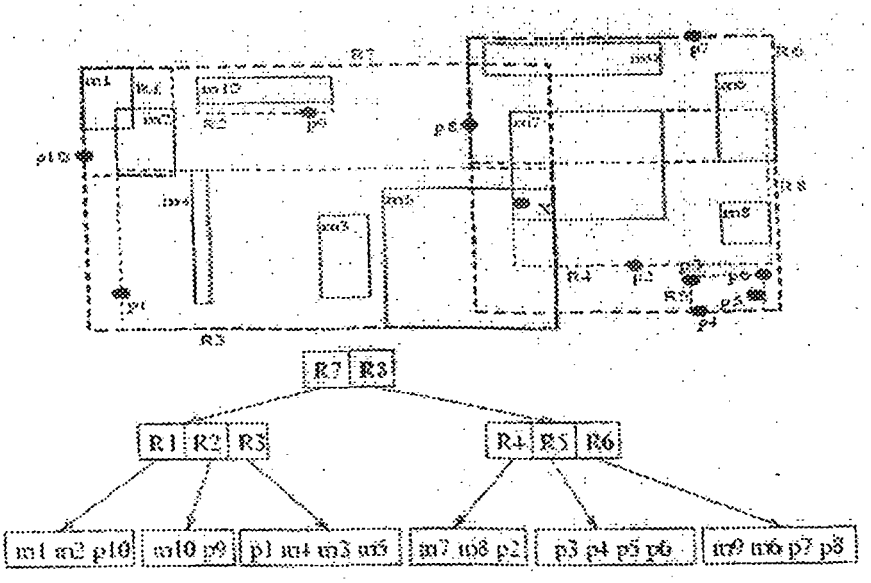
2.9.3. Επικαλυπτόμενες Περιοχές

Η βασική ιδέα της τεχνικής είναι η δυνατότητα διαφορετικά buckets δεδομένων να αντιστοιχούν σε αμοιβαία επικαλυπτόμενους υποχώρους. Η μέθοδος παρουσιάζει καλή απόδοση στην περίπτωση που η βάση περιέχει αντικείμενα με μεγάλο μέγεθος σχετικά με το μέγεθος του δοσμένου χώρου, π. χ. η βάση περιέχει κτίρια ή γεωγραφικά διαμερίσματα.

Ένα όχι σε μεγάλο βαθμό σημαντικό μειονέκτημα της μεθόδου είναι η αμφιλογία σχετικά με τη θέση εισαγωγής νέων στοιχείων στη βάση, για την αντιμετώπιση του οποίου υπάρχουν αρκετοί αλγόριθμοι. Μια ιδέα τέτοιων αλγορίθμων είναι γίνεται η εισαγωγή του νέου αντικειμένου με στόχο την εμφάνιση όσο το δυνατό λιγότερων νέων επικαλύψεων.

2.9.3.1. Το R-Tree

Το κίνητρο για την ανάπτυξή του αποτέλεσε η έλλειψη ευελιξίας των παραδοσιακών δομών δεδομένων (π.χ. B-Trees, πίνακες hash, κτλ.) για την περίπτωση εφαρμογών πολλαπλών διαστάσεων και σύνθετων αντικειμένων στο χώρο. Λαμβάνεται υπόψη η σελιδοποίηση για τη δευτερεύουσα μνήμη και η μέθοδος είναι ιδιαίτερα αποδοτική για τη διαχείριση σημείων .Χρησιμοποιείται δομή δεικτών για d-διάστατα αντικείμενα ανάλογη με αυτή του B+- Tree αλλά υιοθετείται διαφορετική αναπαράσταση των κόμβων. Για την οργάνωση του χώρου χρησιμοποιούνται παραλληλόγραμμα και διαφορετικά πολυδιάστατα γεωμετρικά σχήματα προσεγγίζονται από τα ελάχιστα παραλληλόγραμμα που τα περιέχουν .
Επομένως, παρά τη χρησιμοποίηση προσέγγισης, τα σημαντικά γεωμετρικά χαρακτηριστικά διατηρούνται, δηλαδή διατηρείται η θέση του αντικειμένου, η έκτασή του σε κάθε άξονα και σύνθετα αντικείμενα αναπαρίστανται με μικρό αριθμό bytes.



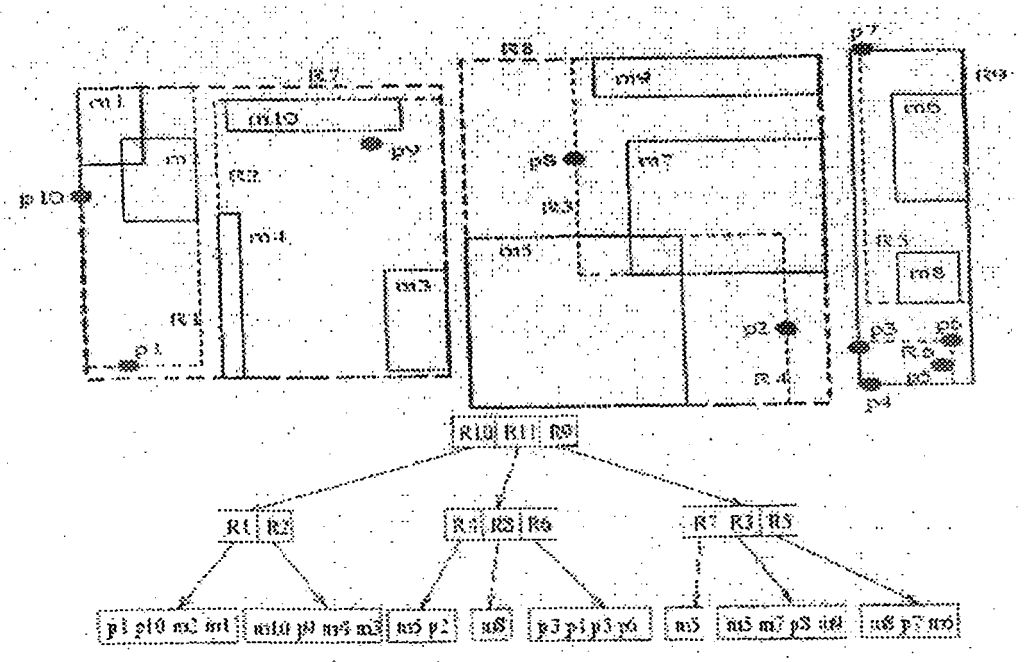
R-Tree

2.9.3.2. Το R*-Tree

Το κίνητρο για την ανάπτυξή του αποτέλεσε το γεγονός ότι η ευρετική προσέγγιση με R- Tree για τη βελτιστοποίηση μιας μόνο παραμέτρου δεν δίνει βέλτιστα αποτελέσματα. Επιπλέον η στατική επιλογή για MBR δεν είναι πάντα καλή για χώρους που μεταβάλλονται παρουσιάζοντας ενδεχομένως κακή απόδοση ανάκτησης. Οι βασικές ιδέες αντιμετώπισης των παραπάνω προβλημάτων είναι αφενός η ελαχιστοποίηση των περιθωρίων ή της επικάλυψης (να απαιτούνται δηλαδή λιγότερα μονοπάτια προσπέλασης) και αφετέρου η βέλτιστη αξιοποίηση του αποθηκευτικού χώρου. Στόχο δε αποτελεί η ταυτόχρονη υλοποίηση και των δύο παραπάνω ιδεών .

Τα βασικά χαρακτηριστικά ενός R*-Tree συνοψίζονται στα ακόλουθα:

- αναγκαστική επανεισαγωγή δεδομένων, κάτι που συνεπάγεται δυναμική επανοργάνωση του δένδρου
- λιγότερες επικαλύψεις και καλύτερη εκμετάλλευση του αποθηκευτικού χώρου
- λιγότερες διασπάσεις με την έννοια ότι απαιτείται ένα καλύτερα ζυγισμένο δένδρο και κατά συνέπεια λιγότερες λειτουργίες εισόδου εξόδου δεδομένων
- τετραγωνικός αριθμός παραλληλογράμμων καταλόγου που συνεπάγεται καλύτερη απόδοση

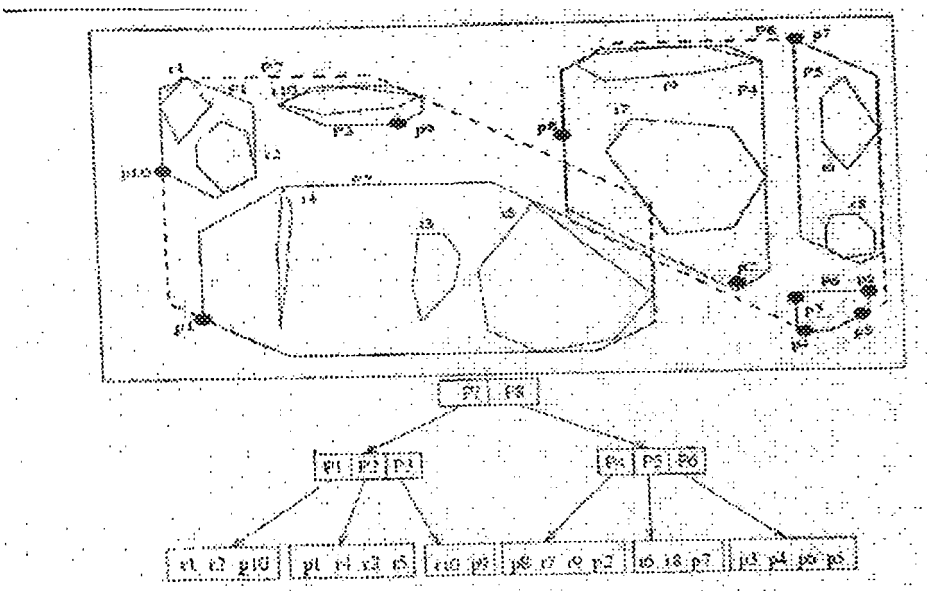


R*-Tree

2.9.3.3. Το P-Tree

Τα αντικείμενα προσεγγίζονται με ελάχιστα πολύτοπα που τα περιβάλλουν χωρίς να είναι αναγκαία η ανάλυση της ιδιαίτερης δομής τους. Χρησιμοποιείται R- Tree για

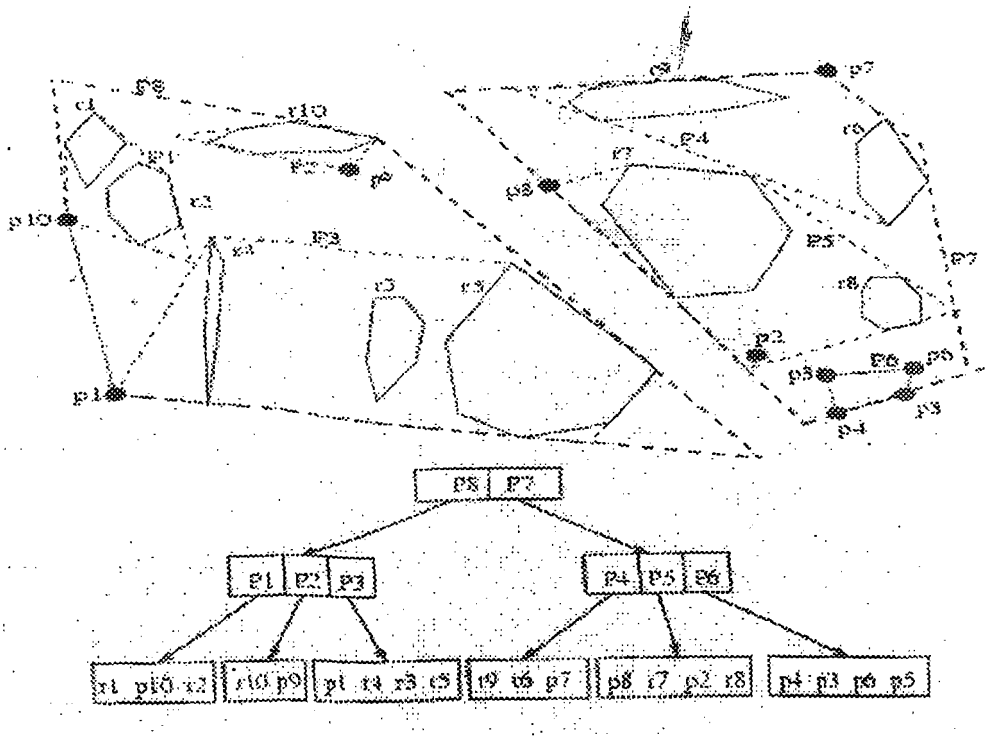
την ιεραρχική δόμηση των πολύτοπων που προκύπτουν και αντιστοιχούν στους εσωτερικούς τους κόμβους. Με τον τρόπο αυτό αυξάνεται το μέγεθος των εισόδων αλλά μειώνεται το πλήθος των απογόνων των εσωτερικών κόμβων .Πιθανό πρόβλημα της μεθόδου είναι η ενδεχόμενη κακή ποιότητα των προσεγγίσεων -για το οποίο έχουν προταθεί αρκετοί αλγόριθμοι -αλλά βασικό της πλεονέκτημα είναι η ανάγκη πολύ μικρού αποθηκευτικού χώρου.



P-Tree

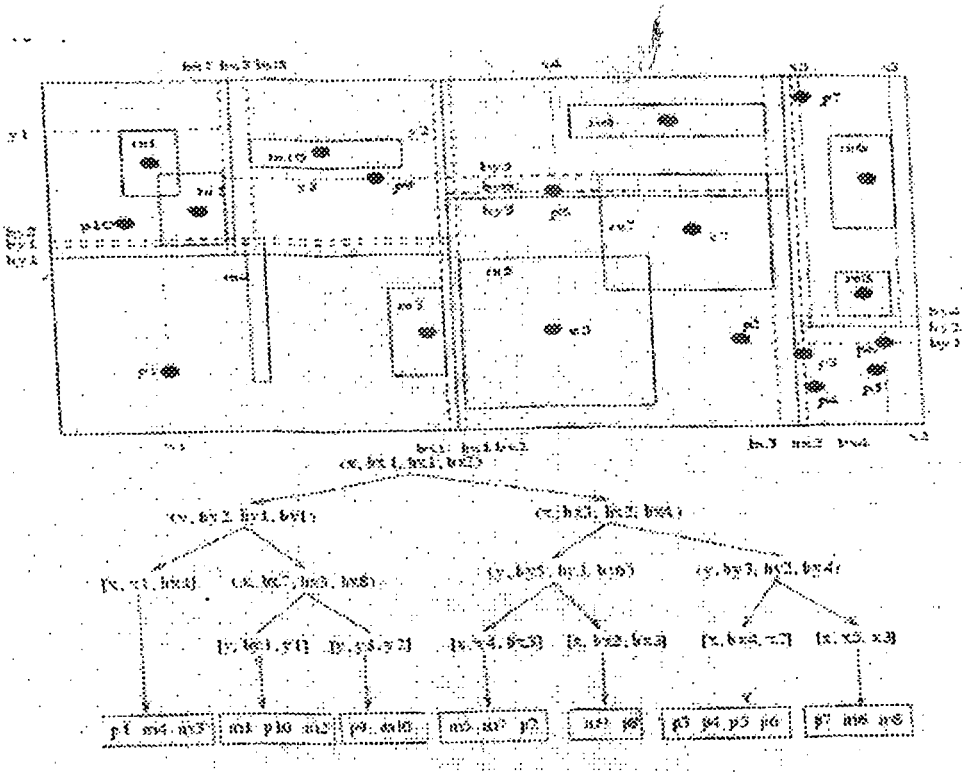
2.9.3.4. Το SP-Tree

Χρησιμοποιούνται πάλι πολύτοπα για την προσέγγιση των αντικειμένων αλλά μεταβάλλεται ο τρόπος αποθήκευσης και διαχείρισής τους. Γίνεται προσπάθεια συνδυασμού των καλών χαρακτηριστικών του δένδρου κελιών και του R*-Tree για την περίπτωση των δύο διαστάσεων. Το πλήθος των κορυφών του χρησιμοποιούμενου πολύτοπου δεν είναι φραγμένο και είναι ανάλογο της επιθυμητής ακρίβειας της προσέγγισης. Επειδή όμως μικρό πλήθος κορυφών προτιμάται για την αποδοτικότερη αποθήκευση, τα πεντάγωνα είναι αυτά που καλύπτουν και τις δύο παραπάνω απαιτήσεις με τον καλύτερο τρόπο.



2.9.3.5. Το SKD-Tree

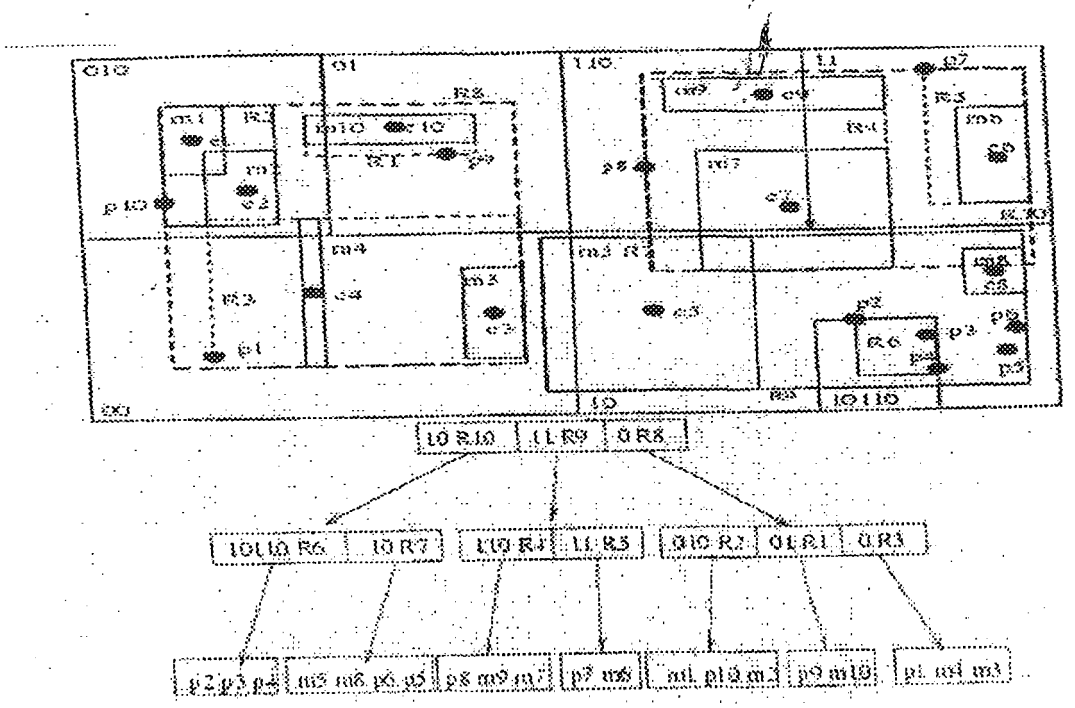
Πρόκειται για παραλλαγή του k-d-tree και χρησιμοποιείται για την αποθήκευση και διαχείριση σύνθετων αντικειμένων επιτρέποντας επικαλύψεις στα παραλληλόγραμμα που χρησιμοποιούνται για την προσέγγιση των αντικειμένων αυτών. Σε περίπτωση διαγραφής κάποιου αντικειμένου ενδέχεται να χρειαστεί να επανεισαχθούν όλα εκείνα τα αντικείμενα που έχουν κοινά σημεία με αυτό που διαγράφεται. Πάντως όσον αφορά στην απόδοση της δομής τόσο ως προς την αναζήτηση όσο και ως προς το ποσοστό χρησιμοποίησης του διαθέσιμου αποθηκευτικού χώρου, είναι παραπλήσια με αυτή των R- Trees.



SKD-Tree

2.9.3.6. Το GBD-Tree

Η δομή αποτελεί επέκταση των BD- Trees που λαμβάνει υπόψη της την ύπαρξη δευτερεύοντος αποθηκευτικού χώρου τον οποίο και διαχειρίζεται. Είναι ζυγισμένα δένδρα, όχι απαραίτητα δυαδικά, που αποθηκεύουν τα σύνθετα αντικείμενα με τη μορφή παραλληλογράμμων που τα προσεγγίζουν .Κάθε φύλλο περιέχει όλα εκείνα τα σύνθετα αντικείμενα των οποίων οι προσεγγίσεις περιέχονται στο χώρο του bucket που αντιστοιχεί στον κόμβο-φύλλο. Οι εσωτερικοί κόμβοι περιέχουν το παραλληλόγραμμο που προσεγγίζει τις περιοχές και τα περιεχόμενα των απογόνων του. Λόγω του τρόπου κατασκευής τους , τα GBD- Trees έχουν πολύ καλή απόδοση σε περιπτώσεις εισαγωγών και διαγραφών .

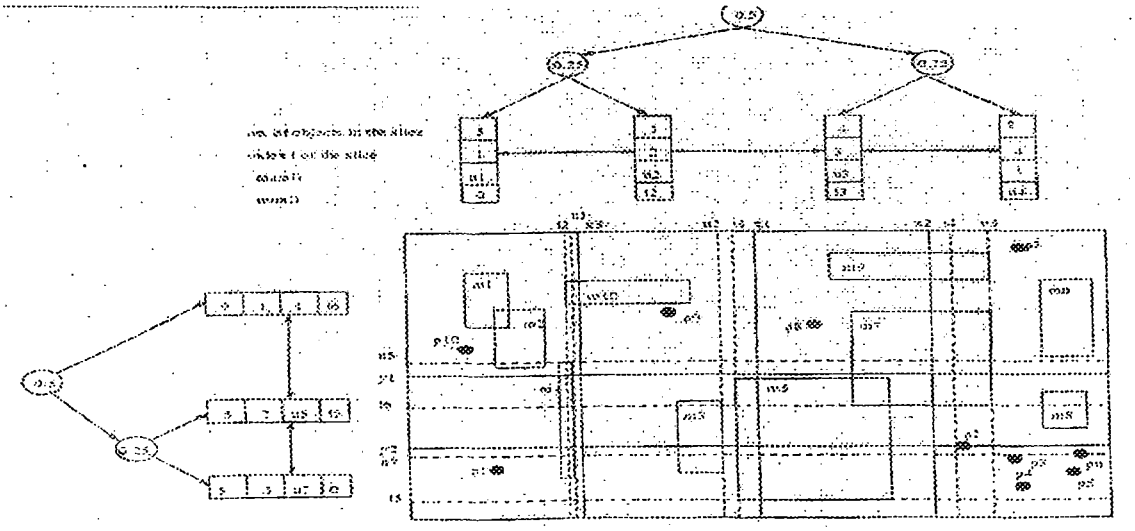


GBD-Tree

2.9.3.7. Το PLOP-Hashing

Πρόκειται για παραλλαγή της μεθόδου κατακερματισμού (hashing) που επιτρέπει την αποθήκευση και διαχείριση σύνθετων αντικειμένων χωρίς την ανάγκη μετατροπής τους σε σημεία. Ο χώρος χωρίζεται με τη μορφή πλέγματος κι ένα σύνθετο αντικείμενο μπορεί να εκτείνεται σε περισσότερα από ένα κελιά. Για την οργάνωση του χώρου χρησιμοποιούνται d δυαδικά δένδρα, όπου d είναι το πλήθος των διαστάσεων του δοσμένου χώρου.

Η τεχνική μπορεί με μικρές αλλαγές να υποστηρίξει αντιγραφή εκτός από επικαλυπτόμενες περιοχές. Πάντως, για ομοιόμορφη κατανομή των δεδομένων εισόδου έχει πολύ καλύτερη απόδοση σε σύγκριση με το R-Tree και το R*-Tree.



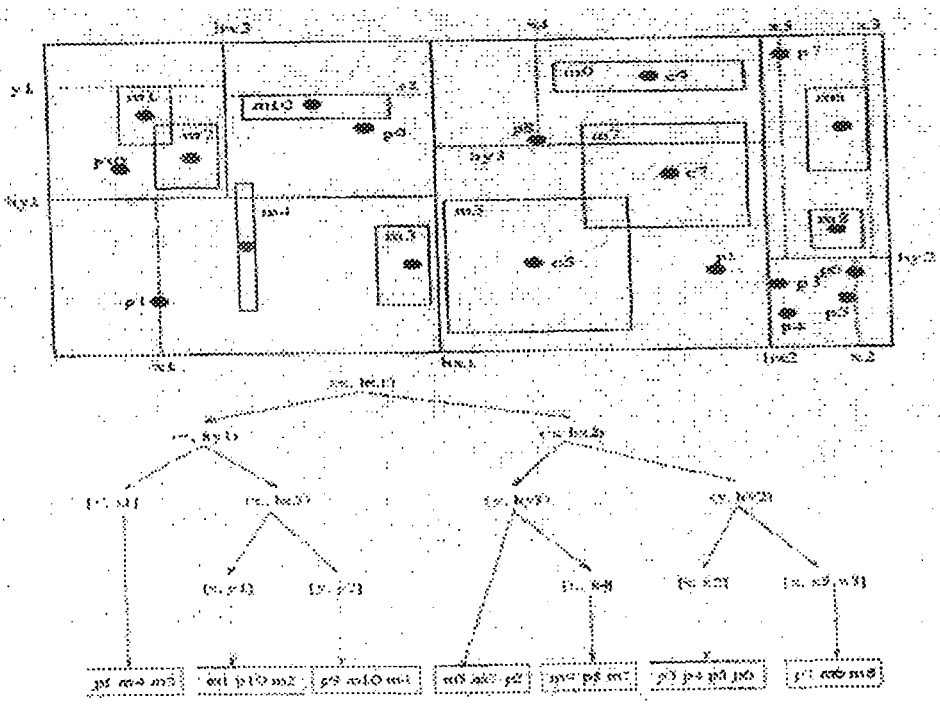
PLOP-Hashing

2.9.4 Δημιουργία Αντιγράφων

Δεν επιτρέπονται επικαλύψεις μεταξύ περιοχών των buckets, υιοθετείται δηλαδή η αρχή του αμοιβαίου αποκλεισμού. Επιπλέον, οι ερωτήσεις για σημεία ακολουθούν ένα κι όχι πολλαπλά μονοπάτια. Η μέθοδος παρουσιάζει πρόβλημα με διαδοχικές λειτουργίες εισαγωγής και διαγραφής γιατί πρέπει να δημιουργηθούν πολλές είσοδοι σε buckets για το ίδιο αντικείμενο με αποτέλεσμα τα δεδομένα να καταλαμβάνουν πολλές σελίδες στο δίσκο. Επιπλέον, όταν ένας χώρος δεδομένων που δεν είναι πλήρης διασπάται μια πιθανή λειτουργία εισαγωγής μπορεί να προκαλέσει μεγέθυνση πολλών περιοχών buckets και τελικά μια διάσπαση μπορεί να προκαλέσει νέες.

2.9.4.1. Το Εκτεταμένο K-D-Tree

Η μέθοδος βασίζεται στην αντιγραφή (clipping). Κάθε εσωτερικός κόμβος περιέχει ένα υπερεπίπεδο d-1 διαστάσεων ενώ τα φύλλα περιέχουν παραλληλόγραμμα με τις διευθύνσεις των σελίδων δεδομένων που περιέχουν ένα συγκεκριμένο τμήμα του προς επεξεργασία χώρου.

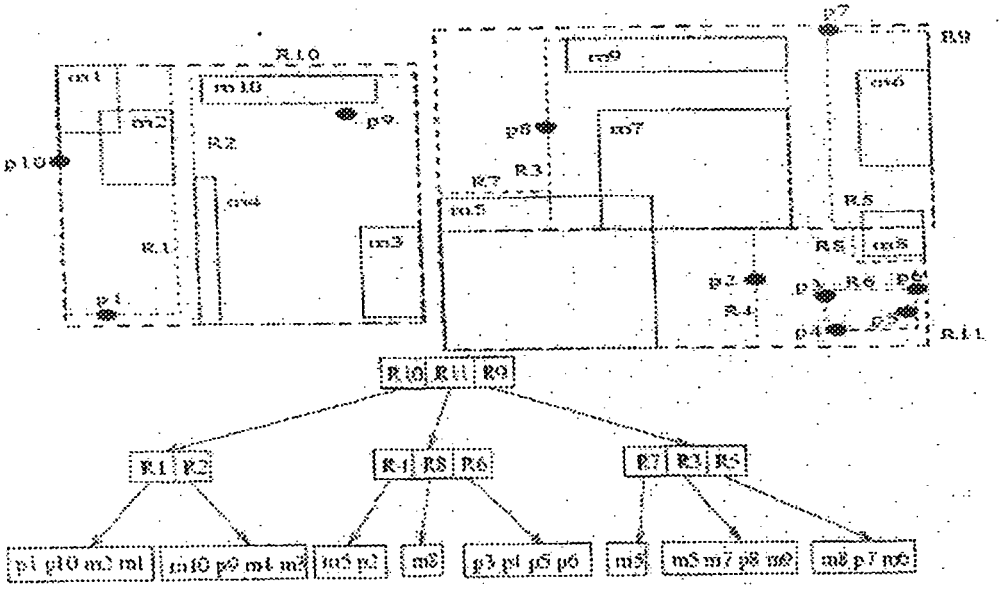


Εκτεταμένο K-D-Tree

2.9.4.2. Το R+-Tree

Η μέθοδος μοιάζει με το R- Tree αλλά βασίζεται στην αντιγραφή. Αντικείμενα που έχουν κοινά σημεία αποθηκεύονται σε περισσότερες από μία σελίδες. Λόγω κατασκευής, οι αναζητήσεις σημείων σε τέτοια δένδρα ισοδυναμούν με απλές διαδρομές πάνω στο δένδρο μέσω μονοπατιών από τη ρίζα στα φύλλα και γίνονται

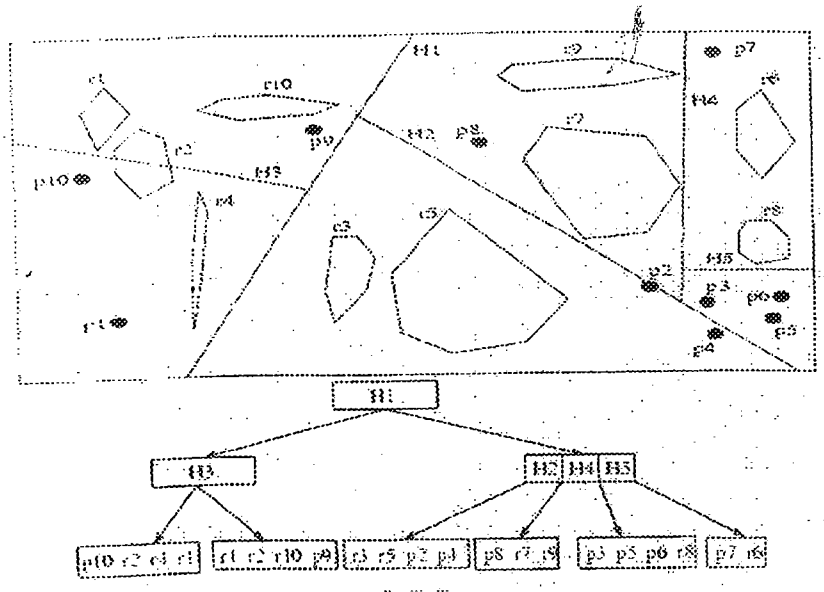
πολύ γρήγορα. Υπερχείλιση σε κάποιο φύλλο συνεπάγεται τη διάσπασή του σε δύο νέα. Η διαφορά του με τις προηγούμενες δομές είναι ότι οι διασπάσεις επιτρέπονται όχι μόνο για τα ανώτερα επίπεδα του δένδρου αλλά και για τα κατώτερα, εισάγοντας ενδεχομένως μεγαλύτερη πολυπλοκότητα στον τρόπο αποθήκευσης των σύνθετων αντικειμένων .Σε περίπτωση που γίνεται σαφής υποχρησιμοποίηση του διαθέσιμου αποθηκευτικού χώρου πρέπει να γίνει αναδιοργάνωση του δένδρου, πράγμα που δεν είναι πάντα δυνατό.



R+-Tree

2.9.4.3. To Cell Tree

Στόχος είναι η διευκόλυνση στην αναζήτηση σύνθετων αντικειμένων με αυθαίρετο σχήμα. Χρησιμοποιείται αντιγραφή για τη διαχείριση μεγάλων βάσεων σύνθετων αντικειμένων, πολυγώνων ή πολυέδρων μεγάλου αριθμού διαστάσεων. Ο χώρος διασπάται σε κυρτούς υποχώρους. Οι εσωτερικοί κόμβοι του δένδρου αντιστοιχούν σε ιεραρχίες εμφωλευμένων πολύτοπων, και αποθηκεύονται σε μία σελίδα του δίσκου. Τα φύλλα αντιστοιχούν σε διαφορετικούς υποχώρους και περιέχουν όλη την πληροφορία που είναι απαραίτητη για να δοθεί απάντηση σε κάποια ερώτηση προς τη βάση. Με τον τρόπο αυτό αποφεύγεται περιττή μετακίνηση πληροφορίας από το δευτερεύοντα αποθηκευτικό χώρο στην κύρια μνήμη, γεγονός που αποτελεί και ένα από τα σημαντικά πλεονεκτήματα της δομής.



Cell Tree

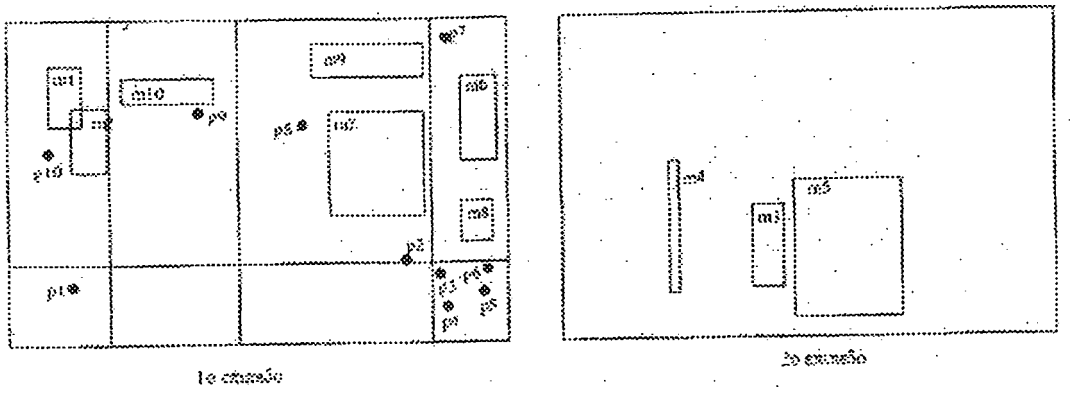
2.9.5. Πολλαπλά Στρώματα

Η μέθοδος μπορεί να θεωρηθεί παραλλαγή της μεθόδου των επικαλυπτόμενων περιοχών αφού οι περιοχές δεδομένων που αντιστοιχούν σε διαφορετικά στρώματα ενδέχεται να επικαλύπτονται. Εντούτοις υπάρχουν σημαντικές διαφορές ανάμεσα στις δύο μεθόδους .Τα στρώματα είναι οργανωμένα με ιεραρχικό τρόπο ενώ κάθε στρώμα είναι ουσιαστικά μια διαφορετική διάσπαση του προς επεξεργασία χώρου. Οι περιοχές στα πλαίσια του ίδιου στρώματος είναι ξένες μεταξύ τους, δηλαδή δεν έχουν κοινά σημεία κι επικαλύψεις ενώ δεν εμφανίζουν δυνατότητες προσαρμογής στις επιπλέον ιδιότητες των αποθηκευμένων σύνθετων αντικειμένων. Τα βασικά πλεονεκτήματα της μεθόδου συνοψίζονται στο γεγονός ότι δίνεται δυνατότητα ευρύτερης επιλογής κατά την αναζήτηση ενός αντικειμένου, λόγω του ότι δεν υπάρχουν επικαλύψεις μεταξύ των διαφορετικών στρωμάτων. Εντούτοις, η προσέγγιση μέσω πολλαπλών στρωμάτων ενδέχεται να αφήνει αχρησιμοποίητα μεγάλα τμήματα του διαθέσιμου αποθηκευτικού χώρου, ειδικά για συγκεκριμένες «κακές» κατανομές των δεδομένων εισόδου. Επιπλέον, συγκεκριμένες ερωτήσεις μπορεί να απαιτούν προσπέλαση όλων των υπαρχόντων στρωμάτων της ιεραρχίας. Τέλος, δεν υπάρχει σαφής μέθοδος για τον τρόπο ομαδοποίησης αντικειμένων που είναι κοντά μεταξύ τους αλλά σε διαφορετικά στρώματα, με αποτέλεσμα την ύπαρξη αμφιλογίας στην απόφαση για το στρώμα στο οποίο πρέπει να αποθηκευτεί ένα σύνθετο αντικείμενο.

2.9.5.1. Το Grid File Πολλαπλών Στρωμάτων

Πρόκειται για παραλλαγή του grid file για την αποθήκευση και διαχείριση σύνθετων αντικειμένων με χρήση ενός grid file πολλών διαταγμένων στρωμάτων, καθένα από α οποία αντιστοιχεί σε ένα διαφορετικό grid file που καλύπτει ολόκληρο τον προς επεξεργασία χώρο. Νέα αντικείμενα εισάγονται στο πρώτο grid file της ιεραρχίας χωρίς να υπάρχει ανάγκη αντιγραφής του ή μετακίνησής μεταξύ διαφορετικών στρωμάτων. Το μέγεθος των buckets αυξάνεται όσο μεταβαίνουμε σε ανώτερο

στρώμα, ενώ τα μεγάλα σε μέγεθος αντικείμενα αποστέλλονται σε ανώτερα στρώματα και σε περίπτωση που αυτό δεν είναι δυνατό δημιουργούνται νέα στρώματα. Πιθανά μειονεκτήματα της τεχνικής είναι η υποχρησιμοποίηση του διαθέσιμου αποθηκευτικού χώρου και το υψηλό κόστος της διατήρησης του καταλόγου.



Grid File Πολλαπλών Στρωμάτων

2.9.5.2. Το R-File

Στοχεύει στη διόρθωση των κακών χαρακτηριστικών του grid file πολλαπλών στρωμάτων .Χρησιμοποιείται ένας απλός κατάλογος. Ο χώρος διασπάται αναδρομικά σε ισομεγέθη τμήματα τα οποία κωδικοποιούνται σε buckets (σελίδες δίσκου) μέσω z-διάταξης. Επιτρέπονται οι επικαλύψεις αλλά όχι και η αντιγραφή αντικειμένων. Κάθε σύνθετο αντικείμενο αποθηκεύεται στο διαθέσιμο bucket με τη μικρότερη έκταση που μπορεί να το περιέχει, γεγονός που μερικές φορές κάνει αναπόφευκτη την υπερχείλιση σελίδων του δίσκου. Ένα ενδιαφέρον χαρακτηριστικό της δομής είναι ο αλγόριθμος διάσπασης του χώρου και των buckets που χρησιμοποιεί με αναδρομικό τρόπο ενώ διατηρούνται ξεχωριστά οι περιοχές του χώρου που περιέχουν αντικείμενα. Ένα μειονέκτημα της τεχνικής είναι το ότι διασπάται όλος ο χώρος ενώ δεικτοδοτείται μόνο εκείνο το υποσύνολο του που πραγματικά περιέχει αντικείμενα με αποτέλεσμα να παρουσιάζει κακή απόδοση για μη ομοιόμορφες κατανομές των δεδομένων εισόδου.

3ο Κεφάλαιο

"Διαχρονικότητα στις δομές δεδομένων"

3° Κεφάλαιο

3.1 Εισαγωγή

Οι κλασσικές δομές δεδομένων είναι εφήμερες με την έννοια ότι οποιαδήποτε αλλαγή σ' αυτές καταστρέφει την παλιά έκδοση και δημιουργεί μια νέα. Σε πολλές, όμως περιοχές της επιστήμης όπως στην υπολογιστική γεωμετρία, στη σύνταξη κειμένου στην υλοποίηση υψηλών γλωσσών προγραμματισμού είναι απαραίτητο να διατηρούνται πολλαπλές εκδόσεις των δομών δεδομένων .Θα καλούμε μια δομή δεδομένων διαχρονική (persistent) αν υποστηρίζει πρόσβαση σε πολλαπλές εκδόσεις. Υπάρχουν δύο μορφές της διαχρονικότητας. Μια δομή ονομάζεται μερικώς διαχρονική (partially persistent) αν υπάρχει δυνατότητα πρόσβασης σε όλες τις εκδόσεις και δυνατότητα ενημέρωσης (τροποποίηση) μόνο στην τελευταία έκδοση. Πλήρως διαχρονική (fully persistent) ονομάζεται η δομή στην οποία μπορεί να πραγματοποιηθεί πρόσβαση και ενημέρωση σε κάθε έκδοση.

Θα συζητηθούν γενικές τεχνικές για τη μετατροπή εφήμερων διασυνδεδεμένων δομών δεδομένων σε διαχρονικές με μικρό κόστος σε χώρο και χρόνο. Αρχικά πρέπει να προσδιορίσουμε τι ακριβώς είναι μια διασυνδεδεμένη δομή δεδομένων και τι είδους λειτουργίες επιτρέπονται στη δομή αυτή.

3.1.1 Ορισμός Διασυνδεδεμένης Δομής Δεδομένων

Ορίζουμε ως διασυνδεδεμένη δομή δεδομένων μια πεπερασμένη συλλογή από κόμβους, καθένας από τους οποίους περιέχει ένα σταθερό αριθμό από πεδία. Κάθε πεδίο μπορεί να είναι είτε πεδίο πληροφορίας, ικανό να διατηρεί ένα κομμάτι πληροφορίας ενός συγκεκριμένου τύπου, όπως ένα bit, έναν ακέραιο, έναν πραγματικό αριθμό, είτε ένα πεδίο δείκτη (pointer field), το οποίο διατηρεί ένα δείκτη σε ένα κόμβο ή την ειδική τιμή null (κανένα κόμβο). Για ευκολία θεωρούμε ότι όλοι οι κόμβοι στη δομή είναι ακριβώς του ίδιου τύπου, δηλαδή έχουν ακριβώς τα ίδια πεδία. Η πρόσβαση στη διασυνδεδεμένη δομή επιτυγχάνεται μέσω ενός σταθερού αριθμού δεικτών οι οποίοι ονομάζονται δείκτες πρόσβασης. Οι δείκτες πρόσβασης δείχνουν σε κόμβους της δομής οι οποίοι ονομάζονται κόμβοι εισόδου (entry nodes). Μπορούμε να αντιστοιχήσουμε μια διασυνδεδεμένη δομή με ένα κατευθυνόμενο γράφο του οποίου οι κορυφές έχουν σταθερό βαθμό εξόδου (out -degree). Αν ένας κόμβος x περιέχει ένα δείκτη σε ένα κόμβο y, θα καλούμε το y ως διάδοχο (successor) του x και το x ως αμέσως προηγούμενος κόμβο (predecessor) του y.

Παράδειγμα Διασυνδεδεμένης Δομής .

Ως παράδειγμα εφαρμογής χρησιμοποιείται ένα δυαδικό δέντρο αναζήτησης. Ένα δυαδικό δέντρο αναζήτησης είναι ένα δυαδικό δέντρο το οποίο περιέχει στους κόμβους του διακεκριμένα αντικείμενα τα οποία επιλέγονται από ένα διατεταγμένο σύνολο. Σε κάθε κόμβο αποθηκεύεται ένα αντικείμενο έτσι ώστε να ισχύει το εξής. Αν x είναι ένας οποιοδήποτε κόμβος, τότε το αντικείμενο του x είναι μεγαλύτερο από όλα τα αντικείμενα τα οποία βρίσκονται στο αριστερό υποδέντρο του και μικρότερο

από όλα τα αντικείμενα τα οποία βρίσκονται στο δεξιό υποδέντρο του x. Ένα δυαδικό δέντρο μπορεί να αναπαρασταθεί ως μια διασυνδεδεμένη δομή στην οποία κάθε κόμβος περιέχει τρία πεδία. Ένα πεδίο με το αντικείμενο πληροφορία και δύο πεδία αριστερό και δεξί στα οποία περιέχονται ένας δεξιός και ένας αριστερός δείκτης οι οποίοι δεικτοδοτούν στο δεξί και στο αριστερό παιδί του κόμβου αντίστοιχα. Η ρίζα του δέντρου είναι ο μοναδικός κόμβος εισόδου στον οποίο μπορεί να πραγματοποιηθεί πρόσβαση στη δομή.

3.2 Επιτρεπτές Λειτουργίες Σε Μια Διασυνδεδεμένη Δομή

Σε μια γενική διασυνδεδεμένη δομή δεδομένων επιτρέπονται δύο είδη λειτουργιών: Λειτουργίες πρόσβασης (access operations) και λειτουργίες ενημέρωσης (update operations).

3.2.1 Λειτουργία Πρόσβασης

Μια λειτουργία πρόσβασης υπολογίζει ένα σύνολο πρόσβασης (accessed set) το οποίο αποτελείται από κόμβούς στους οποίους έχει γίνει πρόσβαση (accessed nodes). Στην αρχή της λειτουργίας το σύνολο πρόσβασης είναι άδειο. Η λειτουργία αποτελείται από μια ακολουθία βημάτων πρόσβασης (access steps) , καθένα από τα οποία προσθέτει ένα κόμβο στο σύνολο πρόσβασης. Αυτός ο κόμβος πρέπει είτε να είναι ένας κόμβος εισόδου ή να δεικτοδοτείται από ένα δείκτη ενός κόμβου στον οποίο έχει γίνει ήδη πρόσβαση. Ο χρόνος μιας λειτουργίας πρόσβασης προσδιορίζεται από τον αριθμό βημάτων πρόσβασης που εκτελέστηκαν. Τελικά η λειτουργία πρόσβασης παράγει ως έξοδο ένα σύνολο πληροφοριών το οποίο περιέχεται τους κόμβους στους οποίους έχει γίνει πρόσβαση.

Παράδειγμα λειτουργίας πρόσβασης

Ένα παράδειγμα μιας λειτουργίας πρόσβασης είναι η αναζήτηση σε ένα δυαδικό δέντρο. Το σύνολο πρόσβασης σχηματίζει ένα μονοπάτι στο δέντρο το οποίο ξεκινά από τη ρίζα και πηγαίνει σε ένα κόμβο κάθε φορά μέχρι να βρεθεί το επιθυμητό αντικείμενο ή να βρεθεί ένας null δείκτης. Η εύρεση ενός null δείκτη σημαίνει ότι το αντικείμενο δε βρίσκεται στο δέντρο.

3.2.2 Λειτουργία Ενημέρωσης

Μια λειτουργία ενημέρωσης σε μια γενική διασυνδεδεμένη δομή αποτελείται από μια ακολουθία βημάτων πρόσβασης και ενημέρωσης (access and update steps). Τα βήματα πρόσβασης υπολογίζουν το σύνολο κόμβων πρόσβασης με όμοιο τρόπο με τη λειτουργία πρόσβασης. Τα βήματα ενημέρωσης αλλάζουν τη δομή δεδομένων. Ένα βήμα ενημέρωσης είναι είτε η δημιουργία ενός νέου κόμβου, ο οποίος προστίθεται στο σύνολο πρόσβασης, είτε η αλλαγή ενός πεδίου πληροφορίας ή ενός δείκτη σε ένα κόμβο. Επισημαίνεται ότι κατά τη δημιουργία ενός νέου κόμβου θα πρέπει να αρχικοποιηθούν τα πεδία πληροφορίας του και να τεθούν σε null οι δείκτες. Ο συνολικός χρόνος για μια λειτουργία ενημέρωσης ορίζεται ως ο συνολικός αριθμός βημάτων πρόσβασης και ενημέρωσης. Ο χρόνος μόνο για τη φάση της ενημέρωσης (update time) προσδιορίζεται από των αριθμό των βημάτων ενημέρωσης.

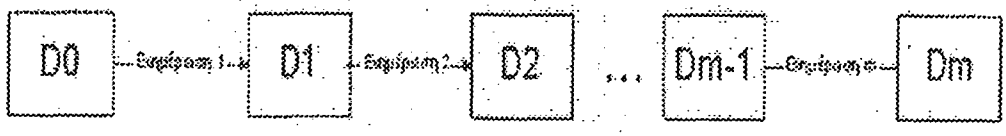
Παράδειγμα λειτουργίας ενημέρωσης

Ένα παράδειγμα μιας λειτουργίας ενημέρωσης είναι η εισαγωγή ενός νέου αντικειμένου σε ένα δυαδικό δέντρο αναζήτησης. Η εισαγωγή αποτελείται από την αναζήτηση του προς εισαγωγή αντικειμένου και την προσθήκη στη θέση του κόμβου, στον οποίο θα έπρεπε να υπάρχει το αντικείμενο, ενός νέου κόμβου που θα περιέχει το νέο αντικείμενο. Μία πιο πολύπλοκη λειτουργία ενημέρωσης είναι η διαγραφή ενός αντικειμένου. Η διαδικασία διαγραφής αποτελείται από τρεις φάσεις. Αρχικά γίνεται η αναζήτηση του αντικειμένου. Στη συνέχεια αν ο κόμβος έστω x, που περιέχει το αντικείμενο, έχει δύο παιδιά τότε ο x αντιμετατίθεται με τον κόμβο ο οποίος προηγείται του x και μπορεί να βρεθεί αρχίζοντας από το αριστερό παιδί και ακολουθώντας δεξιούς δείκτες μέχρι να εμφανιστεί κόμβος χωρίς δεξί παιδί. Μετά την αντιμετάθεση ο x έχει εγγυημένα το πολύ ένα παιδί (το αριστερό). Η τρίτη φάση αποτελείται από τη διαγραφή του x από το δέντρο και την αντικατάσταση του από το μοναδικό του παιδί, εφόσον υπάρχει. Το υποδέντρο το οποίο έχει ρίζα αυτό το παιδί παραμένει ανεπηρέαστο από τη διαγραφή. Ο συνολικός χρόνος για την εισαγωγή και τη διαγραφή είναι το βάθος κάποιου κόμβου του δέντρου συν μια σταθερά. Ο χρόνος ενημέρωσης είναι μόνο μια σταθερά.

3.3 Ορισμοί Διαχρονικότητας

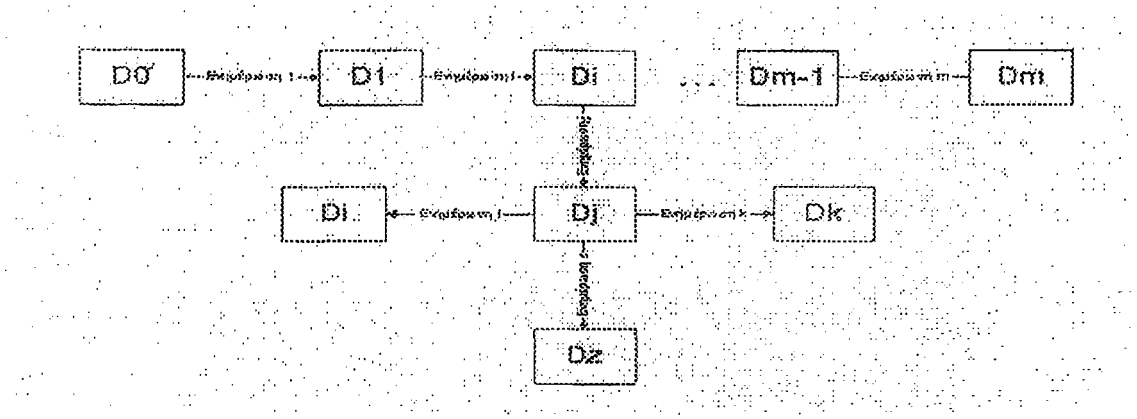
Επιστρέφοντας στην περίπτωση μιας γενικής διασυνδεδεμένης δομής θεωρούμε μια ακολουθία από λειτουργίες πρόσβασης και ενημέρωσης σε μια αρχικά άδεια δομή (όλοι οι δείκτες πρόσβασης είναι null). Δηλώνεται με m ο συνολικός αριθμός των λειτουργιών ενημέρωσης. Επίσης αριθμούμε τις λειτουργίες ενημέρωσης και τις εκδόσεις της δομής με ακεραίους. Η ενημέρωση i είναι η i-οστή ενημέρωση στην ακολουθία. Η έκδοση 0 είναι η αρχική (άδεια) έκδοση, ενώ η έκδοση i είναι η έκδοση που παράγεται από την ενημέρωση i. Η λειτουργίες που θα πραγματοποιηθούν θεωρούμε ότι είναι on-line με την έννοια ότι κάθε λειτουργία γίνεται γνωστή αφού πραγματοποιηθεί η προηγούμενή της.

Οι δομές δεδομένων μπορούν να χαρακτηριστούν ως εφήμερες ή διαχρονικές με βάση τις επιτρεπόμενες ακολουθίες λειτουργιών. Μια εφήμερη δομή υποστηρίζει ακολουθίες στις οποίες κάθε λειτουργία εφαρμόζεται στην πιο πρόσφατη έκδοση της δομής. Μια μερικώς διαχρονική δομή υποστηρίζει ακολουθίες στις οποίες κάθε ενημέρωση μπορεί να εφαρμοστεί μόνο στην πιο πρόσφατη έκδοση (η ενημέρωση i εφαρμόζεται στην έκδοση i-1), ενώ οι προσβάσεις μπορούν να εφαρμοστούν σε οποιαδήποτε προηγούμενη υπάρχουσα έκδοση (η οποία πρέπει να καθορίζεται). Αυτό φαίνεται σχηματικά παρακάτω.



Αναπαράσταση μιας μερικώς διαχρονικής δομής

Μία πλήρως διαχρονική δομή υποστηρίζει οποιαδήποτε ακολουθία στην οποία κάθε λειτουργία εφαρμόζεται σε μια οποιαδήποτε υπάρχουσα έκδοση. Το αποτέλεσμα μιας ενημέρωσης είναι μια εντελώς νέα έκδοση, ξεχωριστή από τις άλλες. Αυτό φαίνεται σχηματικά παρακάτω.



Αναπαράσταση μιας πλήρως διαχρονικής δομής

Για οποιαδήποτε από τα παραπάνω είδη δομών, καλούμε τη λειτουργία η οποία εκτελείται, ως τρέχουσα λειτουργία (current operation) και την έκδοση της δομής στην οποία εφαρμόζεται η τρέχουσα λειτουργία ως τρέχουσα έκδοση (current version). Η τρέχουσα έκδοση δεν είναι απαραίτητα ίδια με την τελευταία έκδοση (εκτός της περίπτωσης της εφήμερης δομής) και γι ' αυτό γενικά η έκδοση θα πρέπει να προσδιορίζεται ως παράμετρος στην τρέχουσα λειτουργία. Σε μια πλήρως διαχρονική δομή, αν μία λειτουργία ενημέρωσης ί εφαρμοστεί σε μια έκδοση j<i, το αποτέλεσμα της ενημέρωσης είναι μια έκδοση ί. Σημειώνεται ότι η έκδοση j δεν αλλάζει από τη λειτουργία της ενημέρωσης αλλά διατηρείται αμετάβλητη. Δηλώνουμε με n τον αριθμό των κόμβων στην τρέχουσα έκδοση.

Έστω ότι δίνεται μία εφήμερη δομή και οι υλοποιήσεις των διαφόρων ειδών λειτουργιών οι οποίες επιτρέπονται στη δομή. Θέλουμε να κάνουμε τη δομή διαχρονική, με την έννοια όπως αναφέρθηκε παραπάνω να επιτρέπονται οι λειτουργίες σε όλες τις εκδόσεις της δομής που προκύπτουν. Το να γίνει μια δομή διαχρονική απαιτεί την κατασκευή μιας δομής δεδομένων που να αναπαριστά όλες τις εκδόσεις ταυτόχρονα, επιτρέποντας πρόσβαση και πιθανές λειτουργίες ενημέρωσης να συμβαίνουν σε οποιαδήποτε έκδοση, οποιαδήποτε χρονική στιγμή. Η δομή δεδομένων αυτή θα αποτελείται από μια διασυνδεδεμένη δομή, στην οποία κάθε έκδοση της εφήμερης δομής θα είναι ενσωματωμένη σε αυτή, ώστε κάθε βήμα πρόσβασης ή ενημέρωση σε μια έκδοση της εφήμερης δομής να μπορεί να προσομοιώνεται (ιδανικά σε 0(1))στο αντίστοιχο τμήμα της διαχρονικής δομής.

3.4 Μερική Διαχρονικότητα (Partial Persistence)

Σκοπός αυτής της ενότητας είναι ο σχεδιασμός μιας γενικής τεχνικής που να μετατρέπει μια εφήμερη διασυνδεδεμένη δομή δεδομένων σε μερικώς διαχρονική. Προτείνονται δύο μέθοδοι. Η πρώτη και απλούστερη μέθοδος είναι η "fat node method", η οποία εφαρμόζεται σε οποιαδήποτε εφήμερη διασυνδεδεμένη δομή

δεδομένων και την κάνει διαχρονική με κόστος χώρου χειρότερης περίπτωσης 0(1) για κάθε βήμα ενημέρωσης και κόστος χρόνου χειρότερης περίπτωσης O(logm) για κάθε βήμα πρόσβασης ή ενημέρωσης. Περισσότερη πολύπλοκη είναι η μέθοδος «node-copying» η οποία εφαρμόζεται σε μία εφήμερη διασυνδεδεμένη δομή της οποίας οι κόμβοι έχουν σταθερό βαθμό εισόδου. Στη μέθοδο αυτή στη μερικώς διαχρονική δομή που δημιουργείται κάθε βήμα ενημέρωσης προσομοιώνεται σε επιμερισμένο κόστος χρόνου και χώρου 0(1) και κάθε βήμα πρόσβασης σε επιμερισμένο κόστος χώρου 0(1).

3.4.1 Γνωστές Μέθοδοι

Αρχικά θα κάνουμε μια ανασκόπηση των αποτελεσμάτων του Overmars ο οποίος μελέτησε τρεις απλούς αλλά γενικούς τρόπους με τους οποίους επιτυγχάνεται μερική διαχρονικότητα. Η πρώτη μέθοδος αποθηκεύει κάθε έκδοση ξεχωριστά, αντιγράφοντας ολόκληρη την εφήμερη δομή για κάθε λειτουργία ενημέρωσης. Αυτή η διαδικασία κοστίζει Ω(n) χρόνο και χώρο για κάθε ενημέρωση.

Μια εναλλακτική μέθοδος είναι να αποθηκεύεται μόνο η αρχική έκδοση καθώς και ολόκληρης της ακολουθίας των λειτουργιών ενημέρωσης. Κάθε φορά που πραγματοποιείται μια λειτουργία πρόσβασης κατασκευάζεται η αντίστοιχη έκδοση της δομής στην οποία ζητείται πρόσβαση από την αρχική έκδοση χρησιμοποιώντας την κατάλληλη ακολουθία ενημερώσεων .Υποθέτοντας ότι μια λειτουργία ενημέρωσης καταναλώνει 0(1) χώρο, η μέθοδος χρησιμοποιεί συνολικά μόνο O(m) χώρο. Η πρόσβαση σε μια έκδοση i, κοστίζει Ω(i) χρόνο ακόμα και αν κάθε λειτουργία ενημέρωσης κοστίζει 0(1). Παρατηρούμε ότι η πρώτη μέθοδος έχει μεγάλο κόστος για κάθε λειτουργία ενημέρωσης και μικρό κόστος για τη λειτουργία πρόσβασης, ενώ η δεύτερη ακριβώς τα αντίθετα.

Μια υβριδική μέθοδος η οποία χρησιμοποιεί τις δύο παραπάνω μεθόδους είναι η εξής: Αποθηκεύεται ολόκληρη η ακολουθία των λειτουργιών ενημέρωσης και επιπρόσθετα κάθε k-οστή έκδοση για κάποιο επιλεγμένο k. Η πρόσβαση στην έκδοση i απαιτεί την κατασκευή τη i-οστής έκδοσης της δομής από την έκδοση k [i/k] εκτελώντας την κατάλληλη ακολουθία λειτουργιών ενημέρωσης. Σε αυτήν τη μέθοδο αν το k είναι μεγάλο τότε έχουμε μεγάλο κόστος στο χρόνο πρόσβασης και μικρό κόστος χώρου για κάθε ενημέρωση, ενώ αν είναι μικρό ο χρόνος πρόσβασης είναι μικρότερος ενώ το κόστος χώρου είναι μεγάλο.

Στη συνέχεια θα παρουσιάσουμε περισσότερο αποδοτικές τεχνικές. Στην ιδανική περίπτωση θα θέλαμε ο χώρος αποθήκευσης που χρησιμοποιείται για τη διαχρονική δομή να είναι 0(1) για κάθε βήμα ενημέρωσης και ο χρόνος ανά λειτουργία να αυξάνει μόνο κατά ένα σταθερό παράγοντα σε σχέση με το χρόνο που απαιτείται στην εφήμερη δομή. Ένας λόγος για τον οποίο τα αποτελέσματα του Overmars είναι μη αποδοτικά είναι επειδή γίνονται πολύ λίγες υποθέσεις για την εφήμερη δομή. Περιορίζοντας το αντικείμενο της διαχρονικότητας μόνο για διασυνδεδεμένες δομές και εστιάζοντας σε λεπτομέρειες των βημάτων ενημέρωσης επιτυγχάνονται πολύ καλύτερα αποτελέσματα.

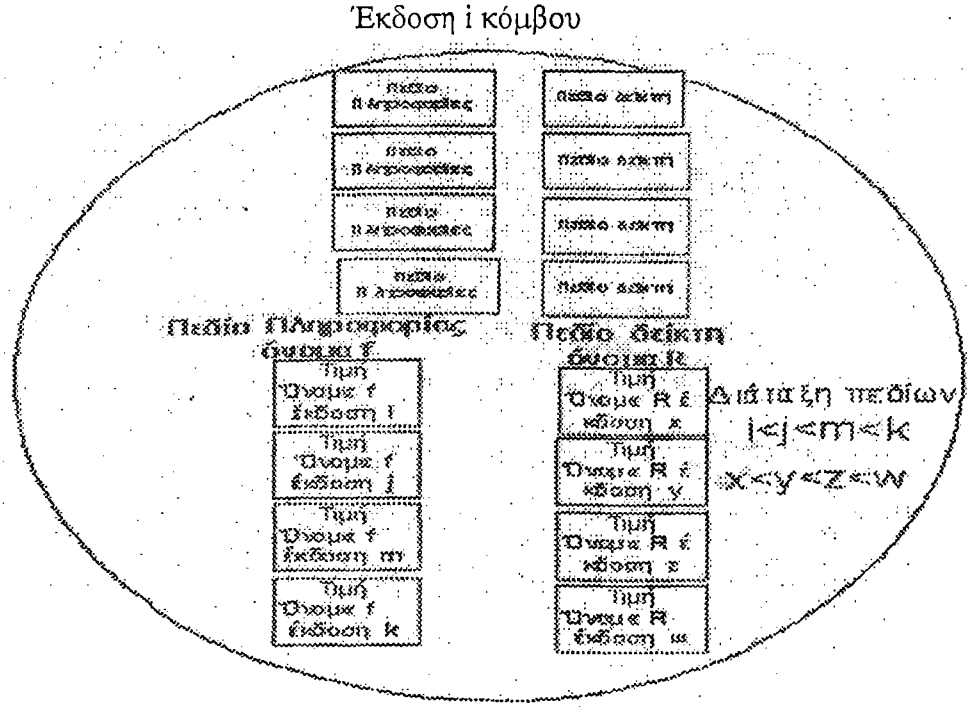
3.5 Η Μέθοδος "Fat Node"

3.5.1 Η Δομή Δεδομένων

Η πρώτη ιδέα είναι να καταγράφονται όλες οι αλλαγές οι οποίες γίνονται σε πεδία του κόμβου μέσα στους ίδιους τους κόμβους χωρίς να διαγράφονται παλιές τιμές των πεδίων. Αυτό απαιτεί να επιτρέπεται οι κόμβοι να γίνονται αυθαίρετα παχιοί, ώστε να κρατάνε έναν αυθαίρετο αριθμό από τιμές για κάθε πεδίο. Πιο αναλυτικά, κάθε κόμβος θα περιέχει την ίδια πληροφορία και τα πεδία δεικτών όπως ο εφήμερος κόμβος (κρατούνται οι αρχικές τιμές των πεδίων), μαζί με τον απαραίτητο χώρο για ένα αυθαίρετο αριθμό από επιπλέον (extra) τιμές πεδίων. Κάθε επιπλέον τιμή πεδίου διατηρείται μαζί με το όνομα του πεδίου και μια στάμπα έκδοσης (αριθμός έκδοσης). Η στάμπα έκδοσης δείχνει την έκδοση στην οποία η τιμή του πεδίου άλλαξε. Επιπλέον, κάθε παχύς κόμβος διαθέτει τη δική του στάμπα έκδοσης, η οποία δείχνει την έκδοση στην οποία ο κόμβος δημιουργήθηκε.

Επιπρόσθετα απαιτείται μια βοηθητική δομή δεδομένων για την αποθήκευση των δεικτών πρόσβασης στις διάφορες εκδόσεις. Αυτή η δομή αποτελείται από έναν πίνακα από δείκτες για κάθε όνομα δείκτη πρόσβασης. Μετά από κάθε λειτουργία i, αποθηκεύουμε τις πρόσφατες τιμές των δεικτών πρόσβασης στις i-οστές θέσεις των πινάκων πρόσβασης. Με αυτή τη δομή, η αρχικοποίηση της πρόσβασης σε οποιαδήποτε έκδοση κοστίζει 0(1) χρόνο.

Η δομή του fat node η οποία δημιουργείται κατά τη διάρκεια της μεθόδου φαίνεται στο παρακάτω σχήμα.



Δομή "Fat Node"

3.5.2 Προσομοίωση λειτουργιών

Τα εφήμερα βήματα ενημέρωσης στη δομή του "fat node" προσομοιώνονται ως εξής. Θεωρούμε τη λειτουργία ενημέρωσης i. Όταν ένα εφήμερο βήμα ενημέρωσης δημιουργεί ένα νέο κόμβο, στην περίπτωση του "fat node" δημιουργείται ένας αντίστοιχος παχύς κόμβος, με στάμπα έκδοσης i, ο οποίος περιέχει τις κατάλληλες αρχικές τιμές πληροφοριών και τα πεδία των δεικτών. Όταν ένα εφήμερο βήμα αλλάζει την τιμή ενός πεδίου στο κόμβο, εμείς δημιουργούμε την αντίστοιχη νέα τιμή στον αντίστοιχο παχύ κόμβο, μαζί με το όνομα και το πεδίο αλλαγμένα ώστε να έχουν στάμπα έκδοσης i. Σημειώνουμε ότι για κάθε πεδίο στον κόμβο μπορεί να αποθηκεύεται μόνο μια τιμή για κάθε έκδοση. Κατά την αποθήκευση της τιμής ενός πεδίου, αν υπάρχει τιμή για το ίδιο πεδίο με την ίδια στάμπα έκδοσης, τότε αντικαθίσταται η παλιά με τη νέα τιμή. Οι αρχικές τιμές των πεδίων θεωρείται ότι έχουν την στάμπα έκδοσης του κόμβου που τα περιέχει.

Η διαχρονική δομή που προκύπτει από την παραπάνω διαδικασία έχει όλες τις εκδόσεις τις εφήμερης δομής ενσωματωμένες σε αυτή. Η πλοήγηση μέσα σε μια διαχρονική δομή γίνεται ως εξής. Όταν ένα εφήμερο βήμα πρόσβασης απαιτεί πρόσβαση στην έκδοση i ενός πεδίου f ενός κόμβου, στη διαχρονική δομή γίνεται πρόσβαση στην τιμή του αντίστοιχου παχιού κόμβου του οποίου το όνομα του πεδίου είναι f, επιλέγοντας ανάμεσα από τις τιμές με το ίδιο όνομα πεδίου f αυτή ε τη μεγαλύτερη στάμπα έκδοσης η οποία δεν είναι μεγαλύτερη της i. Ουσιαστικά δηλαδή γίνεται πρόσβαση στην πιο πρόσφατη τιμή του πεδίου f..

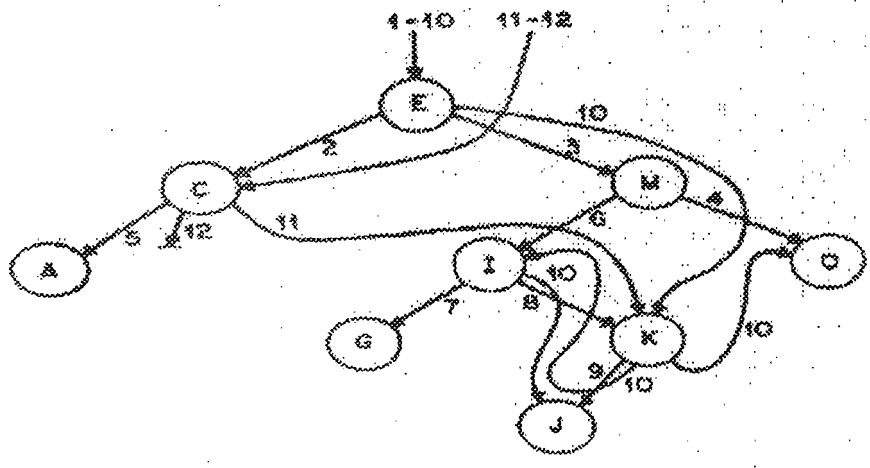
Παρατήρηση: Ο μόνος λόγος για τον οποίο οι κόμβοι έχουν στάμπες έκδοσης είναι για τη διασφάλιση ότι κάθε κόμβος περιέχει μια τιμή ανά όνομα πεδίου σε κάθε έκδοση. Οι στάμπες αυτές δεν είναι απαραίτητες αν υπάρχει κάποιος άλλος μηχανισμός ο οποίος θα παρακολουθεί τη λειτουργία ενημέρωσης i, για νέες τιμές πεδίων σε κόμβους που δημιουργήθηκαν κατά τη διάρκεια της ενημέρωσης i. Για ην πλοήγηση στη δομή με αυτόν το μηχανισμό αρκεί οι αρχικές τιμές των πεδίων ε κάθε κόμβο να έχουν στάμπα έκδοσης 0.

3.5.3 Ανάλυση Κόστους Χώρου Και Χρόνου

Η μέθοδος "fat node" εφαρμόζεται σε οποιαδήποτε διασυνδεδεμένη δομή και χρησιμοποιεί μόνο 0(1) χώρο για κάθε εφήμερο βήμα ενημέρωσης στη χειρότερη περίπτωση, αλλά έχει δύο μειονεκτήματα. Πρώτον οι παχιοί κόμβοι θα πρέπει να αναπαρασταθούν από διασυνδεδεμένες συλλογές από σταθερού μεγέθους κόμβους, γεγονός που περιπλέκει την υλοποίηση. Δεύτερον, η επιλογή του δείκτη σε ένα παχύ κόμβο που θα ακολουθηθεί κατά την προσομοίωση ενός βήματος πρόσβασης κοστίζει 0(m) χρόνο χειρότερης περίπτωσης. Αν οι τιμές σε ένα πεδίο μέσα σε ένα παχύ κόμβο διαταχθούν με βάση τη στάμπα έκδοσης και αποθηκευτούν σε ένα δυαδικό δέντρο αναζήτησης, η προσομοίωση ενός εφήμερου βήματος πρόσβασης και ενημέρωσης θα κοστίζει σε χρόνο O(logm). Αυτό σημαίνει ότι υπάρχει ένα λογαριθμικός παράγοντας ο οποίος επιβαρύνει τις λειτουργίες πρόσβασης και ενημέρωσης.

Παράδειγμα

Το σχήμα 4 δείχνει ένα διαχρονικό δυαδικό δέντρο αναζήτησης το οποίο κατασκευάζεται χρησιμοποιώντας τη μέθοδο του "fat node". Οι λειτουργίες ενημέρωσης είναι οι εισαγωγές και οι διαγραφές οι οποίες πραγματοποιούνται με τον τρόπο που περιγράφτηκε στην εισαγωγή. Οι στάμπες έκδοσης στους κόμβους μπορούν εύκολα να παραληφθούν σε αυτή την εφαρμογή. Αφού οι αρχικοί δείκτες σε κάθε κόμβο είναι γνωστό ότι είναι null, μπορούν και αυτοί να παραληφθούν .(Η έκδοση του πεδίου f δείκτη ενός κόμβου θεωρείται ότι είναι null αν κανένα f πεδίο δείκτη δεν αποθηκεύτηκε στον κόμβο, το οποίο έχει στάμπα έκδοσης μικρότερη ή ίση του i.)



Ένα μερικώς διαχρονικό δέντρο αναζήτησης το οποίο κατασκευάζεται χρησιμοποιώντας τη μέθοδο του "fat node", για την ακολουθία των λειτουργιών ενημέρωσης που αποτελείται από τις εισαγωγές των E, C, M, O, A, I, G, K, J, ακολουθούμενες από τις διαγραφές των M, E και του A. Οι επιπλέον δείκτες διαθέτουν τις δικές τους στάμπες έκδοσης. Οι στάμπες έκδοσης των αρχικών null δεικτών έχουν παραληφθεί.

3.6 Η Μέθοδος "Node Copying"

Τα μειονεκτήματα της μεθόδου "fat node" εξαλείφονται με μια δεύτερη μέθοδο που ονομάζεται "node copying". Σε αυτή τη μέθοδο επιτρέπεται στους κόμβους της διαχρονικής δομής να διατηρούν μόνο ένα σταθερό αριθμό από τιμές πεδίων .Όταν εξαντλείται ο χώρος σε ένα κόμβο, η μέθοδος δημιουργεί ένα καινούργιο αντίγραφο του κόμβου, το οποίο περιέχει μόνο τη νεώτερη τιμή κάθε πεδίου. Πρέπει επίσης να αποθηκευτούν στη νεότερη έκδοση, δείκτες από όλους τους προκάτοχους κόμβους (αμέσως προηγούμενοι κόμβοι) προς το νέο αντίγραφο του κόμβου. Αν δεν υπάρχει χώρος σε έναν προκάτοχο για την αποθήκευση αυτού του δείκτη, τότε πρέπει να δημιουργηθεί και γι ' αυτόν αντίγραφο. Ωστόσο, αν υποθέσουμε ότι η βασική εφήμερη δομή διαθέτει κόμβους οι οποίοι έχουν σταθερό βαθμό εισόδου και προσθέσουμε σε κάθε κόμβο της διαχρονικής δομής τον απαραίτητο εμπλέον χώρο, τότε μπορούμε να πετύχουμε για κάθε βήμα ενημέρωσης 0(1) amortized όριο για τον

αριθμό των κόμβων που γίνονται αντίγραφα και επομένως και για τον απαιτούμενο χρόνο. Η ουσία αυτής της μεθόδου είναι ότι προσθέτουμε επιπρόσθετους δείκτες και κόμβους στη δομή, αυξάνουμε δηλαδή το κόστος στο χώρο με σκοπό να μειώσουμε το κόστος στο χρόνο πρόσβασης και ενημέρωσης.

3.6.1 Η Δομή Δεδομένων

Πριν προχωρήσουμε σε λεπτομέρειες της μεθόδου αναφέρουμε την ορολογία που θα χρησιμοποιήσουμε. Θα καλούμε έναν κόμβο που ανήκει στη βασική εφήμερη δομή ως εφήμερο κόμβο (ephemeral node) και τον κόμβο που ανήκει στη διαχρονική δομή ως διαχρονικό κόμβο (persistent node). Αν x είναι ένας εφήμερος κόμβος που υπάρχει στην έκδοση i της εφήμερης δομής, τότε η έκδοση i του x είναι ο x μαζί με όλες τις τιμές των πεδίων της έκδοσης i της δομής. Θεωρούμε δηλαδή ότι τα πεδία του εφήμερου κόμβου $\bar{x}$ αλλάζουν καθώς αυτός περνά στις διάφορες εκδόσεις της δομής. Δηλώνουμε επίσης με $\bar{x}$ ένα διαχρονικό κόμβο ο οποίος είναι αντίστοιχος με του εφήμερου κόμβου

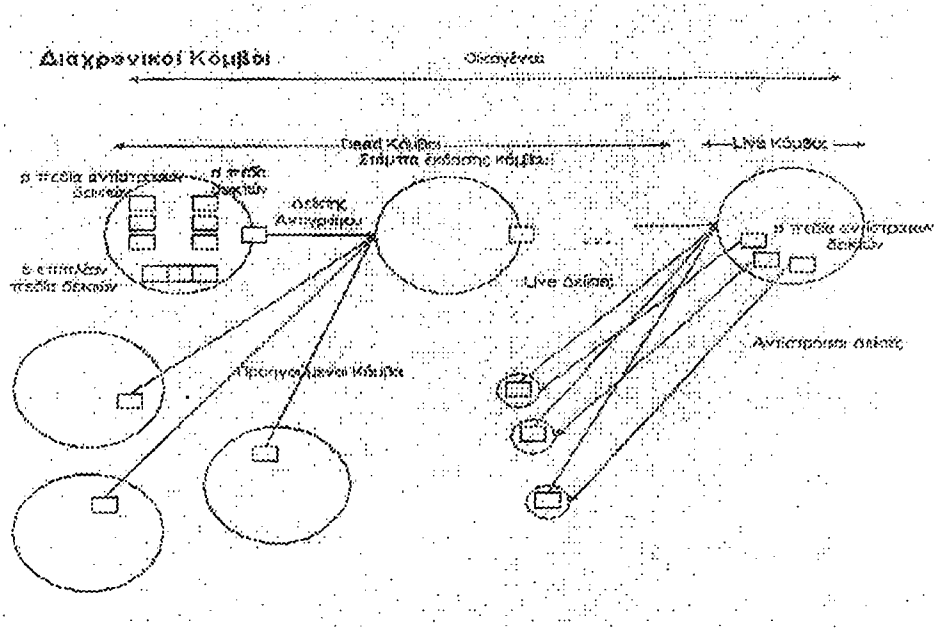
Στην "node copying" μέθοδο κάθε διαχρονικός κόμβος περιέχει μόνο μια έκδοση κάθε πεδίου πληροφορίας, αλλά μπορεί να περιέχει πολλαπλές εκδόσεις των πεδίων δεικτών (μπορούν εύκολα να κατασκευαστούν παραλλαγές της μεθόδου ώστε να επιτρέπουν πολλαπλές εκδόσεις των πεδίων πληροφορίας). Έστω d ο αριθμός των πεδίων δεικτών σε ένα εφήμερο κόμβο και p ο μέγιστος αριθμός των προκατόχων ενός εφήμερο κόμβου σε κάποια έκδοση. Υποθέτουμε ότι το p είναι σταθερό. Κάθε διαχρονικός κόμβος θα περιέχει d + p + e + 1 πεδία δεικτών, όπου το e είναι μια επαρκώς μεγάλη σταθερά η οποία θα επιλεχθεί αργότερα. Από αυτά τα πεδία, τα d είναι τα ίδια πεδία δεικτών με αυτά του εφήμερου κόμβου και περιέχουν τους αρχικούς δείκτες (original pointers), p είναι δείκτες στους προκατόχους του κόμβου (predecessor pointers), e είναι επιπλέον δείκτες και ένας είναι ο δείκτης αντιγράφου (copy pointer). Κάθε διαχρονικός κόμβος έχει τα ίδια πεδία πληροφορίας όπως ο εφήμερος κόμβος, και επιπλέον μια στάμπα έκδοσης για τον ίδιο τον κόμβο και ένα όνομα πεδίου και στάμπα έκδοσης για κάθε επιπλέον δείκτη. Οι αρχικοί δείκτες στον κόμβο θεωρείται ότι έχουν την ίδια στάμπα έκδοσης με αυτή του κόμβου.

Η αντιστοίχηση μεταξύ της εφήμερης δομής και της διαχρονικής δομής έχει ως εξής. Κάθε εφήμερος κόμβος αντιστοιχίζεται σε ένα σύνολο από διαχρονικούς κόμβους που καλείται οικογένεια (family). Τα μέλη της οικογένειας σχηματίζουν μια μονή διασυνδεδεμένη λίστα, η οποία διασυνδέεται μέσω των δεικτών αντιγράφων , με αύξουσα σειρά εκδόσεων. Το νεώτερο μέλος της οικογένειας που είναι τελευταίο στη λίστα καλείται live ενώ τα υπόλοιπα μέλη της οικογένειας dead. Κάθε έκδοση του εφήμερου κόμβου αντιστοιχεί σε ένα μέλος της οικογένειας, αν και πολλές εκδόσεις του εφήμερου κόμβου μπορούν να αντιστοιχούν στο ίδιο μέλος της οικογένειας. Οι live κόμβοι και οι νεώτερες τιμές πεδίων συγκροτούν τη νεώτερη έκδοση της εφήμερης δομής. Ένας δείκτης της διαχρονικής δομή που αναπαριστά ένα δείκτη στη νεώτερη έκδοση της εφήμερης δομής θα καλείται live δείκτης. Για να διευκολυνθεί ο "node copying" κάθε live δείκτης στη διαχρονική δομή έχει έναν αντίστοιχο αντίστροφο δείκτη (inverse pointer). Πχ Αν ο κόμβος $\bar{x}$ περιέχει ένα δείκτη προς

ένα live κόμβο ŷ , τότε ο ŷ περιέχει ένα δείκτη προς στο $\bar{x}$ αποθηκευμένο στα p πεδία προκατόχων.

Όπως και στη μέθοδο "fat node" χρησιμοποιούνται πίνακες πρόσβασης, ένα για κάθε όνομα δείκτη πρόσβασης, για την αποθήκευση των διάφορων εκδόσεων των δεικτών πρόσβασης. Χρησιμοποιώντας αυτούς τους πίνακες η πρόσβαση στον κόμβο εισόδου σε κάθε έκδοση κοστίζει 0(1) χρόνο.

Η δομή node copying η οποία δημιουργείται κατά τη διάρκεια της μεθόδου φαίνεται στο παρακάτω σχήμα.



Δομή node copying

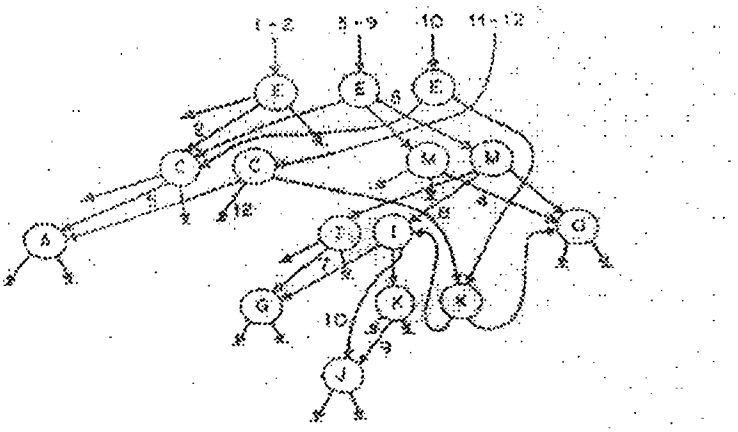
3.6.2 Προσομοίωση Λειτουργιών

Η πλοήγηση σε μια διαχρονική δομή είναι ακριβώς η ίδια με αυτή της μεθόδου "fat node". Για την προσομοίωση ενός εφήμερου βήματος πρόσβασης το οποίο εφαρμόζεται στην έκδοση i και ακολουθεί το δείκτη ενός πεδίου f ενός εφήμερου κόμβου x, στη διαχρονική δομή (node copying) ακολουθείται ο δείκτης με όνομα πεδίου f του αντίστοιχου διαχρονικού κόμβου ο οποίος έχει τη μεγαλύτερη στάμπα έκδοσης η οποία δεν είναι μεγαλύτερη από i. Επιλέγεται δηλαδή και ακολουθείται η πιο πρόσφατη τιμή του πεδίου f του δείκτη.

Η προσομοίωση ενός εφήμερου βήματος πρόσβασης μιας διαχρονικής δομής κοστίζει 0(1) σε χρόνο επειδή ο κόμβος έχει σταθερό μέγεθος και επομένως μπορεί να περιέχει περιορισμένο αριθμό από δείκτες.

Παράδειγμα

Εφαρμόζουμε ως παράδειγμα χρήσης της "node copying" , τη μέθοδο σε ένα δυαδικό δέντρο αναζήτησης. Σε αυτή την εφαρμογή μπορεί να γίνει μια σημαντική απλούστευση. Μπορούμε να παραλείψουμε τους αντίστροφους δείκτες στη διαχρονική δομή, επειδή στην εφήμερη δομή υπάρχει μόνο ένα μονοπάτι πρόσβασης σε κάθε κόμβο, λόγω του η πρόσβαση σε ένα κόμβο απαιτεί την πρόσβαση του προκατόχου του. Γενικά μπορούμε να αποφύγουμε τη χρήση των αντίστροφών δεικτών εφόσον στην εφήμερη δομή μπορεί να γίνει πρόσβαση σε ένα κόμβο αφού προηγηθεί πρόσβαση σε όλους του προκατόχους του. Στην περίπτωση του δυαδικού δέντρου αναζήτησης μπορούμε να παραλείψουμε επίσης τους δείκτες αντιγράφων και τις στάμπες έκδοσης στους κόμβους, εφόσον η "node copying" μέθοδος πραγματοποιηθεί με τον ελεγχόμενο τρόπο, με βάση τον οποίο κάθε φορά που γίνεται αλλαγή σε ένα κόμβο ξαναγυρίζουμε πίσω στους πρόγονους του. Στο παρακάτω σχήμα φαίνεται ένα διαχρονικό δυαδικό δέντρο αναζήτησης κατασκευασμένο με βάση τη μέθοδο "node-copying" χωρίς δείκτες αντιγράφων και στάμπες έκδοσης στους κόμβους. Οι λειτουργίες ενημέρωσης είναι προσθήκες και διαγραφές. Κάθε διαχρονικός κόμβος διαθέτει έναν επιπλέον πεδίο δείκτη, ο οποίος όπως θα δούμε παρακάτω επαρκεί για να διασφαλιστεί 0(m) όριο χώρου στο μέγεθος της διαχρονικής δομής, αφού στην περίπτωση αυτή p=1.



Ένα μερικώς διαχρονικό δέντρο αναζήτησης χρησιμοποιώντας τη μέθοδο node-copying, για την ακολουθία λειτουργιών ενημέρωσης. Οι στάμπες έκδοσης και οι δείκτες αντιγράφων έχουν παραληφθεί από τους κόμβους, επειδή δεν είναι απαραίτητοι.

3.7 Πλήρης Διαχρονικότητα (Full Persistence)

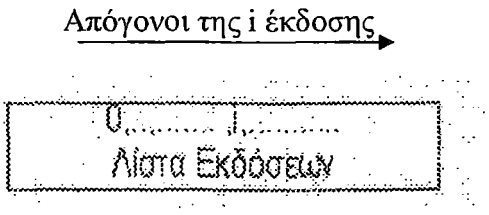
Αυτή η ενότητα ασχολείται με το δυσκολότερο πρόβλημα της μετατροπής μιας εφήμερης δομής σε πλήρως διαχρονική. Αναφορικά, χρησιμοποιώντας την προσέγγιση fat node η μετατροπή μιας εφήμερης διασυνδεδεμένης δομής σε πλήρως διαχρονική μπορεί να γίνει με κόστος χώρου χειρότερης περίπτωσης 0(1) για κάθε βήμα ενημέρωσης και κόστος χρόνου χειρότερης περίπτωσης O(logm) για κάθε βήμα

πρόσβασης και ενημέρωσης. Χρησιμοποιώντας μια παραλλαγή της node copying μεθόδου η οποία ονομάζεται "node splitting", η μετατροπή μιας εφήμερης διασυνδεδεμένης δομής με φραγμένο βαθμό εισόδου σε κάθε κόμβο σε πλήρως διαχρονική μπορεί να γίνει με επιμερισμένο κόστος χρόνου και χώρου O(1) για κάθε βήμα ενημέρωσης και κόστος χρόνου χειρότερης περίπτωσης O(1) για κάθε βήμα πρόσβασης.

3.7.1 Το Δέντρο Εκδόσεων (Version Tree) Και ηΛίστα Εκδόσεων (Version List)

Το πρώτο πρόβλημα που αντιμετωπίζουμε με την πλήρη διαχρονικότητα είναι ότι ενώ οι διάφορες εκδόσεις στη μερικώς διαχρονική δομή έχουν μια φυσική γραμμική διάταξη, οι εκδόσεις μιας πλήρως διαχρονικής δομής είναι μόνο μερικώς διατεταγμένες. Η μερική διάταξη προσδιορίζεται από ένα δέντρο εκδόσεων , του ποίου οι κόμβοι είναι οι εκδόσεις (0 έως m) με την έκδοση i να είναι ο πατέρας της έκδοσης j, εφόσον η j προκύπτει απευθείας από την ενημέρωση της έκδοσης i. Η έκδοση 0 είναι η ρίζα του δέντρου εκδόσεων .Η ακολουθία των ενημερώσεων η οποία δημιουργεί την έκδοση ί, αντιστοιχεί στο μονοπάτι του δέντρου εκδόσεων από τη ρίζα στην i.

Δυστυχώς, η έλλειψη γραμμικής διάταξης στις εκδόσεις καθιστά προβληματική την πλοήγηση στην αναπαράσταση της πλήρως διαχρονικής δομής. Για να εξαλείψουμε αυτήν τη δυσκολία, επιβάλουμε μια συνολική διάταξη στις εκδόσεις η οποία να είναι συνεπής με τη μερική διάταξη που ορίζεται από το δέντρο εκδόσεων. Αναπαριστούμε τη συνολική διάταξη με μια λίστα των εκδόσεων με την κατάλληλη σειρά. Θα καλούμε αυτή τη λίστα ως λίστα εκδόσεων (version list). Όταν μια νέα έκδοση, έστω η ί, δημιουργείται, εισάγουμε την ί, στη λίστα εκδόσεων αμέσως μετά τον πατέρα της (στο δέντρο εκδόσεων). Η προκύπτουσα λίστα ορίζει μια preorder στο δέντρο εκδόσεων .Με βάση αυτό συνεπάγεται ότι η λίστα εκδόσεων έχει την ακόλουθη κρίσιμη ιδιότητα: Για κάθε έκδοση ί, οι απόγονοι της i στο δέντρο εκδόσεων είναι συνεχόμενα παιδιά στη λίστα εκδόσεων ξεκινώντας από την i .



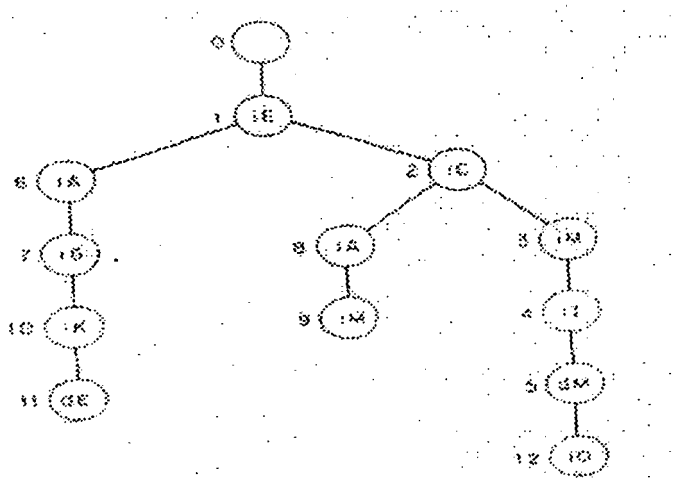
Σχήμα: Απόγονοι της i έκδοσης στη λίστα εκδόσεων

Θα αναφερόμαστε στην κατεύθυνση προς την αρχή της λίστας εκδόσεων (ενός δοσμένου αντικειμένου) ως leftward (προς τα αριστερά) και στην κατεύθυνση προς το τέλος της λίστας ως rightward (προς τα δεξιά).

Επιπρόσθετα για να πραγματοποιηθούν εισαγωγές στη λίστα εκδόσεων θα πρέπει να καθορίσουμε, δοθέντων δύο εκδόσεων ί, j εάν η ί προηγείται η έπεται της j στη λίστα εκδόσεων .Αυτό το πρόβλημα διάταξης της λίστας (list order problem) έχει απασχολήσει πολλούς επιστήμονες (Dietz, Tsakalidhs, Sleator). Τελικά ο Dietz και ο Sleator πρότειναν μια αναπαράσταση της λίστας η οποία υποστηρίζει ερωτήματα

σειράς (order queries) με κόστος χρόνου χειρότερης περίπτωσης 0(1) για κάθε ερώτημα και amortized κόστος χρόνου 0(1) για κάθε εισαγωγή στη λίστα. Οι ίδιοι επιστήμονες έχουν προτείνει επίσης μια πιο πολύπλοκη αναπαράσταση η οποία υποστηρίζει λειτουργίες ερωτημάτων και εισαγωγής με κόστος χρόνου χειρότερης περίπτωσης 0(1).

Στο παρακάτω σχήμα φαίνεται ένα δέντρο εκδόσεων και η αντίστοιχη λίστα εκδόσεων η οποία δημιουργείται.



Ένα δέντρο εκδόσεων. Κάθε κόμβος αναπαριστά μια λειτουργία ενημέρωσης. Το "i" και το "d" υποδεικνύουν εισαγωγή ή διαγραφή αντίστοιχα του *προσδιοριζόμενου αντικειμένου. Οι κόμβοι έχουν ετικέτες που προσδιορίζουν την αντίστοιχη λειτουργία. Η λίστα εκδόσεων που δημιουργείται είναι η 1,6,7,10,11,2,8,9,3,4,5,12.*

3.8 Η Μέθοδος "Fat Node"

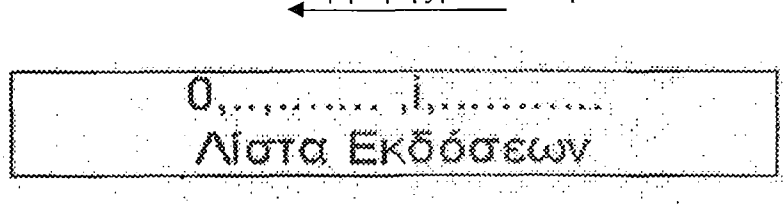
3.8.1 Η Δομή Δεδομένων

Έχοντας λύσει το θέμα της πλοήγησης στις διάφορες εκδόσεις θα ασχοληθούμε με το πώς κατασκευάζονται οι παχιοί κόμβοι ώστε να μετατραπεί μια διασυνδεδεμένη δομή σε πλήρως διαχρονική. Συγκεκριμένα κάθε παχύς κόμβος περιέχει τα ίδια πεδία όπως ένας εφήμερος κόμβος (για την αποθήκευση των αρχικών τιμών των πεδίων), καθώς και επιπλέον χώρο για ένα αυθαίρετο αριθμό επιπλέον τιμών πεδίων, καθένα με ένα όνομα πεδίου και μια στάμπα έκδοσης, καθώς και χώρο για τη στάμπα έκδοσης του ίδιου του κόμβου. Κάθε αρχικό πεδίο του παχιού κόμβου θεωρείται ότι έχει την ίδια στάμπα έκδοσης με αυτή του κόμβου στον οποίο περιέχεται.

3.8.2 Προσομοίωση Λειτουργιών

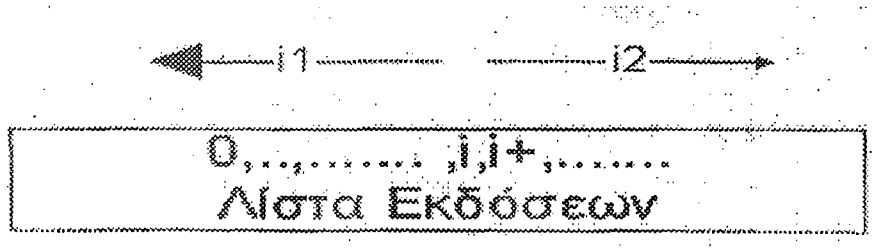
Η πλοήγηση σε μια διαχρονική δομή γίνεται με όμοιο τρόπο με αυτόν της μερικώς διαχρονικής δομής με τη διαφορά ότι οι συγκρίσεις των εκδόσεων γίνονται με βάση

τις θέσεις τους στη λίστα εκδόσεων και όχι με βάση τις αριθμητικές τους τιμές. Για να βρεθεί μια τιμή που αντιστοιχεί σε ένα πεδίο f στην έκδοση i ενός εφήμερου κόμβου x, βρίσκουμε τον παχύ κόμβο $\bar{x}$ που αντιστοιχεί στον x και επιλέγουμε από τις τιμές των πεδίων f αυτή με στάμπα έκδοσης που βρίσκεται δεξιότερα στη λίστα εκδόσεων αλλά όχι πιο δεξιά από την i. Επιλέγουμε δηλαδή ουσιαστικά τη τιμή του πεδίου που έχει έκδοση i ή αν δεν υπάρχει τιμή για την έκδοση i την τιμή που αντιστοιχεί σε έκδοση που είναι ο κοντινότερος πρόγονος της έκδοσης i. Η επιλογή της τιμής με βάση την έκδοση φαίνεται σχηματικά παρακάτω.



Επιλογή της τιμής κατά την πλοήγηση

Η ενημέρωση διαφέρει ελαφρά από αυτό που γίνεται στην περίπτωση της μερικής διαχρονικότητας, εξαιτίας της εισαγωγής νέων εκδόσεων στη μέση της λίστας εκδόσεων η οποία κάνει απαραίτητη την αποθήκευση δύο ενημερωμένων τιμών πεδίων για κάθε ενημέρωσης αντί για μία. Επομένως για να προσομοιώσουμε μια λειτουργία ενημέρωσης στην έκδοση i, προσθέτουμε αρχικά την έκδοση i στη λίστα εκδόσεων. Έπειτα για κάθε βήμα ενημέρωσης κάνουμε τα εξής. Αν το εφήμερο βήμα ενημέρωσης δημιουργεί ένα νέο εφήμερο κόμβο, εμείς δημιουργούμε τον αντίστοιχο νέο παχύ κόμβο με στάμπα έκδοσης i, γεμίζοντας τα αρχικά (original) πεδία του κατάλληλα. Στην περίπτωση που το εφήμερο βήμα ενημέρωσης αλλάζει το πεδίο f ενός εφήμερου κόμβου x κάνουμε την εξής διαδικασία. Έστω με i+ (αν υπάρχει) δηλώνουμε την έκδοση που βρίσκεται μετά την έκδοση i στη λίστα εκδόσεων .Για να προσομοιώσουμε το βήμα ενημέρωσης εντοπίζουμε στον παχύ κόμβο $\bar{x}$ που αντιστοιχεί στον x, τις τιμές v_1 και v_2 του πεδίου f τέτοιες ώστε η v_1 να έχει την δεξιότερη στάμπα έκδοσης η οποία να μη βρίσκεται δεξιά του i και η v_2 να έχει την αριστερότερη στάμπα έκδοσης η οποία να βρίσκεται δεξιά του i (στη λίστα εκδόσεων). Θεωρούμε i_1 και i_2 να είναι οι στάμπες έκδοσης των v_1 και v_2 αντίστοιχα.



Σχήμα : Σχηματική αναπαράσταση των i+, i_1, i_2.

Διακρίνουμε δύο περιπτώσεις.

1. Αν $i_1 = i$, αντικαθιστούμε την v_1 με την νέα τιμή του πεδίου f.

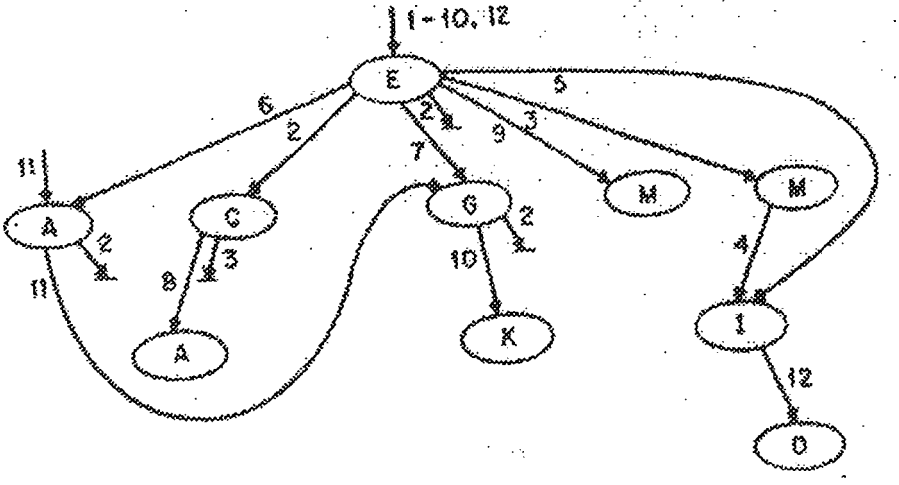
2. Αν $i_1 < i$, προσθέτουμε τη νέα τιμή του πεδίου f στον κόμβο $\bar{x}$ μαζί με το όνομα πεδίου του και τη στάμπα έκδοση του i. Αν επιπλέον $i_1 < i$ και $i+ < i_2$ (ή το i+ υπάρχει ενώ το i_2 όχι), προσθέτουμε στο $\bar{x}$ ένα αντίγραφο του v_i με όνομα πεδίου f και στάμπα έκδοσης του i+ .Αυτό διασφαλίζει ότι η νέα τιμή του πεδίου f θα χρησιμοποιηθεί μόνο στην έκδοση i, και η τιμή νι θα χρησιμοποιηθεί στις εκδόσεις i+ και μετά αλλά όχι στην i_2 έκδοση.

Στο τέλος μιας λειτουργίας ενημέρωσης αποθηκεύουμε τις τρέχουσες τιμές των δεικτών πρόσβασης στις i-οστές θέσεις των πινάκων πρόσβασης.

Έστω ν είναι η τιμή του πεδίου *f* η οποία έχει στάμπα έκδοσης *i* σε ένα παχύ κόμβο $\bar{x}$. Ορίζουμε έγκυρο διάστημα (valid interval) του *ν* να είναι το διάστημα των εκδόσεων στη λίστα εκδόσεων από *i* και πάνω μέχρι την επόμενη στάμπα έκδοσης της τιμής του πεδίου f, αλλά χωρίς να συμπεριλαμβάνεται η επόμενη στάμπα έκδοσης της τιμής του πεδίου f στο $\bar{x}$, ή από i και πάνω μέχρι το τέλος συμπεριλαμβανομένου της τελευταίας έκδοσης στη λίστα εκδόσεων αν δεν υπάρχει τέτοια επόμενη στάμπα έκδοσης. Τα έγκυρα διαστήματα των τιμών ενός πεδίου f σε ένα παχύ κόμβο $\bar{x}$ διαμερίζουν το διάστημα των εκδόσεων Η ορθότητα της μεθόδου fat node μπορεί εύκολα να αποδειχθεί χρησιμοποιώντας μια απόδειξη με επαγωγή πάνω στον αριθμό των βημάτων ενημέρωσης ώστε να δειχθεί η ορθότητα της κατάλληλης αντιστοίχησης μεταξύ της εφήμερης δομής και της διαχρονικής.

Παράδειγμα

Το σχήμα δείχνει ένα πλήρως διαχρονικό δυαδικό δέντρο αναζήτησης χρησιμοποιώντας τη μέθοδο "fat node". Σε αυτή την εφαρμογή μπορούμε να παραλείψουμε τις στάμπες έκδοσης στους κόμβους και τους αρχικούς null δείκτες.



Σχήμα : Ένα πλήρως διαχρονικό δέντρο αναζήτησης κατασκευασμένο χρησιμοποιώντας τη μέθοδο fat node. Οι στάμπες έκδοσης και οι αρχικοί null δείκτες έχουν παραληφθεί.

3.8.3 Ανάλυση Κόστος Χώρου Και Χρόνου

Η μέθοδος fat node διασφαλίζει πλήρη διαχρονικότητα με κόστος χώρου χειρότερης περίπτωσης O(1) για κάθε βήμα ενημέρωσης και κόστος χρόνου χειρότερης περίπτωσης $O(\log_m)$ για κάθε βήμα πρόσβασης και ενημέρωσης, δεδομένου ότι κάθε σύνολο τιμών πεδίων στον παχύ κόμβο αποθηκεύεται σε ένα δέντρο αναζήτησης, διατεταγμένο με βάση τη στάμπα έκδοσης. Ένα πιο ακριβή όριο για το κόστος χρόνου είναι το $O(\log_h)$ για κάθε βήμα πρόσβασης και ενημέρωσης με το h να είναι ο μέγιστος αριθμός των αλλαγών που γίνονται σε έναν εφήμερο κόμβο. Σημειώνουμε ότι η μέθοδος του fat node εφαρμόζεται ακόμη και αν ο βαθμός εισόδου των εφήμερων κόμβων δεν οριοθετείται από μια σταθερά.

3.9. Η Μέθοδος "Node Splitting"

Η μέθοδος "fat node" μπορεί να βελτιωθεί χρησιμοποιώντας μια παραλλαγή της μεθόδου "node copying" η οποία καλείται "node splitting". Η κύρια διαφορά μεταξύ της μεθόδου "node splitting" με την "node copying" είναι ότι στην πρώτη μόλις ένας κόμβος ξεχειλίσει ένα νέο αντίγραφο δημιουργείται και περίπου μισοί από τους επιπλέον δείκτες μετακινούνται από το παλιό αντίγραφο στο νέο αφήνοντας έτσι χώρο και στο παλιό και στο νέο αντίγραφο για μετέπειτα ενημερώσεις.

3.9.1. Η Επαυξημένη Εφήμερη Δομή

Ξεκινούμε την ανάπτυξη της μεθόδου "node splitting" αναφέροντας πως γίνεται η προσθήκη αντίστροφων δεικτών σε μια εφήμερη δομή. Αυτή η προσθήκη παράγει μια δομή την οποία θα ονομάζουμε επαυξημένη εφήμερη δομή (augmented ephemeral structure). Έστω p ένα σταθερό άνω όριο στον αριθμό των δεικτών που δείχνουν προς έναν εφήμερο κόμβο σε οποιαδήποτε έκδοση, συμπεριλαμβανομένου και των δεικτών πρόσβασης. Σε κάθε εφήμερο κόμβο προσθέτουμε p αντίστροφα πεδία (inverse fields) με νέα διακριτά ονόματα πεδίων για να κρατούν τους αντίστροφους δείκτες. Κάθε πεδίο αντιστρόφου είναι ικανό να κρατά είτε ένα δείκτη σε έναν κόμβο είτε ένα όνομα δείκτη πρόσβασης. Αυτό επιτρέπει την αναπαράσταση των αντίστροφων δεικτών για τους κόμβους πρόσβασης. Κατά τη διάρκεια μιας εφήμερης λειτουργίας ενημέρωσης ενημερώνουμε τους αντίστροφούς δείκτες ως εξής.

Θεωρούμε το βήμα ενημέρωσης το οποίο αλλάζει ένα πεδίο δείκτη *f* ενός κόμβου *x* ώστε να δείχνει σε ένα κόμβο *y*. Αν το πεδίο δεν ήταν null πριν τη αλλαγή αλλά περιείχε ένα δείκτη προς ένα κόμβο έστω το *z*, βρίσκουμε στον *z* ένα αντίστροφο πεδίο που περιέχει δείκτη στον *x* και το κάνουμε null. Ανεξαρτήτως αν το πεδίο *f* στον κόμβο *x* ήταν προηγουμένως null, βρίσκουμε στον *y* ένα null αντίστροφο πεδίο δείκτη και αποθηκεύουμε ένα δείκτη προς το *x*. Έπειτα αλλάζουμε το πεδίο f στο x ώστε να δείχνει το y. Επομένως κάθε βήμα το οποίο αλλάζει ένα δείκτη στην αρχική εφήμερη δομή αντιστοιχεί σε τρία (η πιθανώς περισσότερα) βήματα ενημέρωσης στην επαυξημένη δομή.

Θα πρέπει επίσης να ενημερώνουμε τους αντίστροφους δείκτες στην περίπτωση που αλλάζει ένας δείκτης πρόσβασης. Ας υποθέσουμε ότι ένας δείκτης πρόσβασης που ονομάζεται *a* αλλάζει ώστε να δείχνει σε ένα κόμβο x. Αν αυτός ο δείκτης δεν ήταν προηγουμένως null αλλά περιείχε ένα δείκτη σε ένα κόμβο, έστω τον z, τότε βρίσκουμε στον z ένα αντίστροφο πεδίο που περιέχει το όνομα a και το κάνουμε null. Βρίσκουμε στον κόμβο x ένα αντίστροφο πεδίο null και αποθηκεύουμε σε αυτό το όνομα a. Έπειτα αλλάζουμε το δείκτη πρόσβασης a ώστε να δείχνει στο x.

Υπάρχει μια ακόμη λεπτομέρεια στη μέθοδο. Στο τέλος κάθε λειτουργίας ενημέρωσης, εξετάζουμε κάθε δείκτη πρόσβασης. Αν ένας δείκτης πρόσβασης ο οποίος ονομάζεται a περιέχει ένα δείκτη σε ένα κόμβο που ονομάζεται x, βρίσκουμε στον x το αντίστροφο πεδίο το οποίο περιέχει το όνομα a και γράφουμε το όνομα a πάνω από αυτό. Ο λόγος για τον οποίο το κάνουμε αυτό είναι να διασφαλίζουμε ότι η αντιστροφή κάθε δείκτη πρόσβασης τίθεται σαφώς σε κάθε έκδοση, το οποίο είναι σημαντικό στη διαδικασία μετατροπής μιας δομής σε διαχρονική.

Η επαυξημένη εφήμερη δομή έχει τις ακόλουθες σημαντικές ιδιότητες συμμετρίας.

(1) Αν ένας κόμβος x περιέχει ένα δείκτη προς έναν κόμβο y, τότε ο y περιέχει δείκτη προς τον x.
(2) Ένας δείκτης πρόσβασης που ονομάζεται *a* δείχνει σε ένα κόμβο x αν και μόνο αν ο κόμβος x περιέχει το όνομα *a* σε ένα αντίστροφο πεδίο.

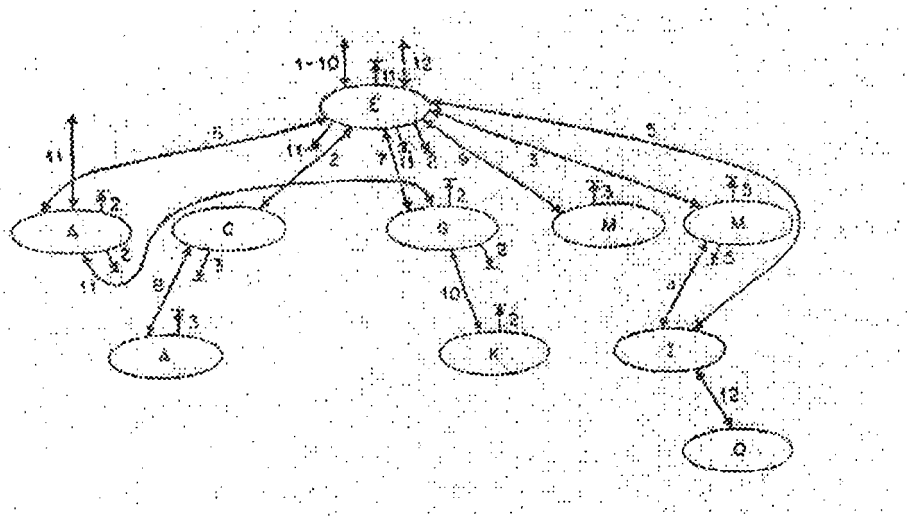
3.9.2. Η Μέθοδος "fat node revisited"

Εφαρμόζουμε τη μέθοδο "fat node" ώστε να κάνουμε την επαυξημένη εφήμερη δομή πλήρως διαχρονική. Στη δομή fat node που προκύπτει ισχύουν οι ακόλουθες ιδιότητες συμμετρίας.

(3) Αν ένα κόμβος $\bar{x}$ περιέχει ένα δείκτη σε ένα κόμβο $\bar{y}$ τέτοιο ώστε το έγκυρο διάστημα του δείκτη να περιλαμβάνει την έκδοση i, τότε το $\bar{y}$ περιέχει ένα δείκτη στο $\bar{x}$ του οποίου το έγκυρο διάστημα περιέχει επίσης την i.
(4) Ο πίνακας πρόσβασης που αντιστοιχεί σε ένα δείκτη πρόσβασης που ονομάζεται a περιέχει ένα δείκτη προς ένα κόμβο $\bar{x}$ στη θέση i, αν και μόνο αν ο κόμβος $\bar{x}$ περιέχει ένα πεδίο με όνομα a και έκδοση i (Αυτή η ιδιότητα ισχύει εξαιτίας της ρύθμισης για το αντίστροφο πεδίο του δείκτη πρόσβασης μετά από κάθε λειτουργία ενημέρωσης στην επαυξημένη εφήμερη δομή).

Το παρακάτω σχήμα δείχνει ένα πλήρως διασυνδεδεμένο δέντρο αναζήτησης με αντίστροφους δείκτες κατασκευασμένο με χρήση της "fat node" μεθόδου. Σε αυτή την εφαρμογή είναι χρήσιμο να γίνονται όλοι οι δείκτες σε ένα διαγραμμένο εφήμερο κόμβο null όταν πραγματοποιείται μια διαγραφή. Αυτό διασφαλίζει ότι κάθε κόμβος σε μια μη επαυξημένη εφήμερη δομή έχει το πολύ ένα εισερχόμενο δείκτη (Διαφορετικά θα είχαμε επιπρόσθετους εισερχόμενους δείκτες από τους σβησμένους κόμβους).

Μια ακόμη χρήσιμη παρατήρηση για τη δομή του "fat node" είναι η εξής. Ας υποθέσουμε ότι ένας παχύς κόμβος x περιέχει μια τιμή v με όνομα πεδίου f και έγκυρο διάστημα I. Έστω i είναι οποιαδήποτε έκδοση του συνόλου I εκτός από την αρχική (την στάμπα έκδοσης του v). Αν προσθέσουμε στο x ένα (πλεονάζον) αντίγραφό του v με όνομα πεδίου f και στάμπα έκδοσης i, δεν επηρεάζεται η αντιστοίχηση της πλοήγησης μεταξύ της εφήμερης και της διαχρονικής δομής ούτε οι συμμετρικές ιδιότητες. Τα δύο αντίγραφα του v έχουν έγκυρα διαστήματα που διαμερίζουν το I.



Σχήμα : Ένα πλήρως διαχρονικό δέντρο αναζήτησης μαζί με αντίστροφους (parent) δείκτες κατασκευασμένο με βάση τη fat node μέθοδο. Οι parent δείκτες φεύγουν από την κορυφή των κόμβων. Κάθε διπλό βέλος δηλώνει δύο δείκτες, ένα αντίστροφο του άλλου με την ίδια στάμπα έκδοσης. Οι στάμπες έκδοσης και οι αρχικοί null δείκτες έχουν παραληφθεί.

Σε μιά δομή δεδομένων "split node" οι συμμετρικές ιδιότητες που δημιουργούνται είναι οι ακόλουθες:

(5) Αν ο κόμβος x περιέχει ένα δείκτη προς έναν κόμβο y τέτοιο ώστε το έγκυρο διάστημα του δείκτη να περιέχει την έκδοση i, τότε κάποιος κόμβος της οικογένειας που περιέχει το y περιέχει ένα δείκτη σε κάποιο κόμβο της οικογένειας που περιέχει το x , ώστε να έχει (ο δείκτης) έγκυρο διάστημα το οποίο περιλαμβάνει την έκδοση i.
(6) Ο πίνακας πρόσβασης που αναπαριστά τον δείκτη πρόσβασης που ονομάζεται a περιέχει ένα δείκτη προς ένα κόμβο x στη θέση i, αν και μόνο αν κάποιος κόμβος της οικογένειας που περιέχει το x περιέχει το όνομα a με στάμπα έκδοσης i.

Χρειαζόμαστε δύο ακόμα ιδέες πριν αρχίσουμε να συζητάμε για τις λειτουργίες ενημέρωσης. Ένας δείκτης με στάμπα έκδοσης i ονομάζεται proper αν ο δείκτης δείχνει σε ένα κόμβο x του οποίου το έγκυρο διάστημα περιέχει το i. Ο δείκτης είναι overlapping αν το έγκυρο διάστημα του δεν περιέχεται μέσα στο έγκυρο διάστημα

του $\bar{x}$. Επεκτείνουμε αυτούς τους ορισμούς στους δείκτες πρόσβασης ως εξής: ο δείκτης ο οποίος είναι αποθηκευμένος στη θέση i ενός πίνακα πρόσβασης ονομάζεται proper αν δεικτοδοτεί ένα κόμβο χ του οποίου το έγκυρο διάστημα περιέχει το i διαφορετικά ονομάζεται overlapping. Με εξαίρεση τη διάρκεια των λειτουργιών ενημέρωσης όλοι οι δείκτες στη δομή "split node" είναι proper και δεν είναι overlapping. Αν ισχύει αυτό, οι συμμετρικές ιδιότητες (5) και (6) γίνονται όπως οι πιο ισχυρές ιδιότητες (3) και (4). Επιπλέον μπορούμε να πλοηγηθούμε σε μια δομή με τον ίδιο τρόπο με αυτόν της μεθόδου "fat node" με κόστος χρόνου χειρότερης περίπτωσης 0(1) για κάθε βήμα πρόσβασης.

3.9.3. Λειτουργίες Ενημέρωσης

Η μέθοδος "node splitting" επηρεάζει τη θέση ότι όλοι οι δείκτες είναι proper και nonoverlapping, και το δύσκολο μέρος διεξαγωγής λειτουργιών ενημέρωσης είναι η αποκατάσταση αυτή της θέσης την οποία θα καλούμε σταθερά δεικτών (pointer invariant). Κάθε λειτουργία ενημέρωσης αποτελείται από δύο φάσεις. Στην πρώτη φάση προσομοιώνονται τα βήματα της λειτουργίας ενημέρωσης σε μια επαυξημένη εφήμερη δομή, προσθέτοντας νέες τιμές στα νέα αντίγραφα των κόμβων .Αυτή η διαδικασία όμως ακυρώνει τη σταθερά δεικτών, και μπορεί να αναγκάσει κάποιους δείκτες να δείχνουν σε λανθασμένους κόμβους πέρα από τα μέρη των έγκυρων διαστημάτων τους. Η δεύτερη φάση διορθώνει αυτή τη ζημιά αντιγράφοντας διακριτικά τους δείκτες. Αφού όμως δημιουργούνται νέοι δείκτες αυτό μπορεί να απαιτεί node splitting, ώστε να δημιουργηθεί χώρος γι ' αυτούς, το οποίο μπορεί να ακυρώσει τη σταθερά δεικτών για κάποιους άλλους δείκτες. Γι ' αυτό και η δεύτερη φάση πραγματοποιεί εναλλαγή μεταξύ των διεργασιών pointer copying και node splitting η οποία σταματά όταν αποκαθίσταται για όλους τους δείκτες η σταθερά δεικτών .

Ας θεωρήσουμε τη λειτουργία ενημέρωσης i. Η πρώτη φάση προχωρά ακριβώς όπως στη μέθοδο "fat node" με τη διαφορά ότι οι καινούργιες τιμές πεδίων δεν αποθηκεύονται σε υπάρχοντες διαχρονικούς κόμβους αλλά σε νέα αντίγραφα των κόμβων .Όταν ένα εφήμερο βήμα ενημέρωσης δημιουργεί ένα νέο κόμβο $\bar{x}$, εμείς δημιουργούμε ένα νέο διαχρονικό κόμβο $\bar{x}$ με στάμπα έκδοσης i, τον οποίο γεμίζουμε με τα κατάλληλα πεδία πληροφοριών και τα αρχικά πεδία δεικτών (Τα τελευταία είναι null). Επόμενες αλλαγές σε αυτό τον κόμβο κατά τη διάρκεια της λειτουργίας ενημέρωσης i, γίνονται κάνοντας overwrite τα κατάλληλα πεδία, εκτός από ότι επιπλέον όποτε ένα πεδίο δείκτη f έχει τεθεί πρώτα σε τιμή διαφορετική από null, αν υπάρχει η i+ αποθηκεύουμε στο $\bar{x}$ ένα νέο null δείκτη με όνομα πεδίου f και στάμπα έκδοσης i+ (Αυτό το κάνουμε για να διατηρήσουμε τις συμμετρικές ιδιότητες).

Ας υποθέσουμε ότι ένα εφήμερο βήμα ενημέρωσης αλλάζει το πεδίο f ενός κόμβου ο οποίος δεν έχει δημιουργηθεί κατά τη διάρκεια της λειτουργίας ενημέρωσης i. Για να προσομοιώσουμε το βήμα, εντοπίζουμε το διαχρονικό κόμβο $\bar{x}$ που αντιστοιχεί στο $\bar{x}$. Αν ο $\bar{x}$ έχει στάμπα έκδοσης i, τροποποιούμε μόνο το κατάλληλο αρχικό πεδίο στο $\bar{x}$.Διαφορετικά προχωρούμε ως εξής. Αν το i ακολουθείται από άλλη έκδοση i+

στη λίστα εκδόσεων και το C(x) είτε δεν υπάρχει είτε έχει στάμπα έκδοσης μεγαλύτερη από i+, δημιουργούμε δύο νέους κόμβους, x' και x''. Επίσης θέτουμε το δείκτη αντιγράφου του x'' να δείχνει στο C(x), του x' να δείχνει στο x'' και του x να δείχνει στο x'. (Ετσι είναι τώρα c(x) = x', c(x') = x'', και c(x'') είναι η παλιά τιμή του c(x)). Δίνουμε στον κόμβο x' μια στάμπα έκδοσης i και στον κόμβο x'' μια στάμπα έκδοσης i+ .Γεμίζουμε τους αρχικούς δείκτες και τα πεδία πληροφοριών των x' και x'' συμβουλευόμενοι αυτά του x ως εξής. Κάθε πεδίο πληροφορίας των x' και x'' είναι ακριβώς όπως του x .Κάθε αρχικό πεδίο δείκτη των x', x'' , γεμίζεται με τη τιμή του πεδίου στο x , που έχει την δεξιότερη στάμπα έκδοσης όχι πιο δεξιά από i (όχι πιο δεξιά από i+) στη λίστα εκδόσεων. Σβήνουμε από τον x όλους τους επιπλέον δείκτες με στάμπες έκδοσης i+ (έχουν αντιγραφεί στα αρχικά πεδία του x'') και μετακινούμε αυτούς τους δείκτες με στάμπες έκδοσης δεξιά του i+ στα επιπλέον πεδία του x'' .Οι αλλαγές στα πεδία του x κατά τη διάρκεια της λειτουργίας ενημέρωσης i καταγράφονται στα αρχικά πεδία του x'= c(x) , ο οποίος αντιστοιχεί στην έκδοση i του x.

Η κατασκευή είναι όμοια αλλά απλούστερη αν i είναι η τελευταία έκδοση της λίστας εκδόσεων ή το c(x) έχει στάμπα έκδοσης του i+. Σε αυτή την περίπτωση δημιουργούμε ένα νέο κόμβο x' με στάμπα έκδοσης i, και κάνουμε το δείκτη αντιγράφου στο x' να δείχνει στο c(x) και του x να δείχνει στο x'. Επίσης αρχικοποιούμε τα πεδία πληροφορίας και τα αρχικά πεδία δεικτών του x' όπως περιγράφηκε παραπάνω. Οι αλλαγές στα πεδία του x κατά τη διάρκεια της λειτουργίας ενημέρωσης i, καταγράφονται στα αρχικά πεδία του x' = c(x) , ο οποίος αντιστοιχεί στην έκδοση i του x.

Κατά τη διάρκεια της πρώτης φάσης, κάνουμε επίσης μια λίστα κάθε κόμβου που έχει δημιουργηθεί κατά τη διάρκεια της φάσης και κάθε κόμβου του οποίου ο διάδοχος (successor) στη λίστα της οικογένειας έχει αλλάξει. Στο τέλος της πρώτης φάσης, επεξεργαζόμαστε κάθε κόμβο της λίστας ως εξής. Επιθεωρούμε όλους τους δείκτες προς το x , αλλάζοντας τους αν είναι απαραίτητο ώστε να γίνουν proper. Βρίσκουμε αυτούς τους δείκτες ακολουθώντας τους αντίστροφους δείκτες από το x, και εξετάζοντας τους κόμβους ή τις θέσεις του πίνακα πρόσβασης που δείχνουν. Κάθε δείκτης προς το x , αν δεν είναι proper μπορεί να γίνει proper αλλάζοντας τον ώστε να δείχνει στο c(x) ή στο c(c(x). Έτσι έχουμε τώρα μια δομή δεδομένων στην οποία όλοι οι δείκτες είναι proper αλλά όχι απαραίτητα nonoverlapping. Κατασκευάζουμε ένα σύνολο S ώστε να περιέχει κάθε κόμβο ο οποίος έχει τουλάχιστον ένα overlapping δείκτη. Οι ενδεχομένως overlapping δείκτες είναι ακριβώς αυτοί οι οποίοι έχουν επιθεωρηθεί και πιθανώς έχουν αλλάξει ώστε να γίνουν proper .

Αφού κατασκευαστεί το σύνολο S, ξεκινάμε τη δεύτερη φάση. Η φάση αυτή πραγματοποιεί επεξεργασία του συνόλου S, η οποία αποτελείται από τα εξής στάδια Αφαίρεση ενός αυθαίρετου κόμβου x , εκτέλεσή των παρακάτω τριών βημάτων και επανάληψη αυτών μέχρι το S να αδειάσει. Μετά το πέρα της φάσης όλοι οι δείκτες θα έχουν γίνει proper και nonoverlapping.

Βήμα 1 (πρόσθεσε νέους δείκτες). Κατασκεύασε μια λίστα L από τους αρχικούς και τους επιπλέον δείκτες του x και επεξεργάσου αυτούς τους δείκτες με σειρά αριστερά προς τα δεξιά με βάση τη στάμπα έκδοσης τους. Για την επεξεργασία ενός δείκτη χρειάζεται να καθοριστεί αν ο δείκτης είναι overlapping η όχι (ένας pointer προς το x θεωρείται ως nonoverlapping). Αν ο δείκτης δεν είναι overlapping συνέχισε με τον επόμενο δείκτη. Αν είναι θεωρούμε ότι έστω y είναι κόμβος στον οποίο δείχνει ο δείκτης και έστω i η στάμπα έκδοση του c(y) . Πρόσθεσε ένα δείκτη προς το c(y) , με στάμπα έκδοσης i και ίδιο όνομα πεδίου όπως ο δείκτης στο y , στη λίστα L. Ο παλιός δείκτης στο y (αλλά όχι απαραίτητα ο νέος δείκτης στο c(y)), είναι τώρα nonoverlapping. Συνέχισε με το επόμενο δείκτη.

Μετά το βήμα 1 όλοι οι δείκτες στο L είναι proper και nonoverlapping

Βήμα 2 (διάσπαση του x). Έστω i είναι η στάμπα έκδοσης του x. Αν όλοι οι δείκτες της λίστας L χωράνε στο x (υπάρχουν το πολύ 2e τέτοιοι δείκτες με στάμπα έκδοσης δεξιά του i στη λίστα έκδοσης) χώρεσε τους όλους στο x και πήδηξε το βήμα 3, επειδή δεν απαιτείται node splitting. Διαφορετικά δούλεψε από το πίσω μέρος του L, χωρίζοντας το σε ομάδες καθεμία από τις οποίες θα χωρά σε ένα (single) κόμβο ώστε να παραμένουν e επιπλέον πεδία δεικτών άδεια, εκτός από την ομάδα που αντιστοιχεί στην αρχή του L, η οποία θα πρέπει να χωρά σε ένα κόμβο χωρίς να αφήνονται επιπλέον πεδία δεικτών άδεια. Αποθήκευσε την πρώτη ομάδα δεικτών του x και κάθε διαδοχική ομάδα σε ένα νέο (δημιουργούμενο) κόμβο, στον οποίο θα δείχνει ο δείκτης αντιγράφου του προηγούμενου κόμβου. Ο τελευταίος νέος κόμβος θα δείχνει στο παλιό c(x) .Η στάμπα έκδοσης κάθε νέου κόμβου θα είναι η μικρότερη στάμπα έκδοσης του δείκτη που είναι αποθηκευμένος σε αυτόν. Όλοι οι δείκτες με αυτή τη στάμπα έκδοσης αποθηκεύονται σε αρχικά πεδία δεικτών, και όλοι οι δείκτες με δεξιότερες στάμπες έκδοσης σε επιπλέον πεδία. Επιπρόσθετες τιμές που χρειάζονται για τα αρχικά πεδία δεικτών και για τα πεδία πληροφορίας μπορούν να ληφθούν από τον προηγούμενο κόμβο στη λίστα της οικογένειας.

Μετά το βήμα 2 κάποιοι από τους δείκτες στο x μπορεί να είναι improper .

Βήμα 3 (κάνε όλους τους δείκτες proper). Με ψάξιμο στο x και σε όλους τους νέους δημιουργούμενους κόμβους, εντόπισε όλους τους δείκτες σε αυτούς στους κόμβους που δείχνουν στο x . Κάνε κάθε τέτοιο δείκτη proper κάνοντας το δείκτη να δείχνει στον κόμβο που τον περιέχει (κάθε τέτοιος δείκτης είναι εγγυημένο ότι είναι nonoverlapping). Ακολουθώντας τους δείκτες που περιέχονται στο x και σε νέους δημιουργούμενους κόμβους, εντόπισε όλους τους υπόλοιπους δείκτες που δείχνουν στο x (αλλά δεν βρίσκονται στην οικογένεια των κόμβων του x). Κάνε κάθε ένα από αυτούς proper κάνοντας το να δείχνει στον τελευταίο κόμβο μεταξύ του x και των νέων δημιουργούμενων κόμβων ο οποίος έχει στάμπα έκδοσης όχι δεξιότερη από αυτή του δείκτη. Αν ένας τέτοιος δείκτης είναι overlapping, πρόσθεσε τον κόμβο που τον περιέχει στο S αν δε υπάρχει ήδη εκεί.

Μετά το βήμα 3 όλοι οι δείκτες είναι proper και όλοι οι κόμβοι που περιέχουν overlapping δείκτες βρίσκονται στο S. Αποδεικνύεται με επαγωγή ότι στο τέλος της δεύτερης φάσης όλοι οι δείκτες θα είναι proper και overlapping. Μια απόδειξη με επαγωγή στον αριθμό των βημάτων ενημέρωσης δείχνει ότι η σωστή αντιστοίχηση

μεταξύ της εφήμερης δομής και της διαχρονικής δομής διατηρείται. Συνεπώς αποδεικνύεται και η ορθότητα της "node splitting" μεθόδου.

Παρατήρηση: Η "node splitting" μέθοδος όπως την περιγράψαμε δημιουργεί πάντα ένα ή δύο νέα αντίγραφα κάθε κόμβου ανάλογα με τη λειτουργία ενημέρωσης. Παρ' όλα αυτά αν οι ενημερώσεις σε ένα κόμβο κατά τη διάρκεια μια λειτουργία ενημέρωσης γίνουν μόνο σε πεδία δεικτών και όλοι οι νέοι δείκτες χωρούν στα επιπλέον πεδία του υπάρχοντος κόμβου, τότε δεν χρειάζεται να δημιουργηθούν νέα αντίγραφα του κόμβου. Αν οι νέοι δείκτες χωρούν στα επιπλέον πεδία του υπάρχοντος κόμβου και σε ακόμα ένα νέο κόμβο, τότε θα πρέπει να δημιουργηθεί μόνο ένα αντίγραφο αντί για δύο. Αυτή η βελτιστοποίηση εξοικονομεί κάποιο χώρο αλλά πολυπλέκει την υλοποίηση της μεθόδου επειδή αυξάνει των αριθμό των περιπτώσεων που θα πρέπει να ληφθούν υπόψη κατά τη διάρκεια της φάσης 1 της λειτουργίας ενημέρωσης.

3.9.4. Ανάλυση Κόστους Χώρου Και Χρόνου

Στην ενότητα αυτή αναλύουμε την απόδοση σε χώρο και σε χρόνο της μεθόδου "node splitting". Η φάση ένα της λειτουργίας ενημέρωσης δημιουργεί 0(1) νέους κόμβους σε κάθε βήμα ενημέρωσης. Τα βήματα 1, 2, 3 της φάσης δύο έχουν κόστος χρόνου 0(1) συν 0(1) χρόνο για κάθε νέο δημιουργούμενο κόμβο, εξαιτίας των ακόλουθων παρατηρήσεων.

(i) Κατά τη διάρκεια του βήματος 1, υπάρχουν το πολύ k+2e=0(1) μη επεξεργασμένοι δείκτες στη λίστα L.

(ii) Οι δείκτες που έχουν επιθεωρηθεί κατά τη διάρκεια του βήματος 3 οι οποίοι δεν βρίσκονται στο $\bar{x}$ ούτε στους νέους κόμβους βρίσκονται το πολύ σε k+2e διαφορετικούς κόμβους, το οποίο σημαίνει ότι το πολύ $(k+2e)^2=0(1)$ τέτοιοι δείκτες έχουν επιθεωρηθεί και το πολύ k+2e=0(1) κόμβοι προστίθενται στο S. Ο χρόνος για να γίνει κάθε επιθεωρούμενος δείκτης proper είναι ανάλογος του ένα συν τον αριθμό των κόμβων που δημιουργήθηκαν στο βήμα 2.

Επομένως ο συνολικός χρόνος που ξοδεύεται κατά τη διάρκεια μιας λειτουργίας ενημέρωσης είναι 0(1) για κάθε βήμα ενημέρωσης συν 0(1) για κάθε νέο δημιουργούμενο κόμβο. Θα καταλήξουμε σε 0(1) επιμερισμένο όριο στον αριθμό των κόμβων που δημιουργούνται σε κάθε βήμα ενημέρωσης, και με βάση αυτό το επιμερισμένο κόστος χρόνου και χώρου για κάθε βήμα ενημέρωσης θα είναι 0(1). Ορίζουμε το δυναμικό μιας δομής "split node" ότι είναι e/(e - k + 1) φορές ο αριθμός των διαχρονικών κόμβος μείον 1 / (e – k + 1) φορές ο αριθμός των άδειων επιπλέον πεδίων δεικτών, αριθμώντας μόνο το πολύ e άδεια επιπλέον πεδία δεικτών για κάθε κόμβο (υπενθυμίζουμε ότι απαιτείται e => k) .Ορίζουμε το επιμερισμένο κόστος χώρου για κάθε βήμα ενημέρωσης να είναι ο αριθμός των νέων κόμβων που δημιουργεί το βήμα συν την καθαρή αύξηση του δυναμικού που προκαλεί. Επειδή το αρχικό δυναμικό είναι 0 και το δυναμικό είναι πάντοτε θετικό, ο συνολικός αριθμός

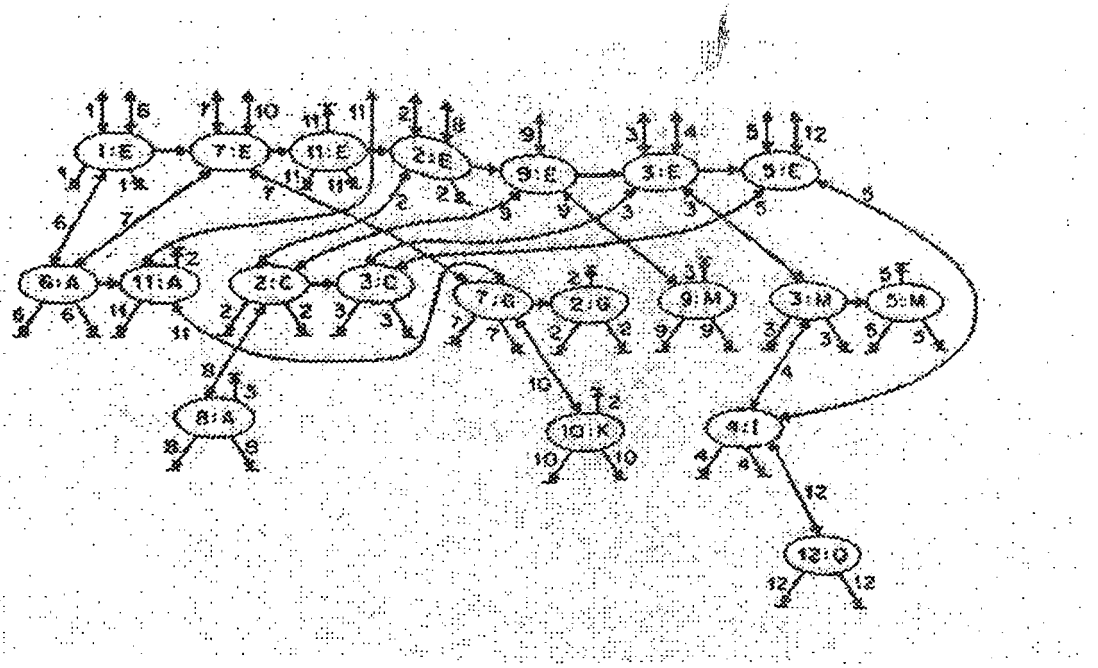
των κόμβων στη δομή "split node" είναι το πολύ ίσος με το συνολικό επιμερισμένο κόστος χώρου των λειτουργιών ενημέρωσης.

Ολοκληρώνουμε την ανάλυση δείχνοντας ότι το επιμερισμένο κόστος χώρου για μια λειτουργία ενημέρωσης είναι γραμμικό σε σχέση με τον αριθμό των βημάτων ενημέρωσης. Ας θεωρήσουμε μια λειτουργία ενημέρωσης που πραγματοποιεί u βήματα ενημέρωσης. Η φάση ένα μιας λειτουργίας ενημέρωσης έχει επιμερισμένο κόστος χώρου O(u) και δημιουργεί το πολύ 2u νέους κόμβους. Έστω l είναι ο αριθμός των νέων κόμβων που δημιουργήθηκαν κατά τη διάρκεια της φάσης δύο. Κάθε νέος κόμβος που δημιουργήθηκε στη φάση ένα ή δύο μπορεί τελικά να οδηγήσει στη δημιουργία μέχρι k νέων δεικτών (ένας νέος εισερχόμενος δείκτης αντιστοιχεί σε κάθε ένα από τους k αρχικούς εξερχόμενους δείκτες). Η δημιουργία ενός νέου κόμβου στη φάση δύο προσθέτει μηδέν στο δυναμικό. Το επιμερισμένο κόστος χώρου της φάσης δύο είναι γι' αυτό το πολύ 1 (για τους νέους δημιουργούμενους κόμβους) συν (2u+l)k / (e-k + 1) (για τα άδεια επιπλέον πεδία δεικτών που γεμίζονται με νέους δείκτες) μείον (e + l)l /(e – k + l) (επειδή κάθε νέος δημιουργούμενος κόμβος στη φάση δύο περιέχει τουλάχιστον e + 1 δείκτες που δε συνεισφέρουν στο δυναμικό, e επιπλέον πεδία δεικτών και ένα αρχικό πεδίο δείκτη). Το άθροισμα είναι 1+(2u+l) k / (e-k+ l)-(e + l)l/(e-k+ 1)=2uk/(e-k+ l)=0(u).

Επομένως το επιμερισμένο κόστος χώρου στη φάση δύο είναι 0(u). Συνεπώς το επιμερισμένο κόστος χώρου για ολόκληρη τη λειτουργία ενημέρωσης είναι 0(u), και άρα 0(1) για κάθε βήμα ενημέρωσης. Συνοψίζοντας, αναφέρουμε ότι δείξαμε ότι η "node splitting" μέθοδος μετατρέπει μια διασυνδεδεμένη δομή δεδομένων με φραγμένο βαθμό εισόδου σε πλήρως διαχρονική με επιμερισμένο κόστος χώρου και χρόνου 0(1) για κάθε βήμα ενημέρωσης και κόστος χρόνου χειρότερης περίπτωσης 0(1) για κάθε βήμα πρόσβασης. Στη λήψη αυτών των 0(1) ορίων αρκεί η χρησιμοποίηση της απλής μεθόδου για τη διατήρηση της λίστας εκδόσεων .

Παράδειγμα

Το σχήμα δείχνει ένα πλήρως διαχρονικό δέντρο αναζήτησης κατασκευασμένο χρησιμοποιώντας τη node-splitting μέθοδο, τροποποιημένη για να αποφεύγει το άσκοπο "node copying". Η τιμή που επιλέχθηκε για το e είναι 1 (κάθε κόμβος περιέχει τρεις αρχικούς δείκτες και δύο επιπλέον). Αυτή η επιλογή δεν επαρκεί για να διασφαλίσει γραμμικό χώρο για τη δομή δεδομένων, αλλά γίνεται ώστε να μπορεί να παρουσιαστεί ευκολότερα το παράδειγμα. Σε αυτό το παράδειγμα ακόμα και με e=1 δεν πραγματοποιείται node splitting κατά τη διάρκεια της δεύτερης φάσης.



Σχήμα : Ένα πλήρως διαχρονικό δέντρο αναζήτησης με αντίστροφους (parent) δείκτες κατασκευασμένο με χρήση της "node splitting" μεθόδου. Κάθε δείκτης αριθμείται με τη στάμπα έκδοσης του. Οι κόμβοι επίσης αριθμούνται με τις στάμπες έκδοσης. Οι δείκτες αντιγράφων αρχίζουν από τις δεξιές μεριές των κόμβων και καταλήγουν στις αριστερές μεριές. Τα πεδία δεικτών των διαγραμμένων κόμβων γίνονται null στην περίπτωση της διαγραφής.

4ο Κεφάλαιο

"Εφαρμογές σε Χρονικές Β.Δ"

4ο Κεφάλαιο

4.1 Εισαγωγή

Σε αυτό το τμήμα παρουσιάζονται οι διαφορετικές *τεχνικές δεικτοδότησης* που αφορούν την υποστήριξη της αποδοτικής πρόσβασης σε *χρονικές Βάσεις Δεδομένων*. Η σύγκριση των τεχνικών είναι βασισμένη σε μία συλλογή μετρικών που αποτελείται από τα σημαντικότερα κριτήρια απόδοσης, συμπεριλαμβανομένου του χώρου που καταναλώνεται, της επεξεργασίας πράξεων ενημέρωσης και του χρόνου ερώτησης για τα αντιπροσωπευτικά ερωτήματα. Πρόσθετα κριτήρια που εξετάζονται είναι οι σελιδοποιήσεις ενός δείκτη, η δυνατότητα να συγκεντρωθούν τα σχετικά στοιχεία μαζί και η δυνατότητα να χωριστούν αποτελεσματικά τα τρέχοντα από τα παλαιά στοιχεία Η σύγκριση είναι βασισμένη στην ανάλυση χειρότερης περίπτωσης και ως εκ τούτου δεν γίνεται καμία υπόθεση για την κατανομή δεδομένων και για την συχνότητα εμφάνισης κάθε είδους ερώτησης. Στη συνέχεια παρουσιάζεται η σύγκριση από κάποια αντιπροσωπευτικά είδη ερωτήσεων και προσδιορίζονται τα δύσκολα προβλήματα στην πρόσβαση των χρονικών δεδομένων και περιγράφεται το πώς οι διαφορετικές μέθοδοι προσπαθούν να λύσουν αυτά τα προβλήματα

4.2 Κριτήρια Σύγκρισης

Στο τμήμα αυτό παραθέτονται και εξηγούνται τα σημεία πάνω στα οποία βασίζεται η σύγκριση των διαφόρων μεθόδων προσπέλασης που ακολουθεί.

- **Είδη ερωτήσεων**

Όπως, έχει ήδη αναφερθεί η απόδοση κάθε τεχνικής δεικτοδότησης θα εξεταστεί ως προς κάποιο είδος ερώτησης. Σε αυτό το σημείο, λοιπόν, παρουσιάζονται τα είδη ερωτήσεων .

Α. Ερωτήσεις στο πεδίο ορισμού του χρόνου συναλλαγής.

Κλάση 1

Δοσμένου ενός συνεχόμενου χρονικού διαστήματος Τ, να βρεθούν όλα τα αντικείμενα που υπάρχουν στη Βάση κατά τη διάρκεια αυτού του χρονικού διαστήματος.

Κλάση 2

Δοσμένης μιας περιοχής κλειδιού και ενός συνεχούς χρονικού διαστήματος Τ, να βρεθούν όλα τα αντικείμενα με κλειδιά στη δοσμένη περιοχή και που υπάρχουν στη Βάση κατά τη διάρκεια αυτού του χρονικού διαστήματος.

Κλάση 3

Δοσμένης μιας περιοχής κλειδιού, να βρεθεί η ιστορία όλων των αντικειμένων στην περιοχή αυτή.

Παράδειγμα της πρώτης κλάσης ερωτήσεων.

Μια ειδική περίπτωση της κλάσης αυτής είναι όταν το χρονικό διάστημα T μειώνεται σε ένα και μόνο χρονικό σημείο στον άξονα του χρόνου συναλλαγής. Τότε η ερώτηση χαρακτηρίζεται transaction pure timeslice ερώτηση και ορίζεται ως εξής. Δοσμένου ενός χρονικού σημείου, να βρεθούν όλα τα αντικείμενα που είναι ζωντανά τη χρονική στιγμή αυτή. Π.χ. " Να βρεθούν όλοι οι εργαζόμενοι που δουλεύουν στην εταιρία τη χρονική στιγμή t". Μια μέθοδος που λύνει αποτελεσματικά μια time_slice ερώτηση, λύνει και μια πιο γενική ερώτηση διαστήματος. Για αυτό η timeslice ερώτηση είναι αντιπροσωπευτική της πρώτης κλάσης.

Παράδειγμα της δεύτερης κλάσης ερωτήσεων.

Αντιπροσωπευτική περίπτωση θεωρείται αυτή που το χρονικό διάστημα μειώνεται σε ένα και μόνο χρονικό σημείο. Αυτή η ερώτηση λέγεται transaction range timeslice ερώτηση. Π.χ. "Να βρεθούν όλοι οι εργαζόμενοι που δουλεύουν στην εταιρία τη χρονική στιγμή t και που το αναγνωριστικό τους ανήκει στην περιοχή K".

Παράδειγμα της τρίτης κλάσης ερωτήσεων .

Αντιπροσωπευτική περίπτωση θεωρείται αυτή που η περιοχή κλειδιού μειώνεται σε μια και μόνο περιοχή κλειδιού. Π.χ " Να βρεθεί η ιστορία της μισθοδοσίας του εργαζόμενου με αναγνωριστικό K". Αυτού του είδους η ερώτηση λέγεται transaction pure key ερώτηση και ακολουθεί το σκεπτικό ότι αν ο εργαζόμενος με αναγνωριστικό K υπήρξε ποτέ στην εταιρία, η απάντηση θα είναι η ιστορία της μισθοδοσίας του εργαζομένου, διαφορετικά η απάντηση θα είναι κενή.

Μια ειδική περίπτωση της pure key ερώτησης είναι η pure key ερώτηση με predicate. Σε αυτό το είδος των ερωτήσεων πρέπει να δίνεται και ένα στιγμιότυπο του εργαζομένου, για να απαντηθεί η ερώτηση. π.χ. " Να βρεθεί η ιστορία της μισθοδοσίας του εργαζομένου με αναγνωριστικό K που υπήρχε τη χρονική στιγμή t.

Εκτός από την παραπάνω κατηγοριοποίηση των ερωτήσεων κατηγοριοποιούνται τα είδη των ερωτήσεων και ως εξής. Στις ερωτήσεις εύρους (range queries) και στις ερωτήσεις ακριβούς ταιριάσματος (exact match) . Στην πρώτη κατηγορία ανήκουν οι range timeslice, οι pure timeslice και οι pure key with predicate ερωτήσεις. Σε αυτό το είδος ερωτήσεων ο χαρακτηρισμός ζωντανές αντιστοιχεί σε ερωτήσεις δύο περιοχών. Συγκεκριμένα ζωντανές λέγονται οι ερωτήσεις που για να απαντηθούν γίνονται οι δύο παρακάτω έλεγχοι, start.time $\leq$ tκαι end.time $\geq$ t Στη δεύτερη κατηγορία ανήκουν οι pure key ερωτήσεις.

Παρατήρηση: Η κλάση 1 μπορεί να θεωρηθεί σαν ειδική περίπτωση της κλάσης 2, αν δεν δοθεί συγκεκριμένη περιοχή κλειδιού. Ενώ η κλάση 3 μπορεί να θεωρηθεί σαν ειδική περίπτωση της 2, αν δε δοθεί συγκεκριμένο χρονικό διάστημα. Κάθε είδος προσπέλασης δεν έχει την ίδια απόδοση για κάθε είδος ερώτησης.

Β. Ερωτήσεις στο valid time πεδίο ορισμού.

Οι βασικές ερωτήσεις στο πεδίο ορισμού έγκυρου χρόνου είναι οι εξής:
Pure timeslice: "Να βρεθούν όλα τα συμβόλαια που είναι σε ισχύ τη χρονική στιγμή u"

Range timeslice: "Να βρεθούν όλα τα συμβόλαια με αριθμούς στην περιοχή K και που είναι σε ισχύ τη στιγμή u".

Γ .Ερωτήσεις και στα δύο πεδία ορισμού.

Μια διχρονική Β.Δ. επιτρέπει ερωτήσεις και στις δύο διαστάσεις του χρόνου. Π.χ. "Να βρεθούν όλα τα συμβόλαια που ήταν σε ισχύ τη χρονική στιγμή u= Ιανουάριος 1, 1994, όπως καταχωρήθηκε στη βάση δεδομένων την transaction_time t= May 1 1993".Από όλα τα συμβόλαια της συλλογής C(t) για t= May 1, 1993, η ερώτηση ανακτά μόνο τα συμβόλαια που θα είναι σε ισχύ τον Ιανουάριο του 1994.

Οι ερωτήσεις της πρώτης και της δεύτερης κλάσης σχετίζονται με ερωτήσεις βασιζόμενες στην έννοια της τομής. Δηλαδή, οι απαντήσεις αποτελούνται από αντικείμενα των οποίων τα διαστήματα περιέχουν το χρονικό σημείο που δίνεται στην ερώτηση ή ένα διάστημα που δίνεται στην ερώτηση. Υπάρχουν, όμως, και άλλες κατηγορίες ερωτήσεων στις οποίες η απάντηση αποτελείται από αντικείμενα με διαστήματα πριν ή μετά ενός χρονικού σημείου, κ.α.

4.2.1 Three Entry Notation

Η three entry notation είναι ένας τρόπος συμβολισμού των ερωτήσεων. Το γενικό πρότυπο είναι το εξής: Key / valid / transaction. Αυτή η σημειογραφία δείχνει ποια χαρακτηριστικά αντικειμένων εμπλέκονται στην ερώτηση και με τι τρόπο. Κάθε πεδίο (key , value, transaction) μπορεί να συμπληρωθεί με ένα από τα παρακάτω : point, range, *, -. Το σύμβολο * σημαίνει ότι οποιαδήποτε τιμή είναι αποδεκτή σε αυτό το πεδίο, ενώ το σύμβολο -σημαίνει ότι αυτό το πεδίο δεν συμμετέχει στην συγκεκριμένη ερώτηση. Αν key = point, τότε σημαίνει ότι ο χρήστης έχει καθορίσει μια μοναδική τιμή για πεδίο κλειδιού του αντικειμένου. Αν valid ή transaction = point, τότε σημαίνει ότι ο χρήστης έχει ορίσει ένα μοναδικό χρονικό σημείο ως valid_time ή ως transaction_time πεδίο ορισμού.

Για να γίνει κατανοητός αυτός ο τρόπος συμβολισμού των ερωτήσεων δίνονται κάποια παραδείγματα:

*/- / point : transaction pure timeslice ερώτηση

range / point / -: valid range timeslice ερώτηση

point / -/ * : transaction pure key ερώτηση

* / point / range : "Να βρεθούν όλα τα συμβόλαια της εταιρίας που ήταν σε ισχύ τον u= Ιανουάριο 1, 1994, όπως καταχωρήθηκε στη βάση κατά το χρονικό διάστημα συναλλαγής T_α = «May 1- May 20, 1993".

4.2.2 Απόδοση Μεθόδων Προσπέλασης

Η απόδοση μιας μεθόδου προσπέλασης μετριέται με βάση τα παρακάτω:

- Τον χώρο αποθήκευσης που απαιτείται για τη φυσική αποθήκευση των εγγραφών των δεδομένων.

- Τον χρόνο που απαιτείται για την ενημέρωση δομών δεδομένων της μεθόδου σαν αποτέλεσμα μιας αλλαγής – χρόνος επεξεργασίας ενημέρωσης (update processing time).

- Τον χρόνο ερώτησης για κάθε ένα από τα βασικά είδη των ερωτήσεων (query time).

Μια μέθοδος προσπέλασης έχει δύο *τρόπους λειτουργίας*:

Λειτουργία Ενημέρωσης: σε αυτή τη φάση δεδομένα εισάγονται, αλλάζουν ή διαγράφονται.

Λειτουργία Ερώτησης: σε αυτή τη φάση διατυπώνονται και απαντώνται ερωτήσεις, χρησιμοποιώντας αυτή τη μέθοδο προσπέλασης.

4.2.3 Διαδικασία Ενημέρωσης

Σε μια μέθοδο προσπέλασης χρόνου συναλλαγής, η είσοδος της μεθόδου προσπέλασης για να πραγματοποιηθεί μία ενημέρωση, θα είναι μια χρονική στιγμή t και όλες οι αλλαγές που συνέβησαν στα δεδομένα αυτή τη χρονική στιγμή. Σε μία ενημέρωση μπορεί να δίνεται επιπλέον σαν είσοδος το μοναδικό κλειδί του αντικειμένου, καθώς και το είδος της αλλαγής (προσθήκη, διαγραφή, ενημέρωση).

Σε μια μέθοδο προσπέλασης έγκυρου χρόνου, η είσοδος της μεθόδου προσπέλασης θα είναι οι αλλαγές, καθώς και τα χρονικά διαστήματα που επηρεάζουν.

Σε μια διχρονική μέθοδο προσπέλασης που καθορίζεται ο χρόνος αλλαγής μαζί με τις αλλαγές και τα επηρεαζόμενα αντικείμενα διαστήματος, η είσοδος είναι όμοια με αυτή των μεθόδων προσπέλασης χρόνου συναλλαγής.

4.2.4 Χώρος

Για μια διχρονική μέθοδο ή μέθοδο χρόνου συναλλαγής, ο χώρος που απαιτείται είναι συνάρτηση του αριθμού των αλλαγών (n) κατά την εξέλιξη, π.χ. αν σε μια Β.Δ. υπάρχει μόνο μια εγγραφή και γίνουν σε αυτή 1000 ενημερώσεις τότε το n=1000. Αν σε μια Β.Δ. γίνουν 1000 εισαγωγές μόνο, τότε n=1000. Αν σε μια Β.Δ. γίνουν 1000 εισαγωγές και στη συνέχεια 1000 διαγραφές, τότε το n=2000. Δηλαδή, το n αντιστοιχεί στις ελάχιστες πληροφορίες που απαιτούνται για την αποθήκευση της εξέλιξης. Για μια μέθοδο έγκυρου χρόνου, ο χώρος που απαιτείται είναι συνάρτηση

του I, όπου I είναι ο αριθμός των αντικειμένων διαστήματος που πρόσφατα αποθηκεύτηκαν στη μέθοδο. π.χ. το μέγεθος της συλλογής.

4.3 Κατηγοριοποίηση Μεθόδων και Σύγκριση

Στο μέρος αυτό, παρουσιάζονται συγκεκριμένες τεχνικές δεικτοδότησης για κάθε είδος μεθόδου προσπέλασης. Για κάθε τεχνική δίνεται μια μικρή περιγραφή, το κόστος της σε απαιτούμενο χώρο, σε χρόνο επεξεργασίας ερωτήσεων και σε χρόνο ενημέρωσης και τελικά προκύπτει για ποιο είδος ερώτησης είναι κατάλληλη η μέθοδος.

4.3.1 Είδη Μεθόδων Προσπέλασης Χρόνου Συναλλαγής

Χαρακτηριστικό και προϋπόθεση για να είναι μια μέθοδος προσπέλασης χρόνου συναλλαγής είναι το εξής. Οι αλλαγές να φτάνουν σε αύξουσα χρονική σειρά. Αυτή η ιδιότητα επηρεάζει σημαντικά τη διαδικασία ενημέρωσης μιας μεθόδου. Αν γίνει προσπάθεια υποστήριξης αλλαγών εκτός σειράς (χαρακτηριστικό των μεθόδων έγκυρου χρόνου), το κόστος ενημέρωσης γίνεται σημαντικά υψηλότερο, πρακτικά απαγορευτικό.

1. key-only Μέθοδοι Προσπέλασης

Τα δεδομένα οργανώνονται με βάση το κλειδί. Δηλαδή, όλες οι εκδοχές ενός συγκεκριμένου κλειδιού τοποθετούνται κοντά λογικά ή φυσικά. Μια τέτοιου είδους οργάνωση κάνει τις αντίστοιχες μεθόδους πιο αποτελεσματικές για transaction pure key ερωτήσεις. Αντιπροσωπευτικές τεχνικές αυτής της κατηγορίας είναι οι εξής.

- *Reverse chaining*

Στην μέθοδο αυτή οι προηγούμενες εκδοχές ενός κλειδιού είναι ενωμένες μεταξύ τους (με δείκτες) σε αντίστροφη χρονολογική σειρά. Είναι μια τεχνική που ξεχωρίζει τα περασμένα δεδομένα από τα τρέχοντα δεδομένα. Αυτό το καταφέρνει διατηρώντας δύο B+tree. Ένα "front" και ένα "back". Το front B+tree δεικτοδοτεί τις εγγραφές που είναι ζωντανές και το back B+tree δεικτοδοτεί τις τελευταίες εκδοχές κλειδιών , των οποίων οι εγγραφές δεν είναι ζωντανές. Η μέθοδος αυτή απαιτεί χώρο O(n/B), χρόνο ενημέρωσης $O(\log_B n)$, $O(\log_B n+k)$ χρόνο ερώτησης, όπου n είναι ο αριθμός των αλλαγών και στην περίπτωση αυτή ισούται με τον αριθμό των εκδοχών και k ο αριθμός των περασμένων εκδοχών του κλειδιού.

- *Accession lists*

Η τεχνική αυτή είναι μια βελτίωση της reverse chaining. Οι εκδοχές κάθε κλειδιού τοποθετούνται μαζί σε μια accession list. Η διαφορά με τη reverse chaining είναι ότι η αναζήτηση μιας εκδοχής ενός κλειδιού δε γίνεται με σειριακή αναζήτηση πάνω στην αλυσίδα ενός κλειδιού, αλλά μέσω ενός ευρετηρίου. Δηλαδή, διατηρείται ένα ευρετήριο πάνω σε κάθε αλυσίδα. Με αυτό τον τρόπο μειώνεται ο χρόνος ερώτησης.

Έτσι, η μέθοδος αυτή κοστίζει χώρο O(n/B), O($\log_B$n) χρόνο ενημέρωσης και O($\log_B$n + $\log_B a$) χρόνο ερώτησης.

- *Time sequence arrays*

Η μέθοδος αυτή διατηρεί ένα διδιάστατο πίνακα. Ο πίνακας αυτό έχει μια γραμμή για κάθε κλειδί που έχει ποτέ δημιουργηθεί, ενώ κάθε στήλη του αντιστοιχεί σε ένα transaction_time instant. Έτσι το στοιχείο (x, y) του πίνακα περιέχει την εκδοχή του κλειδιού x την στιγμή y. Η μέθοδος αυτή έχει προφανώς πολύ μικρό χρόνο ερώτησης αλλά κοστίζει πολύ σε χώρο, ο οποίος είναι O(n^2), και χρόνο ενημέρωσης, που είναι O(n).

- *C lists*

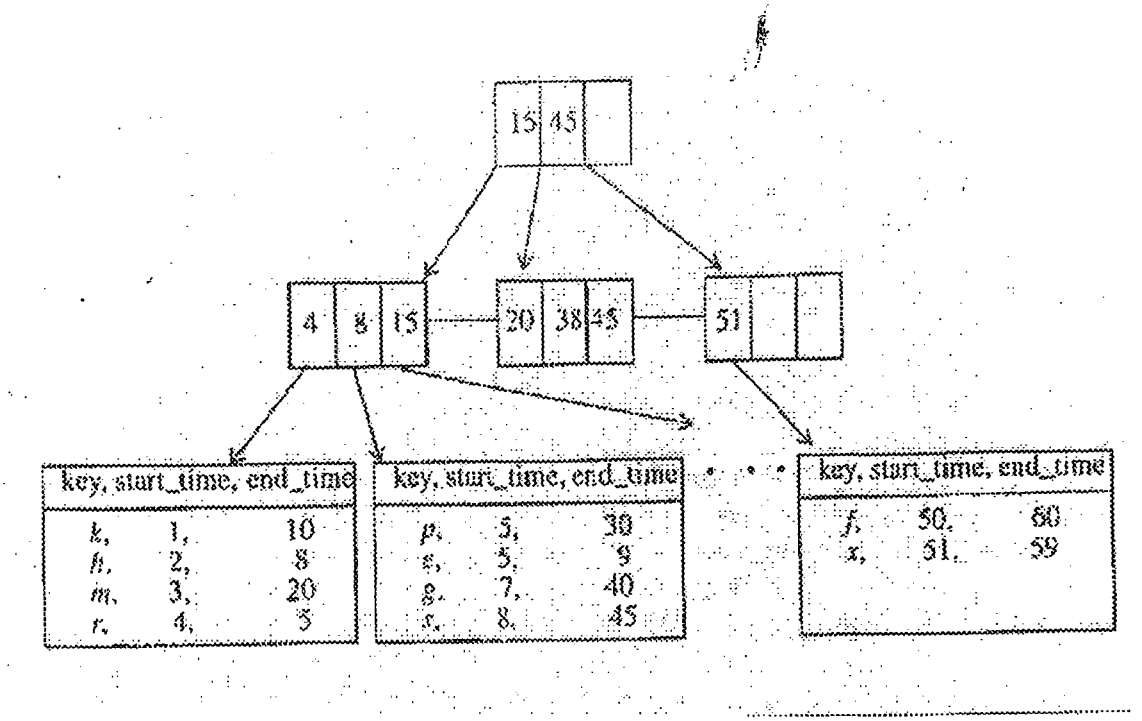
Είναι θεωρητικά η βέλτιστη λύση για transaction pure key with predicate ερωτήσεις. Οι C-lists μοιάζουν με τις accession lists στο ότι και οι δύο ταξινομούν όλες τις εκδοχές ενός κλειδιού μαζί. Υπάρχουν όμως δύο βασικές διαφορές. Πρώτον η προσπέλαση κάθε μίας από τις C-lists γίνεται με την multiversion access μέθοδο (MVAS) και η διατήρηση της είναι πιο πολύπλοκη. Η μέθοδος αυτή απαιτεί χώρο O(n/B), χρόνο ενημέρωσης O($\log_B$n) και χρόνο ερώτησης O($\log_B$n+a/B).

2 Time-Only Μέθοδοι Προσπέλασης

Τα δεδομένα οργανώνονται με βάση τον χρόνο συναλλαγής. Μια τέτοιου είδους οργάνωση κάνει τις αντίστοιχες μεθόδους πιο αποτελεσματικές για transaction pure timeslice και transaction range timeslice ερωτήσεις. Όπως έχει ήδη αναφερθεί, σε αυτού του είδους τις μεθόδους οι αλλαγές γίνονται με χρονολογική σειρά, με αποτέλεσμα να έχουμε σταθερό χρόνο ενημέρωσης, καθώς οι αλλαγές απλά προσαρτώνται στο τέλος του "history log". Αυτό είναι μεγάλο πλεονέκτημα ιδιαίτερα για εφαρμογές όπου οι αλλαγές είναι συχνές και η Βάση Δεδομένων πρέπει να ακολουθεί αυτές τις αλλαγές on-line.

- *Append only tree*

Στη μέθοδο αυτή τα δεδομένα ταξινομούνται με το append only tree, το οποίο είναι ένα πολυδιακλαδιζόμενο δέντρο αναζήτησης. Η δεικτοδότηση γίνεται ως προς τις χρονικές στιγμές έναρξης, όπως φαίνεται και στο παρακάτω σχήμα. Δηλαδή, τα φύλλα περιέχουν μόνο τη χρονική στιγμή έναρξης ενός χρονικού διαστήματος. Κάθε φύλλο δείχνει σε σελίδες αρχείου, στις οποίες οι εγγραφές είναι ταξινομημένες με βάση τη χρονική στιγμή έναρξής τους. Νέες εγγραφές προστίθενται μόνο στο δεξιότερο φύλλο του δέντρου. Στην μέθοδο αυτή το μειονέκτημα είναι ότι χρειάζεται να είναι γνωστό εκ των προτέρων η χρονική στιγμή τέλους. Η τεχνική αυτή δεν ξεχωρίζει τα περασμένα από τρέχοντα δεδομένα και κοστίζει O(n/B) χώρο, O($\log_B$n) χρόνο ενημέρωσης και O(n/B) χρόνο ερώτησης.

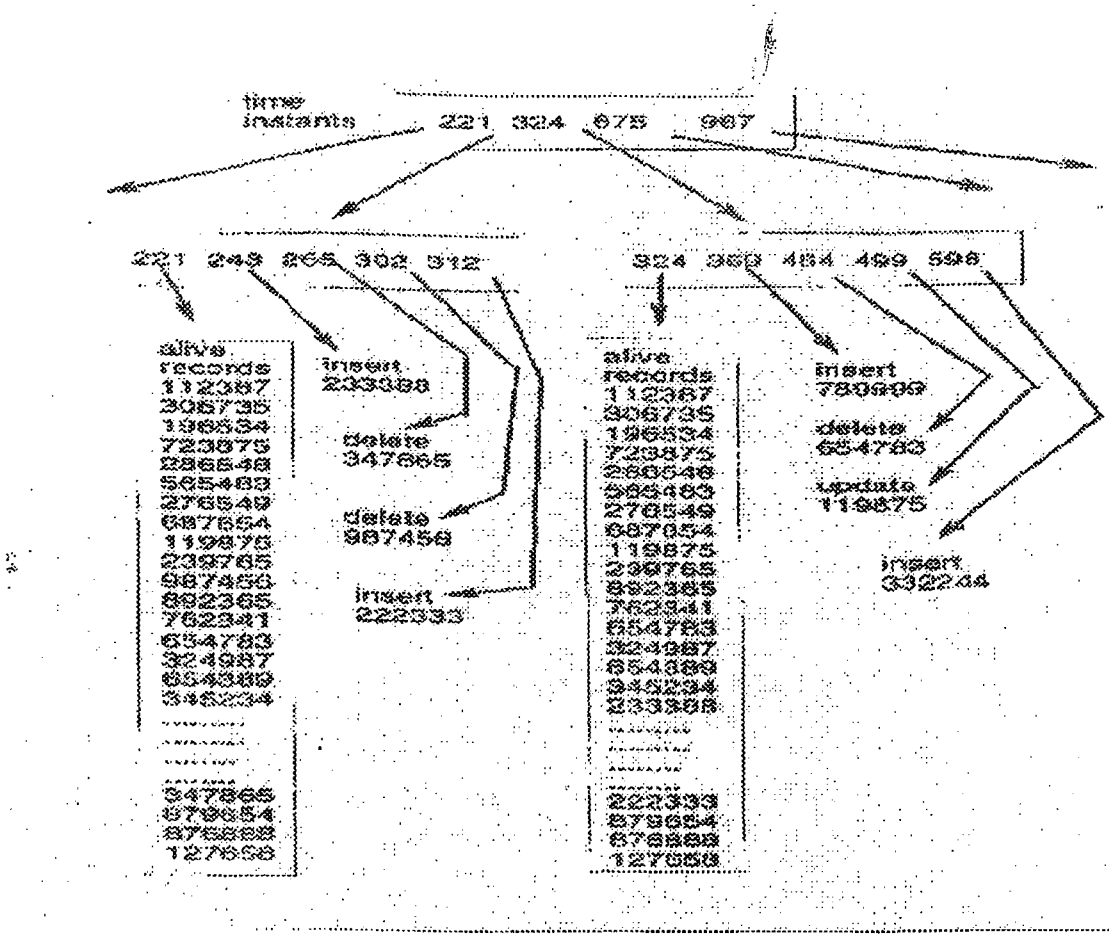


- *Segment tree (Δέντρο ευθυγράμμων τμημάτων)*

Αυτή η μέθοδος χρησιμοποιεί ένα B+ tree πάνω σε όλα τα κλειδιά που εισαχθήκανε ποτέ στη Βάση Δεδομένων. Ο κόμβος-φύλλο ενός τέτοιου δέντρου περιέχει εισόδους της μορφής key(p_1,p_2) όπου p_1 είναι ένας δείκτης στο Appendix tree που οργανώνει την εξέλιξη του συγκεκριμένου κλειδιού και p_2 ένας δείκτης που δείχνει στην τελευταία εκδοχή του κλειδιού. Το segment tree (δέντρο ευθυγράμμων τμημάτων) απαιτεί O(n/B) χώρο, O($\log_B$s) χρόνο ενημέρωσης και O(I$\log_B$s+a) I/Os για την pure key ερώτηση και O(K$\log_B$n) I/Os για την range timeslice ερώτηση, όπου s είναι ο αριθμός των διακριτών κλειδιών και K ο αριθμός των διακριτών κλειδιών της περιοχής κλειδιού για την range timeslice ερώτηση.

- *Time index*

Η μέθοδος αυτή υλοποιείται με τη χρήση ενός B+tree. κάθε κόμβος φύλλο της δομής χρονικής δεικτοδότησης είναι της μορφής (t, b), όπου t είναι ένα σημείο του χρόνου και b είναι ένας δείκτης σε ένα καλάθι (bucket). Ο δείκτης b της πρώτης entry ενός φύλλου δείχνει σε ένα καλάθι που κρατά όλες τις εγγραφές που είναι ζωντανές αυτή τη στιγμή. Οι υπόλοιπες entries του φύλλου δείχνουν σε ένα καλάθι που περιέχει τις αλλαγές που συμβαίνουν. Το time index φαίνεται στο παρακάτω σχήμα. Από τη δομή του time index προκύπτει και το προτέρημα του ότι δεν χρειάζεται να είναι εκ των προτέρων γνωστό το end_time. Οι απαιτήσεις της μεθόδου είναι O(n^2 /B) χώρος, O(n /B) χρόνος ενημέρωσης και O(I$\log_B$n+*a*/B) χρόνος ερώτησης.

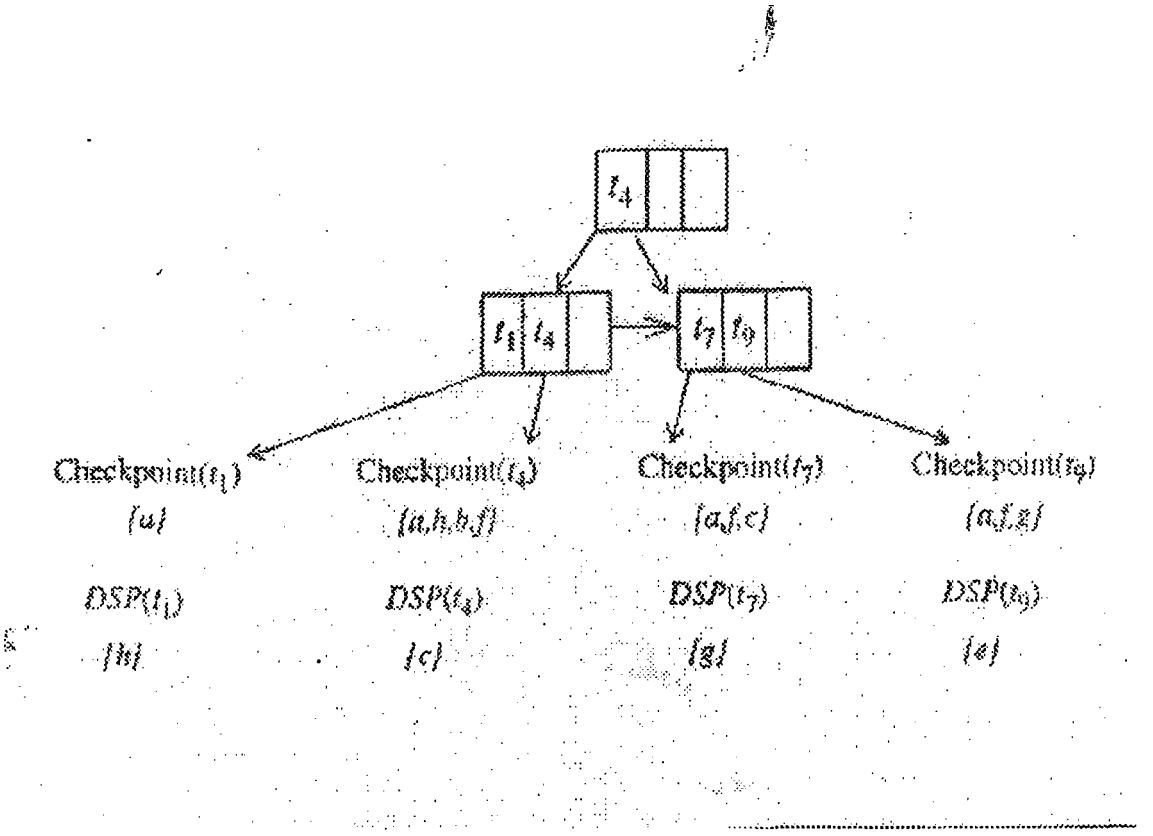


- *Archival time index*

Η τεχνική αυτή δεν ταξινομεί στιγμιότυπα συναλλαγών, αλλά αριθμούς εκδοχών (version numbers). Οι αριθμοί εκδοχών δημιουργούνται ως εξής. Το χρονικό στιγμιότυπο συναλλαγής της πρώτης αλλαγής έχει αριθμό εκδοχής μηδέν, της δεύτερης ένα κ.ο.κ. Η τεχνική αυτή χωρίζει παρελθόντα από τρέχοντα δεδομένα και δεικτοδοτεί τα δεύτερα με B+tree και τα πρώτα με μια πολυπλοκότερη δομή, την PVAS. Η μέθοδος αυτή κοστίζει O(n/B) χώρο, $O(\log_B n)$ χρόνο ενημέρωσης και $O(\log_2 n+a/B)$ για μια pure timeslice ερώτηση και $O(\log_2 n+s/B)$ για μια range timeslice ερώτηση.

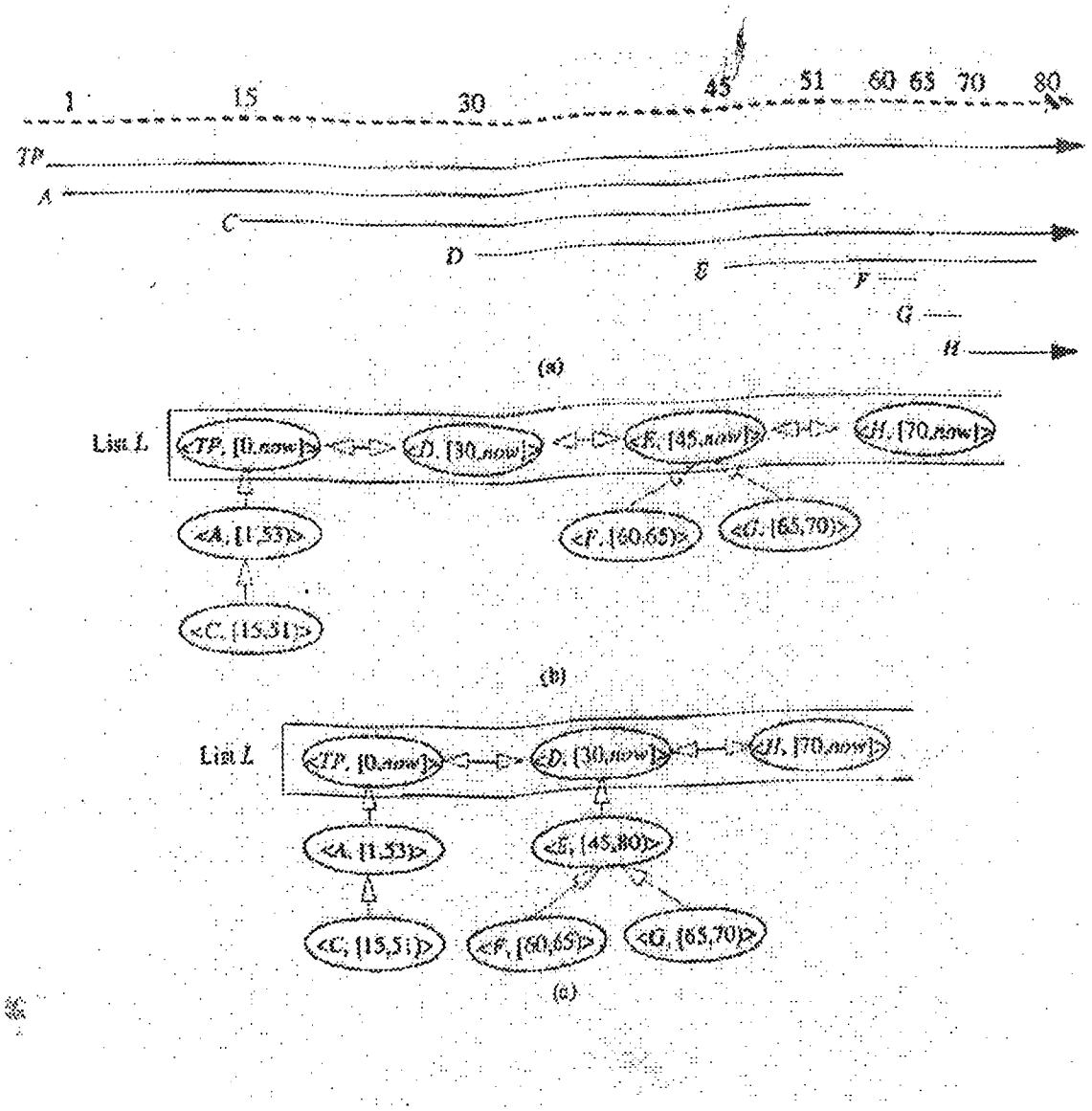
- *Checkpoint index*

Το checkpoint index υποθέτει ότι τα χρονικά διαστήματα ταξινομούνται με βάση το start_time τους. Όταν κάποια στιγμή t δημιουργείται ένα checkpoint, όλα τα αντικείμενα που είναι ζωντανά τη στιγμή αυτή αποθηκεύονται στο checkpoint. Υπάρχει και μια ξεχωριστή δομή που ονομάζεται data stream pointer (DSP) η οποία δείχνει στο πρώτο αντικείμενο που γεννιέται μετά τη χρονική στιγμή t. Το DSP παρέχει προσπέλαση σε μια ταξινομημένη ως προς το start_time των χρονικών διαστημάτων λίστα αντικειμένων που γεννιούνται μεταξύ των checkpoints. Τα χρονικά στιγμιότυπα του checkpoint δεικτοδοτούνται με μια B+tree δομή, όπως φαίνεται στο παρακάτω σχήμα. Η τεχνική αυτή έχει πολύ κακή απόδοση (O(n/B)) ως προς όλα τα κριτήρια απόδοσης.



- *Snapshot index*

Είναι η τεχνική που καταφέρνει την I/O βέλτιστη λύση για την transaction pure timeslice ερώτηση. Αποτελείται από τρεις δομές δεδομένων. Ένα πολυεπίπεδο ευρετήριο για προσπέλαση στα περασμένα δεδομένα βάση της χρονικής στιγμής t. Μια πολυσυνδεδεμένη δομή μεταξύ των σελίδων των φύλλων του πολυεπίπεδου ευρετηρίου. Αυτή χρησιμοποιείται για τη δημιουργία απάντησης σε μια ερώτηση που αφορά το χρόνο t. Τέλος μια συνάρτηση κατακερματισμού για προσπέλαση σε εγγραφές με βάση το κλειδί. Αυτή χρησιμοποιείται για τη διαδικασία ενημέρωσης. Σχηματικά το snapshot index φαίνεται στο επόμενο σχήμα.



Για να γίνει κατανοητό το σχήμα, πρέπει πρώτα να εξηγηθεί η έννοια της 'useful' σελίδας. Μια σελίδα δεδομένων λέγεται useful, όλες τις χρονικές στιγμές για τις οποίες η σελίδα αυτή ήταν acceptor σελίδα και εφόσον σταματήσει να είναι acceptor σελίδα, όλες τις στιγμές που περιέχει τουλάχιστον u*B ζωντανές εγγραφές. Acceptor σελίδα κάθε στιγμή είναι η σελίδα που αποθηκεύει τις εισόδους τη στιγμή αυτή. Κάθε στιγμή μπορεί να είναι μόνο μία. Στο (a) σχήμα φαίνονται τα χρήσιμα χρονικά διαστήματα κάθε σελίδας δεδομένων τη χρονική στιγμή 80. Δηλαδή, ένα ανοιχτό χρονικό διάστημα τη στιγμή t=80 αναπαριστά μια σελίδα δεδομένων που είναι χρήσιμη (useful) μέχρι αυτή τη στιγμή. Στο (b) σχήμα αναπαρίσταται η εξέλιξη του σχήματος τη στιγμή t = 79. Δηλαδή, now = 79. Κάθε σελίδα αντιπροσωπεύεται από ένα tuple της μορφής <page-id, page-usefuleness .period>. Η σελίδα TP υποδηλώνει την κορυφή της λίστας L, ενώ η τρέχουσα acceptor σελίδα είναι πάντα αυτή που βρίσκεται στο τέλος της L. Τέλος, στο (c) σχήμα φαίνεται το snapshot index τη χρονική στιγμή t = 80. Τη στιγμή αυτή η σελίδα E γίνεται μη-χρήσιμη, με αποτέλεσμα να διαγράφεται μαζί με το υποδέντρο της από την L και να τοποθετείται κάτω από την σελίδα που υπάρχει στη λίστα πριν από αυτή, δηλαδή την D. Η μέθοδος αυτή πετυχαίνει O(n/B) χώρο, O(1), δηλαδή σταθερό χρόνο ενημέρωσης και $O(\log_B m+a/B)$ χρόνο ερώτησης για την pure timeslice ερώτηση.

- *Windows method (Μέθοδος Παραθύρων)*

Στη μέθοδο αυτή ο χώρος του χρόνου χωρίζεται σε συνεχόμενα παράθυρα και για κάθε παράθυρο υπάρχει μια λίστα όλων των διαστημάτων που περιέχονται στο χρονικό διάστημα του παραθύρου. Τα παράθυρα δεικτοδοτούνται με μια B-tree δομή. Η μέθοδος αυτή πετυχαίνει όμοια πολύ καλή απόδοση με το snapshot index και ως προς τα τρία κριτήρια απόδοσης.

3. Μέθοδοι Προσπέλασης Χρόνου Συναλλαγής/Κλειδιού (Transaction/Key)

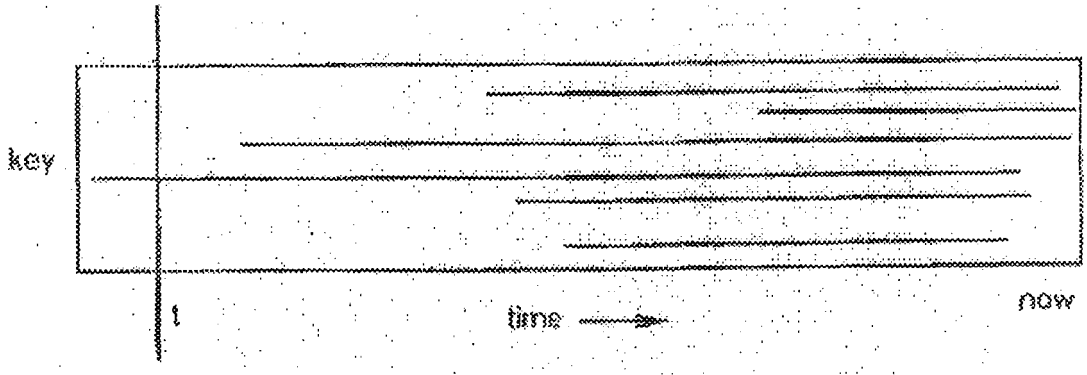
Τα δεδομένα οργανώνονται με βάση και το κλειδί και τον χρόνο συναλλαγής. Μια τέτοιου είδους οργάνωση είναι αποτελεσματικότερη για transaction range timeslice ερωτήσεις. Τα λογικά συσχετιζόμενα δεδομένα για μια συγκεκριμένη ερώτηση τοποθετούνται κοντά και έτσι ελαχιστοποιείται ο αριθμός των σελίδων που προσπελαύνονται. Οι μέθοδοι προσπέλασης σε αυτή την κατηγορία βασίζονται σε κάποια μορφή ισοζυγισμένου δέντρου, του οποίου οι σελίδες των φύλλων αντιστοιχούν δυναμικά σε περιοχές του δισδιάστατου transaction time key χώρου. Οι αλλαγές μπορεί να συμβαίνουν σε αύξουσα χρονική σειρά, αλλά δεν συμβαίνουν σε σειρά ως προς τα κλειδιά. Έτσι, η διαδικασία ενημέρωσης έχει λογαριθμικό κόστος ανά αλλαγή, για την τοποθέτηση των δεδομένων σύμφωνα με τις τιμές κλειδιού στον time-key χώρο.

Οι time-key μέθοδοι προσπέλασης μπορούν να υλοποιηθούν με δύο τρόπους. Με χρήση R-tree ή με χρήση B+tree. Το R tree έχει το πλεονέκτημα ότι μπορεί και υποστηρίζει επιπρόσθετες διαστάσεις στο ίδιο ευρετήριο. Δηλαδή, στην περίπτωση αυτή μπορεί να παραστήσει και τη διάσταση του χρόνου και τη διάσταση του κλειδιού. Στην πράξη τα R trees έχουν πολύ καλή απόδοση στη μέση περίπτωση. Τα μειονεκτήματα του R tree, όμως, είναι ότι δεν μπορεί να εγγυηθεί καλή απόδοση στη διαδικασία ενημέρωσης και στο χρόνο ερώτησης σε παθολογικά χειρότερες περιπτώσεις, καθώς και το ότι πρέπει να είναι εξ αρχής γνωστό η χρονική στιγμή τέλους του χρονικού διαστήματος. Ειδικά το τελευταίο χαρακτηριστικό κάνει τις R tree δομές δύσκολο να εφαρμοσθούν στον πραγματικό κόσμο. Παρακάτω αρχικά παρουσιάζονται κάποιες R- tree τεχνικές και στη συνέχεια κάποιες B-tree δομές.

- *Postgres*

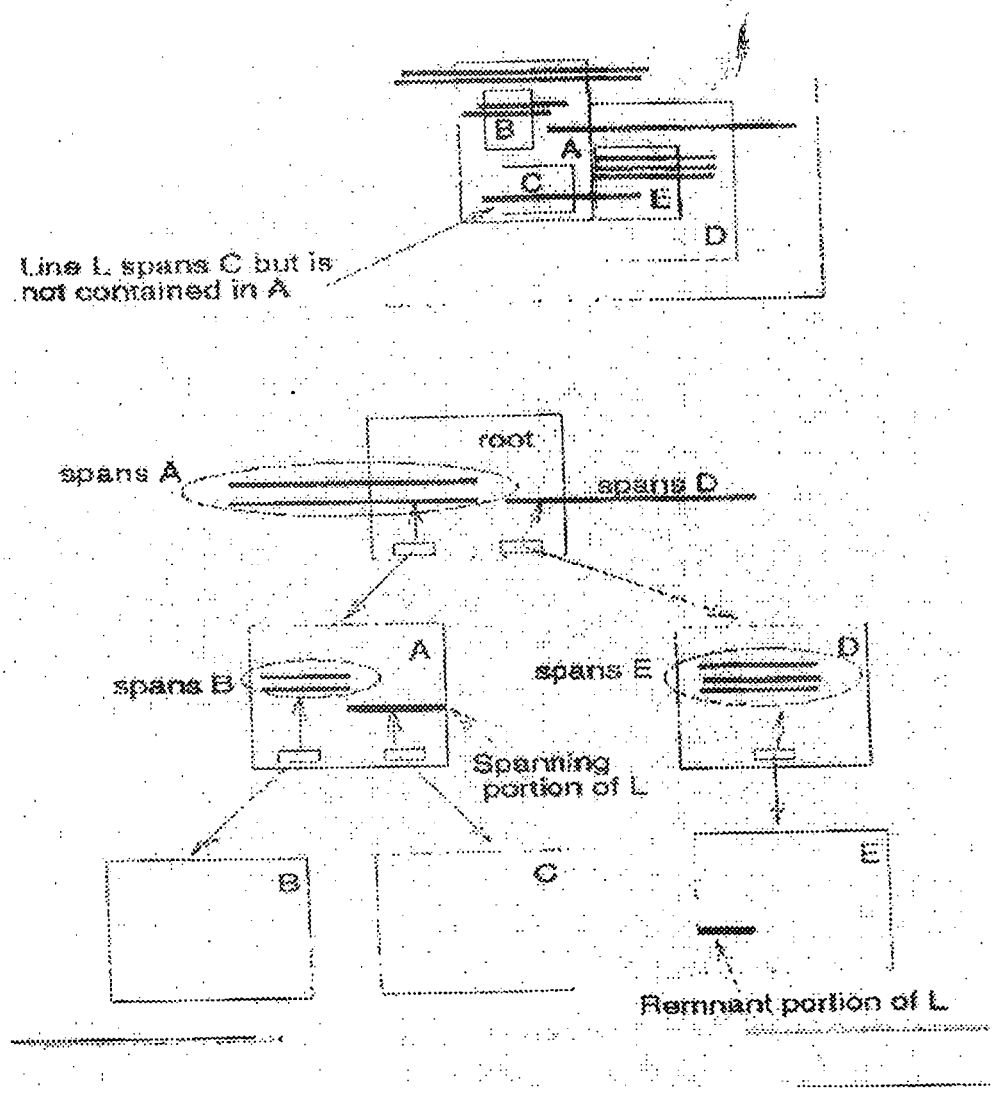
Η Postgress είναι μια τεχνική που χωρίζει τα παλιά από τα τωρινά δεδομένα. Συγκεκριμένα, τα περασμένα δεδομένα τα αποθηκεύει σε WORM συσκευές και τα τρέχοντα δεδομένα τα αποθηκεύει σε μαγνητικό δίσκο. Και στα δύο μέσα αποθήκευσης τα δεδομένα ταξινομούνται με τη βοήθεια δύο R-trees. Ένα για κάθε μέσο. Η μετακίνηση δεδομένων από το μαγνητικό δίσκο στις WORM συσκευές γίνεται μέσω μιας διαδικασίας που λέγεται Vacuum Cleaner και με την οποία μεταφέρονται ανά κάποιο χρονικό διάστημα όλες οι εγγραφές που δεν είναι ζωντανές από το μαγνητικό δίσκο, στον οπτικό. Αυτό έχει σαν αποτέλεσμα οι σελίδες στους οπτικούς δίσκους να έχουν κοινή χρονική στιγμή τέλους, όπως φαίνεται στο σχήμα. Η

τεχνική αυτή απαιτεί O(n/B) χώρο, $O(\log_B n)$ χρόνο ενημέρωσης και O(n/B) χρόνο ερώτησης. Η απόδοση της μεθόδου βελτιώνεται σημαντικά αν υπάρχει φυσική ομαδοποίηση δεδομένων.



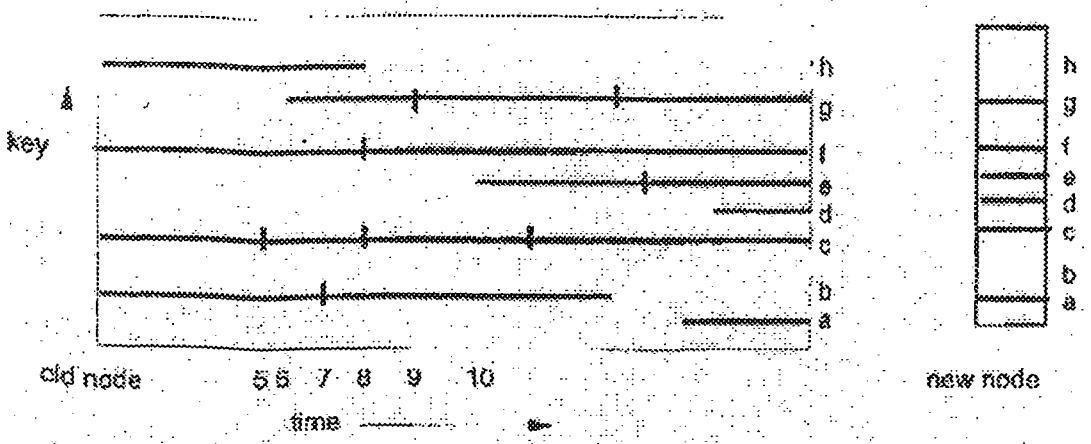
- *Segment R tree*

Ένα segment-tree (δέντρο ευθυγράμμων τμημάτων) αποθηκεύει τις χρονικές στιγμές τέλους των χρονικών διαστημάτων στους κόμβους-φύλλα. Κάθε εσωτερικός κόμβος σχετίζεται με μια περιοχή που περιέχει όλα τα σημεία τέλους που βρίσκονται στο υποδέντρό του. Ένα χρονικό διάστημα *I* αποθηκεύεται στον υψηλότερο εσωτερικό κόμβο *u* , όταν το *I* καλύπτει την περιοχή του *u* και δεν καλύπτει την περιοχή του πατέρα του *u*. Ένα SR-tree φαίνεται στο επόμενο σχήμα. Με την τεχνική αυτή τα μεγάλου μήκους διαστήματα αποθηκεύονται σε ψηλότερα επίπεδα του δέντρου. Αυτό έχει σαν συνέπεια την μείωση του φαινομένου της επικάλυψης. Η τεχνική αυτή έχει το ίδιο κόστος με την Postgress, εκτός από το χώρο ο οποίος μειώνεται σε $O((n/B)\log_B n)$.

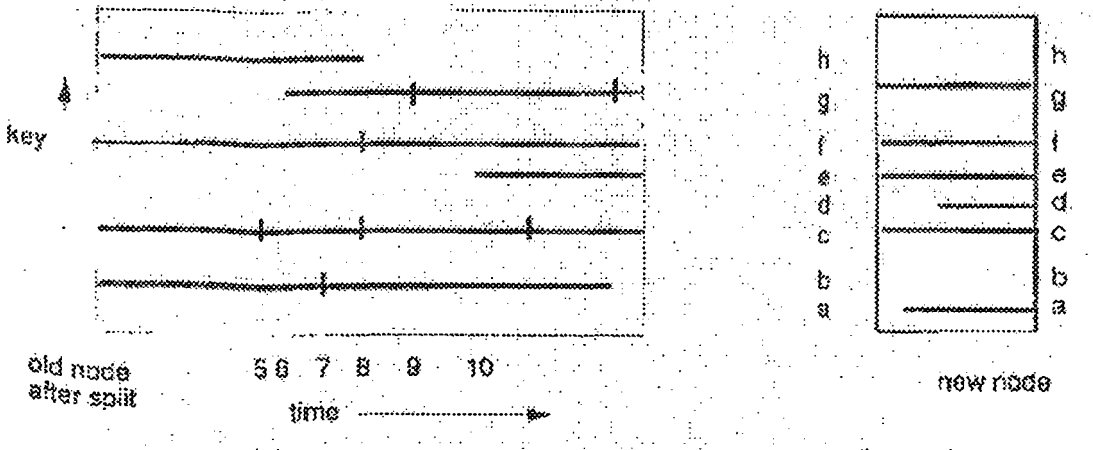


- *Write once B tree*

Το WOBT είναι μια παραλλαγή του B+tree που πληρεί και τους περιορισμούς των συσκευών WORM (Write Once Read Many). Δηλαδή, όταν μια σελίδα γράφεται δεν μπορούν να προστεθούν σε αυτή αργότερα άλλα δεδομένα ή να αλλάξουν κάποια από αυτά που περιέχει. Κάθε νέα είσοδος στο ευρετήριο καταλαμβάνει μια ολόκληρη σελίδα. Οι κόμβοι του δέντρου είναι συλλογές σελίδων. Οι εγγραφές εκδοχών περιέχουν μόνο τη χρονική στιγμή έναρξης συναλλαγής. Μια νέα εκδοχή του ίδιου κλειδιού θα αποθηκευτεί στον ίδιο κόμβο. Η χρονική στιγμή τέλου της προηγούμενης εγγραφής θα είναι η χρονική στιγμή έναρξης αυτής. Οι κόμβοι αντιπροσωπεύουν ένα τετράγωνο στον χώρο χρόνου συναλλαγής/κλειδιού. Όταν ένας κόμβος γεμίσει γίνεται διάσπαση είτε ως προς το χρόνο (και συγκεκριμένα την παρούσα χρονική στιγμή), είτε πρώτα ως προς το χρόνο και στη συνέχεια, αν χρειάζεται, ως προς το κλειδί. Ο τρόπος διάσπασης σε ένα WOBT φαίνεται στα επόμενα σχήματα. Οι απαιτήσεις αυτής της μεθόδου είναι O(n/B) σε χώρο, $O(\log_B m)$ σε χρόνο ενημέρωσης και $O(I\log_B n+a/B)$ σε χρόνο ερώτησης.



(a) The WOBT splits at current time, copying current records into a new node.

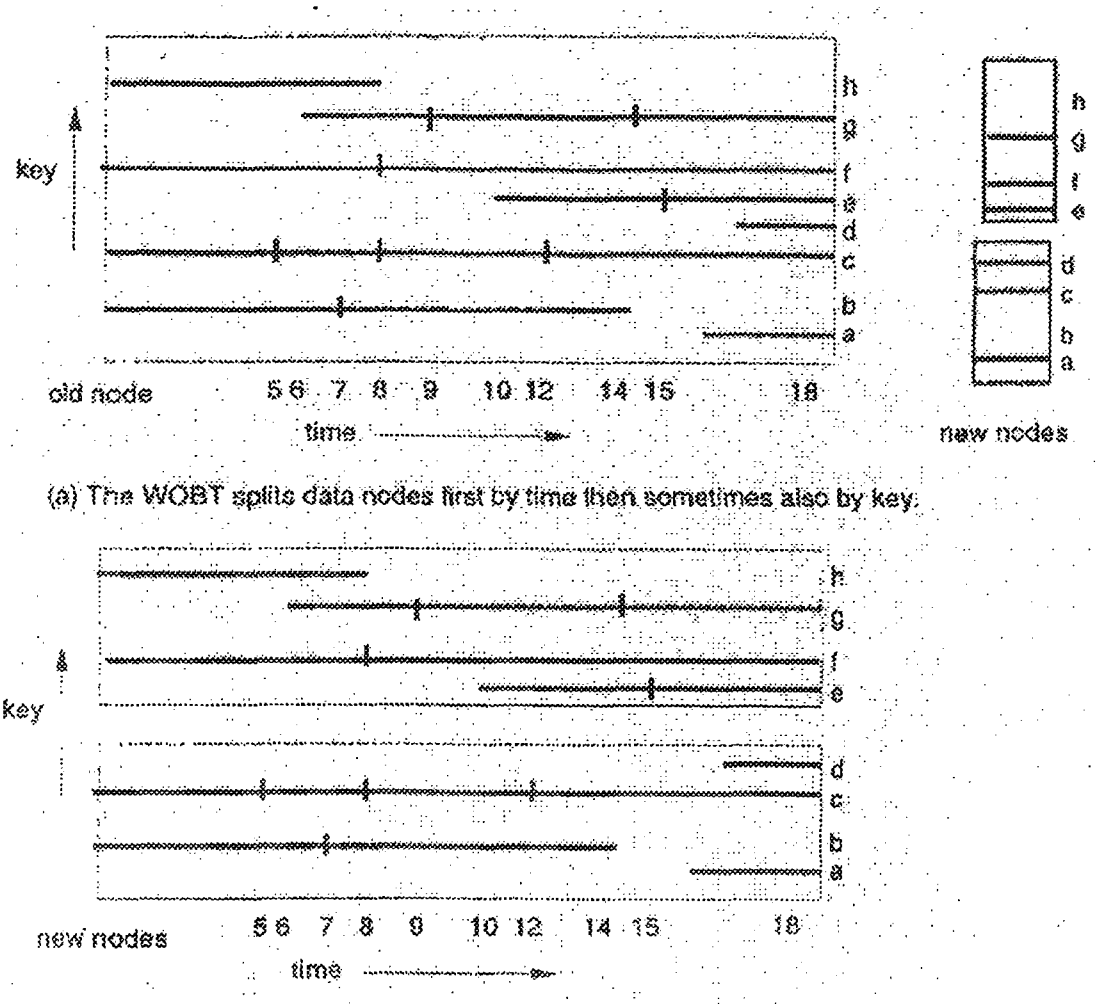


(b) The TSB tree can choose other times to split.

- *Time split B tree*

Είναι μια τροποποίηση του WOBT που επιτρέπει απλές διασπάσεις κλειδιού και κρατά τα τρέχοντα δεδομένα σε ένα μέσο που επιτρέπει αλλαγή δεδομένων (ένα μαγνητικό δίσκο) όταν γίνει η διάσπαση ως προς το χρόνο. Η διαφορά του με το WOBT είναι ότι διασπά αποτελεσματικά μόνο ως προς το χρόνο ή μόνο ως προς το κλειδί και ότι διάσπαση ως προς το χρόνο δεν είναι απαραίτητο να γίνει την τρέχουσα χρονική στιγμή. Τα χαρακτηριστικά του TSB φαίνονται στα επόμενα

σχήματα . Αυτή η τεχνική απαιτεί O(n/B) χώρο, O($\log_B$n) χρόνο ενημέρωσης και O($\log_B$n+*a*) χρόνο ερώτησης για την pure key time ερώτηση.



(a) The WOBT splits data nodes first by time then sometimes also by key.

(b) The TSB-tree can split by key alone.

- *Persistent B tree fat node*

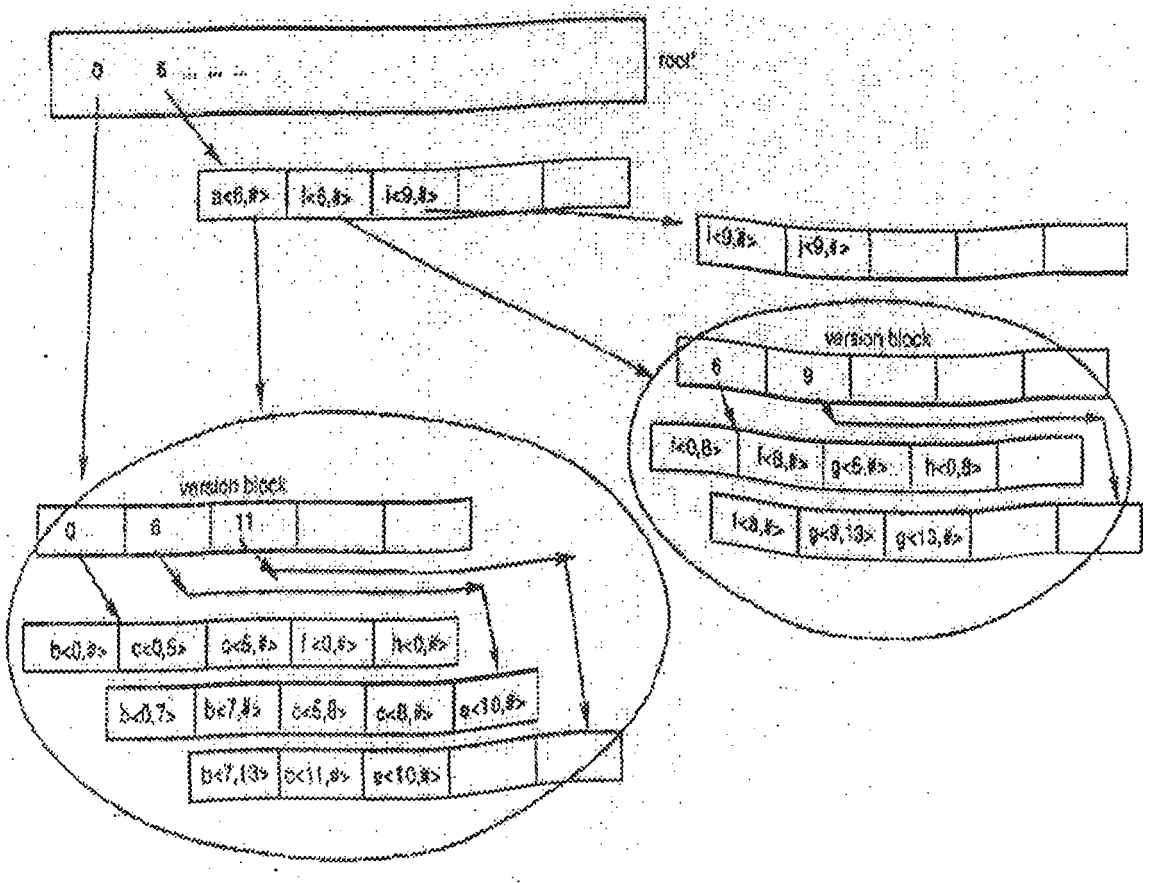
Η μέθοδος αυτή απαιτεί O(n) χώρο, O($\log_B$m) χρόνο ενημέρωσης και O($\log_B$n($\log_B$m,+α/B)) χρόνο ερώτησης. Βασίζεται στη δομή του B+ tree και περιγράφεται από το παρακάτω σχήμα. Τη χρονική στιγμή 0 η Βάση Δεδομένων περιέχει εγγραφές με κλειδιά h, f, c και b. Τη χρονική στιγμή 5, μια νέα εκδοχή της εγγραφής με κλειδί c. Τη χρονική στιγμή 6, εισάγεται η εγγραφή g. Κάθε χρονική στιγμή που συμβαίνει μια αλλαγή αντιστοιχεί σε μια νέα έκδοση της βάσης δεδομένων .Υποθέτοντας ότι η χωρητικότητα κάθε σελίδας είναι 5 εγγραφές, γίνονται τα παρακάτω. Αρχικά (πρώτος κύκλος του σχήματος), κάθε φορά που δημιουργείται μια νέα εκδοχή, δημιουργείται μια νέα σελίδα δεδομένων μόνο τις εγγραφές που είναι ζωντανές μετά την αλλαγή αυτή. Μια άλλη σελίδα που λέγεται version block δεικτοδοτεί τις σελίδες αυτές. Έτσι, αν αναζητούνται στιγμές πριν τη χρονική στιγμή

5, ακολουθείται η πρώτη σελίδα, ενώ αν αναζητούνται χρονικές στιγμές από τη στιγμή 5 και μετά ακολουθείται η δεύτερη σελίδα. Η εγγραφή 'c2' είναι η ενημερωμένη εκδοχή της εγγραφής 'c'. Ένα σύνολο σελίδων δεδομένων μαζί με το version block τους, λέγεται fat node.

Στη συνέχεια (κύκλος 2 στο σχήμα), τη χρονική στιγμή 6 εισάγεται μια νέα εγγραφή με κλειδί g. Έτσι, μια νέα σελίδα δεδομένων τοποθετείται και ενημερώνεται και το version block. Μετά (στον τρίτο κύκλο), τη στιγμή 7 , γίνεται μια ενημέρωση της εγγραφής b. Τέλος (στον τέταρτο κύκλο του σχήματος), τη χρονική στιγμή 8 γίνεται ενημέρωση στις εγγραφές με κλειδί c και f.

- *Persistent B tree fat field*

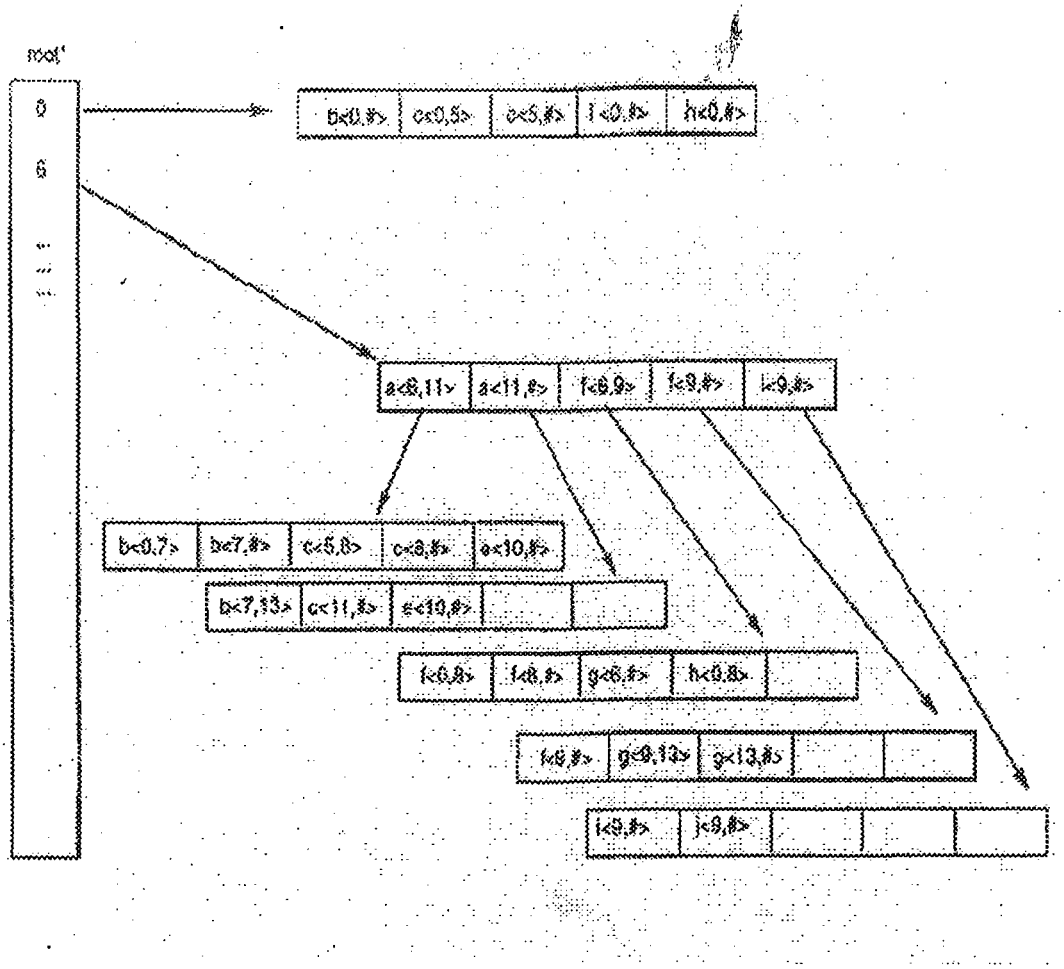
Η μέθοδος αυτή απαιτεί O(n/B) χώρο, $O(\log_B m)$ χρόνο ενημέρωσης και $O(\log_B n(\log_B m + \alpha/B))$ χρόνο ερώτησης. Βασίζεται στη δομή του B+tree και περιγράφεται στο παρακάτω σχήμα.



Η fat field μέθοδος αποθηκεύει εγγραφές σε μια σελίδα όσο αυτή χωράει. Τη χρονική στιγμή 5 μια νέα εκδοχή της εγγραφής c τοποθετείται στη σελίδα και ταυτόχρονα ορίζεται και η χρονική στιγμή τέλους της αμέσως προηγούμενης εκδοχής. Τη χρονική στιγμή 6 συμβαίνει υπερχείλιση. Στον καινούριο κόμβο υπάρχουν τέσσερα διακριτά κλειδιά και συνεπώς γίνεται διάσπαση με βάση το κλειδί. Τη χρονική στιγμή 9 , εισάγονται οι i και j και ενημερώνεται η g, οπότε προκύπτει νέα διάσπαση και στη νέα ρίζα εισάγεται ακόμη μια entry. Τη χρονική στιγμή 11, μια νέα εκδοχή της εγγραφής c δημιουργεί υπερχείλιση με μόνο μια αντιγραφή και όχι διάσπαση κλειδιού. Οι fat κόμβοι δημιουργούνται όταν η πρώτη λειτουργία αντιγραφής γίνει. Η αναζήτηση για ένα συγκεκριμένο κλειδί ακολουθεί τους δείκτες που ξεκινούν πριν ή τη στιγμή της αναζήτησης και δεν τελειώνουν πριν την στιγμή αναζήτησης. Η αναζήτηση μέσα σε ένα version block ή root* γίνεται ακολουθώντας τη μεγαλύτερη χρονική στιγμή που είναι ίση ή μικρότερη της στιγμής αναζήτησης.

- *Multiversion B tree*

ΤΟ MVBT προσφέρει την I/O βέλτιστη λύση για μια transaction range timeslice ερώτηση. Δηλαδή, κοστίζει O(n/B) σε χώρο, $O(\log_B m)$ σε χρόνο ενημέρωσης και $O(I\log_B n+a/B)$ σε χρόνο ερώτησης. Είναι μια τεχνική που μοιάζει πολύ με την WOBT, αλλά υποστηρίζει αποδοτικά και διαγραφές και χρησιμοποιεί ένα είδος αντιγραφής κόμβου. Χρησιμοποιεί μια root* δομή, όπως φαίνεται στο σχήμα. Όταν μια ρίζα κάνει μια διάσπαση ως προς το χρόνο, ο 'αδερφός' της ρίζας γίνεται και αυτός ρίζα. Τότε μια νέα είσοδος τοποθετείται στον μεταβλητού μήκους πίνακα η οποία δείχνει την νέα ρίζα. Αν η ρίζα κάνει διάσπαση και ως προς το χρόνο και ως προς το κλειδί, τότε το νέο δέντρο έχει ακόμα ένα επίπεδο.



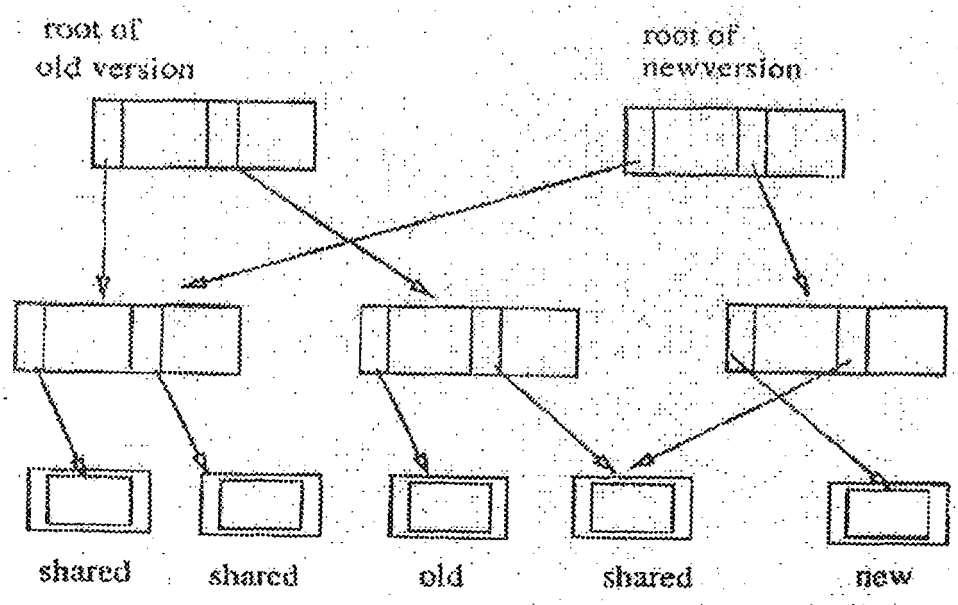
Τη χρονική στιγμή 5, μια νέα εκδοχή της εγγραφής c τοποθετείται στη σελίδα και ταυτόχρονα ορίζεται και η χρονική στιγμή τέλους της αμέσως προηγούμενης εκδοχής. Τη χρονική στιγμή 6 συμβαίνει υπερχείλιση. Στον καινούριο κόμβο υπάρχουν τέσσερα διακριτά κλειδιά και συνεπώς γίνεται διάσπαση με βάση το κλειδί. Τη χρονική στιγμή 9, εισάγονται οι i και j και ενημερώνεται η g, οπότε προκύπτει νέα διάσπαση και στη νέα ρίζα εισάγεται ακόμη μια εκδοχή. Τη χρονική στιγμή 11, μια νέα εκδοχή της εγγραφής c δημιουργεί υπερχείλιση με μόνο μια αντιγραφή και όχι διάσπαση κλειδιού. Αυτό έχει σαν αποτέλεσμα την εισαγωγή μιας νέας εισόδου (entry) στο ευρετήριο.

- *Multiversion δομή προσπέλασης*

Το MVAS είναι όμοιο με το MVBT, με τη διαφορά ότι η σταθερά στο ασυμπτωτικό όριο πολυπλοκότητας χώρου είναι μικρότερη και χρησιμοποιούνται καλύτεροι αλγόριθμοι διαχείρισης των περιπτώσεων διάσπασης ή συγχώνευσης. Παρόλα αυτά, ασυμπτωτικά οι απαιτήσεις αυτής της μεθόδου είναι ίδιες με του MVBT. Δηλαδή, $O(n/B)$ χώρος, $O(\log_B m)$ χρόνος ενημέρωσης, $O(I\log_B n+a/B))$ χρόνο ερώτησης.

- *Overlapping B+tree .*

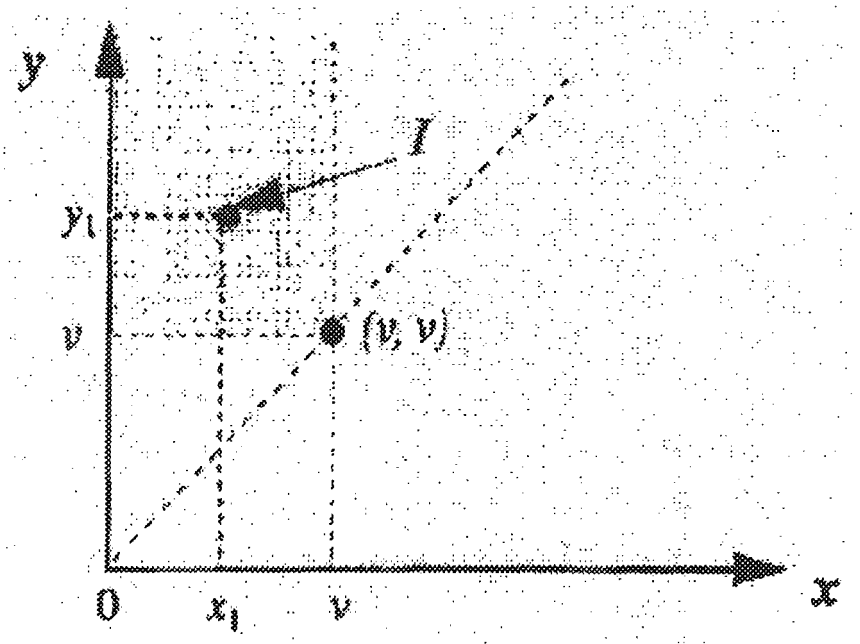
Είναι μια νέα δομή προσπέλασης που βασίζεται σε ένα B+tree. Όταν μια νέα εκδοχή εισάγεται σαν ενημέρωση σε μια σελίδα φύλλου, δημιουργούνται αντίγραφα όλου του μονοπατιού από το φύλλο στη ρίζα, αλλάζοντας κάποιες αναφορές όπου είναι απαραίτητο. Κάθε νέα εκδοχή έχει μια ξεχωριστή ρίζα και τα υποδένδρα κάθε ρίζας ίσως να μοιράζονται. Δηλαδή, ένας κόμβος ίσως να ανήκει σε περισσότερο από ένα υποδέντρα, όπως φαίνεται στο σχήμα. Η τεχνική έχει πολύ καλή απόδοση. Συγκεκριμένα, κοστίζει $O(n\log_B n)$ χώρο, $O(\log_B m)$ χρόνο ενημέρωσης και $O(I\log_B n+a/B))$ χρόνο ερώτησης.



4.3.2 Είδη Μεθόδων Προσπέλασης για Βάσεις Δεδομένων Έγκυρου Χρόνου

Όπως έχει ήδη αναφερθεί, μια Βάση Δεδομένων έγκυρου χρόνου, πρέπει να διατηρεί μια δυναμική συλλογή από αντικείμενα διαστημάτων. Η βέλτιστη συντήρηση μιας Βάσης Δεδομένων έγκυρου χρόνου είναι πολύ δύσκολη διαδικασία, καθώς σε ένα περιβάλλον Βάσεων Δεδομένων έγκυρου χρόνου, η ομαδοποίηση των δεδομένων σε σελίδες μπορεί να αλλάξει δραματικά από τις ενημερώσεις που πραγματοποιούνται. Οι διαγραφές είναι φυσικές και οι εισαγωγές μπορούν να συμβούν οπουδήποτε στο πεδίο έγκυρου χρόνου. Μια timeslice ερώτηση έγκυρου χρόνου είναι στη πραγματικότητα μια ειδική περίπτωση μιας δύο διαστάσεων ερώτησης περιοχής. Ένα χρονικό διάστημα περιέχει ένα σημείο ερώτησης u, αν και μόνο αν η χρονική στιγμή έναρξης του είναι μικρότερη ή ίση του υ και η χρονική στιγμή τέλους του είναι μεγαλύτερη ή ίση του u. Έστω ότι ένα χρονικό διάστημα $I=(x_i,y_i)$ αντιστοιχεί στο σημείο (x_i,y_i) του δισδιάστατου χώρου. Τότε ένα χρονικό διάστημα περιέχει ένα query point u, αν και μόνο αν το αντίστοιχο σημείο στο δισδιάστατο χώρο βρίσκεται

μέσα στο τετράγωνο που σχηματίζεται από τις ευθείες x=0, x=u, y=u και y= ∞ ,όπως φαίνεται στο σχήμα . Εφόσον η χρονική στιγμή τέλους ενός χρονικού διαστήματος είναι πάντα μεγαλύτερη ή ίση της χρονικής στιγμής έναρξης, όλα τα χρονικά διαστήματα αναπαρίστανται με σημεία πάνω από τη διαγώνιο x=y .Αυτός ο συμβολισμός χρησιμοποιείται στο δέντρο προτεραιότητας, το οποίο είναι η δομή δεδομένων που παρέχει τη βέλτιστη λύση για τη κύρια μνήμη. Έχουν γίνει πολλές προσπάθειες για να μεταφερθεί αυτή η δομή στη δευτερεύουσα μνήμη. Παρακάτω περιγράφονται μερικές.



- *Metablock tree*

Αυτή η τεχνική δεικτοδότησης δημιουργήθηκε με σκοπό τη διαχείριση του δυναμικού χρονικού διαστήματος. Το metablock tree είναι μια μέθοδος προσπέλασης που χωρίζει τον πάνω διαγώνιο πίνακα του δισδιάστατου χώρου σε metablocks, κάθε ένα από τα οποία έχει B^2 σημεία δεδομένων. Το metablock tree είναι μια ημιδυναμική δομή, αφού υποστηρίζει μόνο εισαγωγές στοιχείων και όχι διαγραφές. Η τεχνική αυτή κοστίζει $O(I/B)$ χώρο, $O(\log_B I+(\log_B I)^2/B)$ χρόνο ενημέρωσης και $O(\log_B I+a/B)$ χρόνος ερώτησης.

- *External segment tree*

Το external segment tree είναι μια σελιδοποιημένη έκδοση του segment tree (δέντρου ευθυγράμμων τμημάτων). Η μέθοδος αυτή έχει κόστος $O((I/B)\log_2 V)$ χώρο, $O(\log_2 V)$ χρόνο ενημέρωσης και $O(\log_2 V+a)$ χρόνο ερώτησης, όπου V είναι οι πιθανές τιμές των άκρων των ευθυγράμμων τμημάτων.

- *MAP21 μέθοδοι*

Στη μέθοδο αυτή ένα χρονικό διάστημα (x,y) έγκυρου χρόνου αναπαρίσταται με ένα σημείο $z = x10^s + y$, όπου s είναι ο μέγιστος αριθμός ψηφίων που χρειάζονται για την αναπαράσταση οποιουδήποτε χρονικού σημείου στο πεδίο ορισμού έγκυρου χρόνου. Τα σημεία αυτά γίνονται με ένα B-δέντρο. Αυτή η μέθοδος έχει καλό χρόνο ενημέρωσης, αλλά ο χρόνος ερώτησης είναι αρκετά μεγάλος.

4.3.3 Είδη Διχρονικών Μεθόδων Προσπέλασης

Η αναζήτηση μεταξύ των διαστημάτων μιας αποθηκευμένης συλλογής $C(t_i)$, περιλαμβάνει μια μέθοδο προσπέλασης για κάθε αποθηκευμένη $C(t_i)$. Αυτές οι συλλογές και οι μέθοδοι που τις συνοδεύουν στη συνέχεια μπορούν να δεικτοδοτηθούν από ένα κανονικό B-tree στο t_i, δηλαδή στον άξονα χρόνου συναλλαγής.

- *M - IVTT*

Είναι μια τεχνική δεικτοδότησης, στην οποία οι αλλαγές μεταξύ των $C(t_i)$ λέγονται Patches και κάθε αποθηκευμένη συλλογή δεικτοδοτείται με ένα MAP21. Η M-IVTT μπορεί να θεωρηθεί επέκταση του χρονικού δεικτοδοτητή σε ένα διχρονικό περιβάλλον. Ο χώρος, ο χρόνος ενημέρωσης και ο χρόνος ερώτησης αυξάνονται ανάλογα με το πόσο συχνά δεικτοδοτούνται οι συλλογές.

- *Διχρονικό Δέντρο Διαστήματος*

Η μέθοδος αυτή για να απαντήσει μια */point/point και */range/point χρειάζεται $O((n+V)/B)$ χώρο, $O(\log_B(m+V))$ I/Os ανά αλλαγή και $O(\log_B V+\log_B n+a)$ I/Os χρόνο ερώτησης.

- *Διχρονικό R tree*

Η μέθοδος αυτή για να απαντήσει μια range/point/point και range/range/point με χρήση R*tree χρειάζεται λίγο περισσότερο χώρο αλλά πολύ καλύτερο χρόνο ενημέρωσης και χρόνο ερώτησης σε σχέση με το R δέντρο.

5ο Κεφάλαιο

"Συμπεράσματα..."

5 · Συμπεράσματα...

Στη μελέτη αυτή παρουσιάστηκαν διάφορες τεχνικές δεικτοδότησης που έχουν προταθεί για αποτελεσματική προσπέλαση σε χρονικά δεδομένα. Το μεγαλύτερο τμήμα αυτού της εργασίας ασχολείται με τις Βάσεις Δεδομένων χρόνου συναλλαγής (transactin time), μιας και οι περισσότερες μελέτες έχουν γίνει για αυτές. Περιγράφονται, όμως, και κάποιες προσεγγίσεις για Βάσεις Δεδομένων έγκυρου χρόνου (valid time) και για διχρονικές (bitemporal) Βάσεις Δεδομένων. Οι συγκρίσεις γίνονται ως προς κάποια βασικά κριτήρια απόδοσης, τα οποία είναι: απαιτήσεις σε χώρο, χαρακτηριστικά ενημέρωσης και απόδοση ερώτησης. Κατά τον υπολογισμό της απόδοσης λαμβάνεται υπόψη και το είδος της ερώτησης. Παράλληλα πρέπει να επισημανθεί ότι εφόσον είναι πρακτικά αδύνατο να προσομοιωθούν όλες οι μέθοδοι με την ίδια είσοδο, η σύγκριση βασίζεται στην απόδοση χειρότερης περίπτωσης των εξεταζόμενων μεθόδων.

Η απόδοση ερωτήματος μετριέται κατά τριών βασικών συναλλαγής-χρονικών ερωτημάτων: το βασικό-κλειδί, το βασικό- timeslice, και τα ερωτήματα πεδίο-timeslice, ή, χρησιμοποιώντας τη σημείωση τριών-εισόδων, οι "σημείο/-/*", το "*/-/σημείο," και ερωτήματα "πεδίο- /σημείο", αντίστοιχα. Πρόσθετα, εξετάσθηκαν διάφορα θέματα όπως σελιδοποιήσεις ευρετηρίων και η δυνατότητα μιας μεθόδου για την αποτελεσματική μετακίνηση στοιχείων σε κάποια άλλα μέσα (όπως μια συσκευή WORM). Επιπλέον, ζητήματα όπως η συγκέντρωση δεδομένων, οι σελιδοποιήσεις δεικτών, η μετακίνηση των στοιχείων στους οπτικούς δίσκους, κ.λ.π, μπορούν επίσης να είναι λίγο πολύ σημαντικά, σύμφωνα με την εφαρμογή. Από τη μελέτη δεν προέκυψε κάποιο τελικό συμπέρασμα όσον αφορά την σαφή υπεροχή κάποιας μεθόδου. Εντούτοις, δεν μπορεί να θεωρηθεί ότι η εφαρμογή και η πειραματική αξιολόγηση των μεθόδων προσπέλασης δεν συνεισφέρουν ουσιαστικά δεδομένου ότι αποκαλύπτονται συχνά οι ανεπάρκειες και τα προβλήματα που δεν είναι προφανείς από ένα σχέδιο ή ένα θεωρητικό πρότυπο. Προκειμένου να κατασταθούν τέτοιες συγκριτικές αξιολογήσεις ευκολότερες στο να εκτελέσουν και στο να ελέγξουν, είναι ουσιαστικό να παρασχεθεί η ανεξάρτητη πλατφόρμας πρόσβαση στις εφαρμογές ενός ευρέως φάσματος μεθόδων προσπέλασης.

Παρουσιάστηκαν επίσης πολλές και διαφορετικές μέθοδοι προσπέλασης σε βάσεις σύνθετων αντικειμένων (χωρικές Βάσεις Δεδομένων), των οποίων τα πλεονεκτήματα και τα αδύνατα σημεία δεν είναι πάντα ευδιάκριτα και σαφή, ακόμα και μέσω πειραματικής μελέτης και αξιολόγησής τους. Το γεγονός αυτό οφείλεται στα ποικίλα και διαφορετικά κριτήρια και παραμέτρους με βάση τα οποία μπορεί να γίνει η σύγκριση στην απόδοση αυτών των μεθόδων. Η απόδοση τόσο στο χρόνο όσο και στη χρησιμοποίηση του χώρου από κάθε τέτοια μέθοδο εξαρτάται σημαντικά από τα δεδομένα εισόδου, από τις συγκεκριμένες ερωτήσεις που τίθενται προς τη βάση καθώς και από των πλήθος των λειτουργιών εισαγωγής και διαγραφής που πρέπει να γίνουν. Εντούτοις, παρά τις εγγενείς δυσκολίες, ο σχεδιασμός, η υλοποίηση και η πειραματική/θεωρητική αξιολόγηση διαφορετικών μεθόδων προσπέλασης δεδομένων έχει θεμελιώδη σημασία για την αποκάλυψη των ισχυρότερων και για τον προσδιορισμό των περιπτώσεων στις οποίες είναι αποδοτικότερο να χρησιμοποιείται καθεμία από αυτές. Πάντως πρέπει να αναφερθεί ότι μια από τις καλύτερες και πιο αντιπροσωπευτικές πλατφόρμες δοκιμής και πρακτικού ελέγχου των -αρχικά - θεωρητικά θεμελιωμένων μεθόδων προσπέλασης και διαχείρισης σύνθετων δεδομένων αποτελεί ο παγκόσμιος ιστός (WWW), και πολλά είναι τα ερευνητικά και τεχνολογικά ιδρύματα καθώς και οι εταιρίες υψηλής τεχνολογίας που έχουν -εδώ και χρόνια -στρέψει την προσοχή τους προς αυτή την κατεύθυνση.
Η έρευνα στα χωρικά συστήματα βάσεων δεδομένων έχει οδηγήσει σε ένα πλήθος χωρικών μεθόδων προσπέλασης. Ακόμη και για τους εμπειρογνώμονες γίνεται όλο και περισσότερο δύσκολο να αναγνωριστούν οι αξίες και οι αδυναμίες τους, δεδομένου ότι κάθε νέα μέθοδος φαίνεται να απαιτεί την ανωτερότητα τουλάχιστον σε μια μέθοδο προσπέλασης . Αυτή η έρευνα δεν προσπάθησε να επιλύσει αυτό το πρόβλημα αλλά μάλλον να δώσει μια επισκόπηση των πλεονεκτημάτων και των μειονεκτημάτων ποικίλων δομών. Αυτό δεν θα αποτελέσει καμία έκπληξη ότι αυτή τη στιγμή καμία μέθοδος προσπέλασης δεν έχει αποδειχθεί ανώτερη από όλους τους ανταγωνιστές της υπό οποιαδήποτε έννοια. Ακόμα κι αν μια συγκριτική μέτρηση επιδόσεων κηρύσσει μια δομή ως σαφή νικητή, μια άλλη συγκριτική μέτρηση επιδόσεων μπορεί να αποδείξει την ίδια δομή κατώτερη.

Αλλά γιατί είναι τέτοιες συγκρίσεις τόσο δύσκολες;
Επειδή υπάρχουν τόσα πολλά διαφορετικά κριτήρια για να καθορίσουν την ευνοϊκότερη συνθήκη και τόσες πολλές παράμετροι που καθορίζουν την απόδοση. Και η απόδοση χώρου και χρόνου μιας μεθόδου προσπέλασης εξαρτώνται έντονα από τα επεξεργαζόμενα δεδομένα και από τα ερωτήματα που ζητούνται. Μια μέθοδος προσπέλασης που αποδίδει ικανοποιητικά για τα ισοπροσανατολισμένα ορθογώνια μπορεί να αποτύχει για τις αυθαίρετα προσανατολισμένες γραμμές. Τα έντονα συσχετιζόμενα στοιχεία μπορούν να καταστήσουν μια ειδάλλως γρήγορη μέθοδο προσπέλασης άσχετη για οποιαδήποτε πρακτική εφαρμογή. Ένας δείκτης που έχει βελτιστοποιηθεί για τα ερωτήματα σημείου μπορεί να είναι ιδιαίτερα ανεπαρκής για τα αυθαίρετα ερωτήματα περιοχών. Οι μεγάλοι αριθμοί εισαγωγών και διαγραφών μπορούν να επιδεινώσουν μια δομή που είναι αποδοτική σε ένα στατικότερο περιβάλλον.

Η πρωτοβουλία να ιδρυθεί μια τυποποιημένη δοκιμή για την αξιολόγηση και τη σύγκριση των μεθόδων προσπέλασης υπό διαφορετικών όρων ήταν ένα σημαντικό βήμα στη σωστή κατεύθυνση. Το World Wide Web παρέχει μια κατάλληλη υποδομή για να έχει πρόσβαση και να διανείμει τέτοιες συγκριτικές μετρήσεις επιδόσεων .

Εντούτοις, παραμένει λίγο δύσκολο να συγκρίνει ή να ταξινομήσει τις διαφορετικές μεθόδους προσπέλασης. Οι πειραματικές συγκριτικές μετρήσεις επιδόσεων πρέπει να μελετηθούν με προσοχή και μπορούν μόνο να είναι ένας πρώτος δείκτης για τη δυνατότητα χρησιμοποίησης.

Όταν η μεταφορά τεχνολογίας, δηλαδή στη χρήση των μεθόδων προσπέλασης στα εμπορικά προϊόντα, οι περισσότεροι προμηθευτές προσφεύγουν στις δομές που είναι εύκολες στην κατανόηση και εφαρμογή. Τα Quadtrees, τα R-δέντρα και z-διάταξη είναι χαρακτηριστικά παραδείγματα. Η απόδοση φαίνεται να είναι δευτερεύουσας σπουδαιότητας στην επιλογή, η οποία δεν αποτελεί καμία έκπληξη δεδομένων των σχετικά μικρών διαφορών μεταξύ των μεθόδων σε ουσιαστικά όλες τις δημοσιευμένες αναλύσεις.

Επίσης η συγκεκριμένη εργασία ασχολήθηκε με την έννοια της διαχρονικότητας στις δομές δεδομένων. Με βάση τη διαχρονικότητα οι δομές δεδομένων μπορούν να διαχωριστούν σε δύο κατηγορίες στις εφήμερες και τις διαχρονικές. Στις εφήμερες δομές δεδομένων οποιαδήποτε αλλαγή στη δομή καταστρέφει την παλιότερη έκδοση επιτρέποντας πρόσβαση μόνο στην καινούργια έκδοση. Αντίθετα οι διαχρονικές δομές επιτρέπουν πρόσβαση σε οποιαδήποτε έκδοση παλιά η καινούργια σε οποιαδήποτε χρονική στιγμή. Ακόμη παρουσιάστηκαν οι διάφορες λειτουργίες που μπορούν να εφαρμοστούν στη διασυνδεδεμένη δομή. Παρουσιάζονται αποδοτικές τεχνικές με τις οποίες μια εφήμερη δομή μετατρέπεται σε μερικώς διαχρονική. Συγκεκριμένα μελετούνται δύο μέθοδοι: Η μέθοδος "fat node" και η μέθοδος "node copying", πολύπλοκες τεχνικές μετατροπής μιας εφήμερης δομής σε πλήρως διαχρονική.

ΑΝΑΦΟΡΕΣ – ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] James R. Driscoll, Neil Sarnak, Daniel D. Sleator, Robert E. Tarjan, Making Data Structures Persistent, *Journal of Computer and System Sciences*,

[2] Volker Gaede , Oliver Gunther , Multidimensional Access Methods, *ACM Computing Surveys Vol. 30, No. 2, June 1998, 170-230*

[3] Betty Salzberg, Vassilis Tsotras, Comparison for Access Methods for Time-Evolving Data, *ACM Computing Surveys, Vol 31, No. 2, June 1999, 158-221.*

Επίσης χρησιμοποιήθηκε υλικό από μεταπτυχιακές εργασίες των φοιτητών:

Γλύκου Φωτεινή
Σταύρου Αθανασόπουλου
Βλαχογιάννης Γιώργος ,
Στο μεταπτυχιακό μάθημα «Βασικές Δομές Δεδομένων» του Τμήματος Μηχανικών Η/Υ και Πληροφορικής.

ΕΥΡΕΤΗΡΙΟ