ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βελτιστοποίηση επερωτήσεων με χρήση υλοποιημένων όψεων

Σπύρος Γ. Ζουπάνος

Επιβλέπων: Αλέξιος Δελής, Αναπληρωτής Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2006

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βελτιστοποίηση επερωτήσεων με χρήση υλοποιημένων όψεων

Σπύρος Γ. Ζουπάνος
Α.Μ.: M754

ΕΠΙΒΛΕΠΩΝ:

Αλέξιος Δελής, Αναπληρωτής Καθηγητής ΕΚΠΑ

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Αλέξιος Δελής, Αναπληρωτής Καθηγητής ΕΚΠΑ

Παναγιώτης Ροντογιάννης, Επίκουρος Καθηγητής ΕΚΠΑ

Σεπτέμβριος 2006

ΠΕΡΙΛΗΨΗ

Στόχος της συγκεκριμένης διπλωματικής εργασίας είναι να εξετάσει την βελτίωση που μπορεί να επιτευχθεί κατά την εκτέλεση επερωτήσεων χρησιμοποιώντας ένα κατάλληλο σύνολο από υλοποιημένες όψεις. Για να μπορέσουν να πραγματοποιηθούν οι μετρήσεις που επιθυμούμε κατασκευάστηκε ένα βελτιστοποιητής επερωτήσεων ο οποίος μπορεί να χρησιμοποιήσει παράλληλα και υλοποιημένες όψεις.

Συνοπτικά η λειτουργία του συγκεκριμένου βελτιστοποιητή είναι η εξής:

- Μετατροπή της επερώτησης του χρήστη σε δένδρο σχεσιακής άλγεβρας.
- Βελτιστοποίηση της συγκεκριμένης επερώτησης και των παραλλαγών της χρησιμοποιώντας ευριστικούς κανόνες.
- Αναζήτηση για κάθε βελτιστοποιημένη παραλλαγή της επερώτησης υλοποιημένων όψεων που μπορεί να είναι χρήσιμες και η χρήση τους θα επιφέρει το μέγιστο όφελος.
- Δημιουργία για κάθε βελτιστοποιημένη παραλλαγή της επερώτησης νέων αλγεβρικών πλάνων με χρήση όψεων.
- Εκτίμηση του κόστους όλων των αλγεβρικών πλάνων.
- Εύρεση της βελτίωσης που πραγματοποιείται με χρήση υλοποιημένων όψεων αλλά και γραφική παρουσίαση των αποτελεσμάτων της βελτιστοποίησης.

Συμπεράσματα της παραπάνω διαδικασίας είναι ότι αν χρησιμοποιήσουμε 50.000 υλοποιημένες όψεις για επερωτήσεις πάνω σε μια βάση δεδομένων τεσσάρων πινάκων μπορούμε να έχουμε μείωση του κόστους πάνω από 80%. Αν κάνουμε μια πιο προσεκτική επιλογή των υλοποιημένων όψεων που χρησιμοποιούμε μπορούμε να επιτύχουμε ακόμα και με 500 υλοποιημένες όψεις μείωση πάνω από 50%.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Βελτιστοποίηση επερωτήσεων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: βελτιστοποίηση, επερώτηση, υλοποιημένες όψεις, ευριστικός βελτιστοποιητής, εκτιμητής κόστους

ABSTRACT

Aim of the specific diplome thesis is to examine the amount of improvement that can be achieved with the use of a specific set of materialized views in query optimization. In order to perform our experiments and measurements we constructed a query optimizer that can also use a set of materialized views in order to optimize the given queries.

The basic steps that our optimizer follows are:

- Transformation of the given query in a relational algebra tree.
- Heuristically optimize the given query and all its (left-deep tree) variations.
- For each heuristically optimized variation search the set of materialized views and find the most cost effective views that can replace parts of the specific variation.
- Use the found views of the previous step in order to construct alternative plans with the use of these materialized views.
- Estimate the cost of all plans.
- Calculate the amount of improvement that achieved with the use of views and graphically present the outcome of the optimization process.

Conclusions of the procedure described above is that if we use 50.000 materialized views for queries on a four table database we can achieve above 80% reduction on the original cost of these queries. With a more sophisticated selection of views we can achieve even with 500 views a reduction above 50%.

SUBJECT AREA: Query Optimisation

KEYWORDS: query, optimisation, materialized views, heuristic optimiser, cost estimator

Ευχαριστώ θερμά τον καθηγητή μου Αλέξη Δελή για την σταθερή καθοδήγησή του κατά την διάρκεια και της αναζήτησης του θέματος αλλά και της υλοποίησης της συγκεκριμένης διπλωματικής.

1. Εισαγωγή

Ένα διαχρονικό ερώτημα αλλά και στόχος στις βάσεις δεδομένων είναι η βελτιστοποίηση των επερωτήσεων που υποβάλλει κάποιος σε ένα ΣΔΒΔ (Σύστημα Διαχείρισης Βάσεων Δεδομένων). Θεωρούμε ότι μια επερώτηση βελτιστοποιείται όταν μειώνεται το κόστος (υπολογισμοί, διάβασμα από δίσκους κλπ) για το σύστημα που την επεξεργάζεται. Αυτό έχει ως αποτέλεσμα το συγκεκριμένο σύστημα να είναι σε θέση να δώσει απάντηση σε μικρότερο χρονικό διάστημα για την ίδια επερώτηση.

Ένας από τους τρόπους που μπορεί κάποιος να ακολουθήσει για να βελτιώσει μια επερώτηση είναι να χρησιμοποιήσει προαποθηκευμένες (υλοποιημένες) όψεις που είναι δυνατόν να προϋπάρχουν ήδη στο ΣΔΒΔ. Με τον τρόπο αυτό θα μπορέσει να χρησιμοποιήσει τα ήδη προϋπολογισμένα αποτελέσματα των όψεων και να τα συνδυάσει με άλλα που υπολογίζει εκείνη την στιγμή με στόχο να δημιουργήσει το πλήρες σύνολο των αποτελεσμάτων που χρειάζεται.

Μια υλοποιημένη όψη μπορεί να αποτελεί την πλήρη απάντηση μια επερώτησης ή πιθανόν να χρειάζεται κάποιου είδους επεξεργασία, όπως αναφέρεται παραπάνω, έτσι ώστε τελικά να είναι χρήσιμη. Δηλαδή η όψη μπορεί να αποτελέσει κάποιο ενδιάμεσο αποτέλεσμα στον συνολικό υπολογισμό της απάντησης μιας επερώτησης. Αξιόλογες προσπάθειες βελτίωσης των επερωτήσεων με χρήση κοινών ενδιάμεσων αποτελεσμάτων ανάμεσα σε επερωτήσεις έχουν γίνει ήδη και όχι μόνο με χρήση όψεων σαν ενδιάμεσα αποτελέσματα [1]. Στην περίπτωσή μας όμως τα ενδιάμεσα αποτελέσματα που θα χρησιμοποιηθούν δεν προέρχονται από κάποια άλλη επερώτηση που βελτιστοποιείται παράλληλα εκείνη την στιγμή αλλά είναι ήδη αποθηκευμένα στην βάση μας και πρέπει να τα ανακτήσουμε.

Στόχος της συγκεκριμένης εργασίας είναι να παρουσιάσει το μέγεθος της βελτίωσης που παρατηρείται στην εκτέλεση των επερωτήσεων αν χρησιμοποιηθούν υλοποιημένες όψεις. Αναλυτικότερα, έχει αναπτυχθεί ένα πρόγραμμα στο οποίο μπορεί να δοθεί μια επερώτηση και λαμβάνοντας υπ' όψη του τις υλοποιημένες όψεις που υπάρχουν, μπορεί να υπολογίσει το κόστος απάντησης της επερώτησης χωρίς την χρήση όψεων αλλά και με την χρήση αυτών. Ακόμα μπορεί να παρουσιάσει με γραφικό τρόπο τις πράξεις που θα πραγματοποιηθούν για την εκτέλεση της επερώτησης με τον κάθε τρόπο καθώς και το

κόστος της κάθε πράξης.

Συνεπώς δίνεται στον χρήστη η δυνατότητα να παρακολουθήσει τα στάδια που ακολουθεί μια επερώτηση από την στιγμή που δίνεται στο σύστημα μέχρι την εκτέλεσή της αλλά και να κατανοήσει και να διαπιστώσει προσωπικά το μέγεθος της βελτίωσης που παρουσιάζεται όταν χρησιμοποιούνται υλοποιημένες όψεις.

2. Ανάλυση της εργασίας

2.1 Διάρθρωση της εργασίας

Η εργασία μπορεί να χωριστεί σε τρία τμήματα:

1. Το βασικό τμήμα, το οποίο είναι και η κύρια εφαρμογή, είναι ένας βελτιστοποιητής επερωτήσεων βάσης δεδομένων που λαμβάνει υπ' όψη του και τις υλοποιημένες όψεις που υπάρχουν στο σύστημα. Ως είσοδο δέχεται την επερώτηση που θέλουμε να του υποβάλλουμε αλλά παράλληλα πρέπει να του έχουμε χορηγήσει και στατιστικά στοιχεία που αφορούν τα δεδομένα της βάσης δεδομένων μας (ποιοι πίνακες υπάρχουν, ποιες στήλες έχουν οι πίνακες, μέγιστες και ελάχιστες τιμές ανά στήλη σε κάθε πίνακα κλπ.) καθώς και τις περιγραφές των υλοποιημένων όψεων του συστήματος. Ως έξοδο μας δίνει όλες τις δυνατές βελτιστοποιημένες παραλλαγές της συγκεκριμένης επερώτησης καθώς και ένα ενδεικτικό κόστος που θα χρειαστεί ένα ΣΔΒΔ για να δώσει αποτελέσματα για την κάθε παραλλαγή. Επιπλέον είναι σε θέση να μας δώσει και την επί της εκατό (%) βελτίωση που θα έχουμε αν χρησιμοποιήσουμε το βελτιστοποιημένο πλάνο το οποίο χρησιμοποιεί όψεις σε αντίθεση με εκείνο που δεν χρησιμοποιεί.

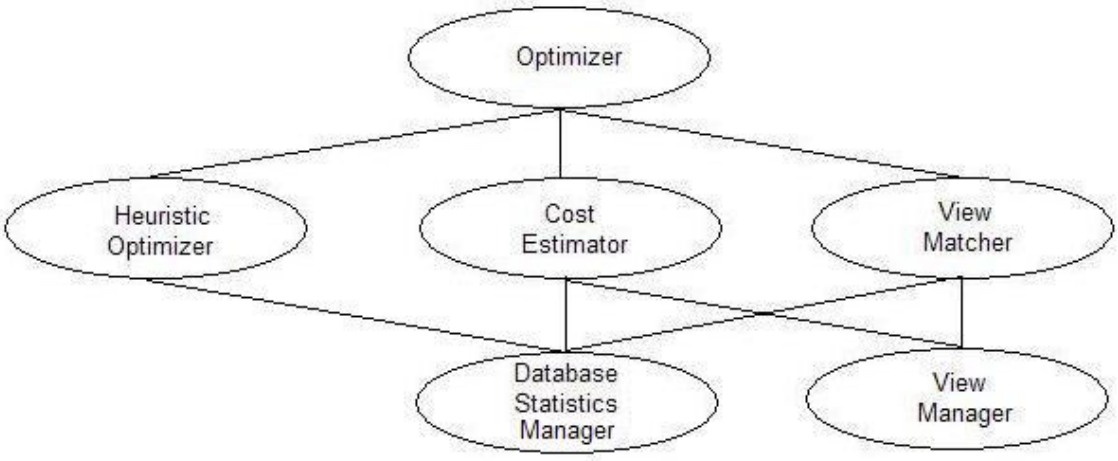
2. Το δεύτερο τμήμα της εργασίας είναι ένα γραφικό περιβάλλον το οποίο σχεδιάζει το πλάνο εκτέλεσης της επερώτησης και σε κάθε στάδιο (ζεύξη, προβολή κλπ.), μας αναφέρει το κόστος μέχρι εκείνο το στάδιο, τον αριθμό των γραμμών του αποτελέσματος καθώς και το μέγεθος σε bytes της γραμμής του αποτελέσματος. Ουσιαστικά η εφαρμογή αυτή παρουσιάζει με γραφικό τρόπο τα αποτελέσματα του βελτιστοποιητή.

3. Τέλος το τρίτο και τελευταίο τμήμα είναι ένα πρόγραμμα το οποίο έχει ως στόχο να δημιουργήσει ένα μεγάλο αριθμό όψεων βασισμένο στα στατιστικά στοιχεία και τις πληροφορίες που θα του έχουμε δώσει για τα δεδομένα που είναι αποθηκευμένα στην βάση μας. Το συγκεκριμένο πρόγραμμα το χρησιμοποιούμε για να πραγματοποιήσουμε πειράματα ώστε να μελετήσουμε το πως συμπεριφέρεται η κύρια εφαρμογή μας με διαφορετικό αριθμό όψεων.

2.2 Βελτιστοποιητής

2.2.1 Γενική περιγραφή λειτουργίας του βελτιστοποιητή

Ο βελτιστοποιητής που σχεδιάσαμε συνεργάζεται με διαφορετικά αντικείμενα έτσι ώστε να είναι σε θέση να μας δώσει μια απάντηση στο ερώτημα ποιο πλάνο εκτέλεσης είναι το βέλτιστο. Τα σχετικά αντικείμενα καθώς και η σχέση τους με τον βελτιστοποιητή φαίνονται στο σχήμα που ακολουθεί.



Εικόνα 1: Συσχέτιση αντικειμένων του προγράμματος

Ο βελτιστοποιητής (optimizer) αρχικά δέχεται μια επερώτηση από τον χρήστη. Στην συνέχεια αναλαμβάνει να την μετατρέψει σε σχεσιακή άλγεβρα και δημιουργεί ισοδύναμες παραλλαγές της επερώτησης οι οποίες στο μόνο σημείο που διαφέρουν είναι διάταξη των πινάκων που παίρνουν μέρος στην επερώτηση.

Μετά στέλνει όλες αυτές τις ισοδύναμες εκφράσεις στον ευριστικό βελτιστοποιητή (heuristic optimizer), ο οποίος αναλαμβάνει να τις βελτιστοποιήσει πλήρως αλλά και μερικώς (θα δούμε στην ενότητα του ευριστικού βελτιστοποιητή τι εννοούμε ως πλήρη ή μερική βελτιστοποίηση). Για να πραγματοποιηθεί κάτι τέτοιο πρέπει ο τελευταίος να

επικοινωνήσει με την σειρά του με το αντικείμενο που διαχειρίζεται τα στατιστικά που αφορούν τους πίνακες της βάσης δεδομένων μας (database statistics manager). Όταν ο βελτιστοποιητής λάβει τις νέες εκφράσεις από τον ευριστικό βελτιστοποιητή τότε τις στέλνει στον εκτιμητή κόστους (cost estimator). Στόχος του τελευταίου είναι να υπολογίσει το κόστος εκτέλεσης της κάθε σχέσης – πλάνου που του δίνεται. Για να το πετύχει αυτό πρέπει να επικοινωνήσει με τον database statistics manager. Το συγκεκριμένο αντικείμενο θα του δώσει πληροφορίες που αφορούν τον αριθμό των πινάκων της βάσης, ποιες στήλες έχει κάθε πίνακας, πληροφορίες για τα στοιχεία της κάθε στήλης και άλλα.

Όταν ο εκτιμητής κόστους τελειώσει με την εκτίμηση του κόστους του κάθε πλάνου τότε επιστρέφει τα αποτελέσματα στον βελτιστοποιητή. Ο τελευταίος στέλνει τα βελτιστοποιήμενα πλάνα που έλαβε από τον ευριστικό βελτιστοποιητή και στο αντικείμενο που αντιστοιχεί όψεις (view matcher). Στόχος του συγκεκριμένου αντικείμενου είναι να ψάξει και να βρει όψεις που καταρχήν μπορούν να χρησιμοποιηθούν για να απαντήσουν μέρος ή και ολόκληρη την βελτιστοποιημένη έκφραση που του έχει δοθεί. Για να βρει τις κατάλληλες όψεις πρέπει να επικοινωνήσει και με το αντικείμενο το οποίο διαχειρίζεται τις αποθηκευμένες όψεις και αυτό είναι ο view manager. Ρόλος του view manager είναι να αποθηκεύει τις διαθέσιμες όψεις και παράλληλα να προσφέρει έναν γρήγορο τρόπο προσπέλασής τους (κατακερματισμός). Αφού λάβει ο view matcher ένα σύνολο πιθανών όψεων που μπορεί να είναι χρήσιμες από τον view manager, εξετάζει αν είναι πραγματικά χρήσιμες και αν ναι, τι πράξεις χρειάζονται να πραγματοποιηθούν πάνω τους για να μπορέσουν να εισαχθούν στο κατάλληλο σημείο της έκφρασης που μας ενδιαφέρει.

Όταν πια δημιουργηθούν οι νέες εκφράσεις με χρήση όψεων από τον view matcher επιστρέφονται στον βελτιστοποιητή ο οποίος τις στέλνει στον εκτιμητή κόστους για να λάβει εκτίμηση για το κόστος της κάθε έκφρασης. Ο τελευταίος επικοινωνεί και με τον view manager διότι θα χρειαστεί κάποια στοιχεία που αφορούν την όψη (π.χ. αριθμό γραμμών και μέγεθος γραμμών) που καλό είναι να μην τα υπολογίζει κάθε φορά από την αρχή.

Κλείνοντας, ανάλογα την επιλογή που έχουμε δώσει κατά την εκτέλεσή στον βελτιστοποιητή θα μας επιστρέψει είτε όλα τα αποτελέσματα είτε τα βέλτιστα σε μορφή που να μπορούμε να την επεξεργαστούμε – δούμε με το γραφικό περιβάλλον που τα σχεδιάζει ή εναλλακτικά απλά θα μας τυπώνει στην οθόνη την επί τις εκατό βελτίωση που είχαμε με χρήση των όψεων.

Στην συνέχεια ακολουθεί περιγραφή βασικών λειτουργιών του βελτιστοποιητή καθώς και των αντικειμένων με τα οποία συνεργάζεται.

2.2.1 Μετατροπή επερώτησης σε σχεσιακή άλγεβρα

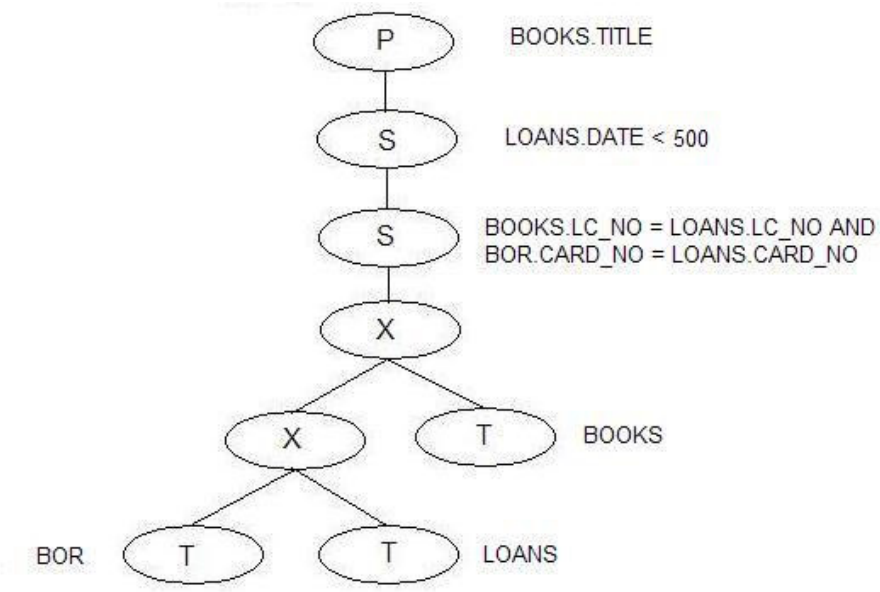
Το συγκεκριμένο τμήμα δέχεται μια επερώτηση από τον χρήστη σε μια μορφή που μοιάζει με SQL (θα περιγραφεί στην συνέχεια) και αναλαμβάνει να την μετατρέψει σε σχεσιακή άλγεβρα. Ένα παράδειγμα επερώτησης που μπορούμε να υποβάλλουμε εμφανίζεται στον πίνακα που ακολουθεί.

```
<query>
      <projectcollumns>
            BOOKS.TITLE
      </projectcollumns>
      <constrains>
            LOANS.DATE l 500
      </constrains>
      <joins>
            BOOKS.LC_NO e LOANS.LC_NO
            BOR.CARD_NO e LOANS.CARD_NO
      </joins>
      <tables>
            BOR
            LOANS
            BOOKS
      </tables>
</query>
```

Πίνακας 1: Παράδειγμα επερώτησης χρήστη

Αρχικά αναφέρουμε τις στήλες πάνω στις οποίες θέλουμε να πραγματοποιηθεί μια προβολή. Στην συγκεκριμένη περίπτωση θέλουμε να μας εμφανιστεί στο αποτέλεσμα η στήλη TITLE του πίνακα BOOKS. Το συγκεκριμένο κομμάτι αντιστοιχεί στο select τμήμα μιας SQL επερώτησης. Στην συνέχεια ακολουθούν οι περιορισμοί. Οι περιορισμοί θέτουν τα όρια μέσα στα οποία θέλουμε να λαμβάνει τιμές μια στήλη κάποιου πίνακα. Μετά τους περιορισμούς αυτούς αναφέρουμε τις στήλες πάνω στις οποίες θέλουμε να πραγματοποιήσουμε κάποια ζεύξη. Οι περιορισμοί αλλά και οι ζεύξεις ουσιαστικά αποτελούν το where κομμάτι μιας SQL επερώτησής. Αυτό που πρέπει να αναφερθεί στο

σημείο αυτό είναι ότι θεωρούμε ότι υπάρχει σύζευξη ανάμεσα σε όλους τους περιορισμούς αλλά και στις ζεύξεις. Δηλαδή θέλουμε στα αποτελέσματα που (θεωρητικά) θα μας εμφανιστούν να ισχύουν όλοι οι περιορισμοί αλλά και οι ζεύξεις ταυτόχρονα για κάθε γραμμή. Αποτέλεσμα του συγκεκριμένου τμήματος του προγράμματος είναι να σχεδιάσει ένα δένδρο σχεσιακής άλγεβρας που αντιστοιχεί στην παραπάνω επερώτηση. Για την συγκεκριμένη επερώτηση του πίνακα 1 θα έχουμε το δένδρο που ακολουθεί.



Εικόνα 2: Σχεσιακό αλγεβρικό δένδρο επερώτησης χρήστη

Αρχικά θα έχουμε έναν κόμβο PROJECT (αργότερα θα αναφέρεται και ως P κόμβος) που θα συμβολίζει τις προβολές που θέλουμε να πραγματοποιούνται (BOOKS.TITLE) στην συγκεκριμένη περίπτωση. Σαν δεύτερος κόμβος είναι ένας κόμβος τύπου SELECT (αργότερα θα αναφέρεται και ως S κόμβος) που θα αναγράφει τους περιορισμούς που θέλουμε (LOANS.DATE < 500). Τρίτος κόμβος θα είναι πάλι ένας κόμβος τύπου SELECT και θα αναγράφει τις ζεύξεις που θέλουμε να πραγματοποιηθούν (BOOKS.LC_NO = LOANS.LC_NO και BOR.CARD_NO = LOANS.CARD_NO). Σε αυτή τη φάση αντιμετωπίζουμε και τις ζεύξεις περίπου σαν τους απλούς περιορισμούς τιμής που αναφέραμε πιο πριν. Δηλαδή δεν τον αντιστοιχίζουμε με κάποιον αλγόριθμο ζεύξης και

θεωρούμε ότι απλώς ζητάμε μια ισότητα ανάμεσα στις τιμές δύο διαφορετικών στηλών. Ο επόμενος κόμβος που συναντάμε είναι ένας κόμβος τύπου X. Ο συγκεκριμένος κόμβος στην φάση που βρισκόμαστε αντιστοιχεί σε καρτεσιανό γινόμενο. Οι κόμβοι που ακολουθούν στην συνέχεια είναι κατανοητοί. Αντιστοιχούν είτε σε καρτεσιανό γινόμενο είτε σε κάποιο πίνακα.

Σίγουρα το να εκτελούμε μια πράξη καρτεσιανού γινομένου και στην συνέχεια να πραγματοποιούμε την ζεύξη με μορφή περιορισμού είναι ένας μη αποδοτικός τρόπος εκτέλεσης μια επερώτησης. Το πρώτο μας μέλημα κατά την ευριστική βελτιστοποίηση είναι να μετατρέψουμε καρτεσιανά γινόμενα σε ζεύξεις. Οι συμβολισμοί παραμένουν οι ίδιοι αλλά από το σημείο αυτό και μετά κάνουμε την ακόλουθη σύμβαση: Όταν συναντάμε έναν κόμβο καρτεσιανού γινομένου που έχει ως πατέρα του έναν κόμβο τύπου S που περιέχει ζεύξη ανάμεσα σε στήλες διαφορετικών πινάκων τότε θεωρούμε ότι οι δύο κόμβοι αυτοί αποτελούν μια ενιαία ζεύξη (μπορούμε να τους θεωρήσουμε και σαν ένα κόμβο). Αν ο κόμβος καρτεσιανού γινομένου έχει ως πατέρα του έναν οποιοδήποτε άλλο κόμβο (ακόμα και τύπου S αλλά με απλό περιορισμό) τότε τον θεωρούμε σαν έναν απλό κόμβο καρτεσιανού γινομένου.

2.2.2 Ευριστικός βελτιστοποιητής (heuristic optimizer)

2.2.2.1 Εισαγωγικά

Στόχος αυτού του αντικειμένου είναι να βελτιστοποιήσει ένα πλάνο εκτέλεσης χρησιμοποιώντας διάφορους ευριστικούς κανόνες. Όταν μιλάμε για πλάνο εκτέλεσης ουσιαστικά αναφερόμαστε σε ένα δέντρο το οποίο αναπαριστά μια σχέση σχεσιακής άλγεβρας (όπως αυτό της εικόνας 2). Το συγκεκριμένο τμήμα αναλαμβάνει να πάρει μια τέτοια σχέση και να πραγματοποιήσει κάποιες πράξεις - μετατροπές. Οι μετατροπές αυτές δεν θα αλλάξουν το αποτέλεσμα της επερώτησης (δηλαδή οι δύο σχέσεις θα είναι όμοιες ως προς το τελικό αποτέλεσμα) αλλά θα μας δώσουν καλύτερους χρόνους εκτέλεσης της επερώτησης. Αυτό θα συμβεί λόγω μικρότερων ενδιάμεσων αποτελεσμάτων αλλά και τεχνικών που μπορεί να χρησιμοποιηθούν (διάφορα είδη ζεύξεων σε αντίθεση με

καρτεσιανό γινόμενο κλπ) .

Όπως καταλαβαίνουμε το βασικό στοιχείο το οποίο κρύβεται πίσω από τον ευριστικό βελτιστοποιητή είναι η μετατροπή μιας έκφρασης η οποία είναι γραμμένη σε σχεσιακή άλγεβρα, σε μια άλλη ισοδύναμη της με μικρότερο κόστος εκτέλεσης. Για να το πετύχουμε αυτό θα μας βοηθήσει ένας (ευριστικός) αλγόριθμος που μας εγγυάται χαμηλότερο κόστος εκτέλεσης της επερώτησης, στις πιο πολλές περιπτώσεις, συνδυαζόμενος με ένα σύνολο κανόνων μετατροπής αλγεβρικών εκφράσεων έτσι ώστε να εξασφαλίζουμε ότι το αρχικό μας αποτέλεσμα είναι ισοδύναμο με το τελικό.

2.2.2.2 Κανόνες μετατροπής αλγεβρικών εκφράσεων

Οι κανόνες μετατροπής αλγεβρικών εκφράσεων που θα μας χρησιμεύσουν είναι οι παρακάτω:

1. Αντιμεταθετική ιδιότητα για καρτεσιανό γινόμενο (commutative law for Cartesian product)

$$E_1 \times E_2 \equiv E_2 \times E_1$$

2. Επικάλυψη επιλογών (cascade of selections)

$\sigma_{F_1}(\sigma_{F_2}(E)) \equiv \sigma_{F_1 \wedge F_2}(E)$ και αφού $F_1 \wedge F_2 = F_2 \wedge F_1$ τότε προκύπτει άμεσα το ότι οι επιλογές μπορούν να αντιμετατεθούν $\sigma_{F_1}(\sigma_{F_2}(E)) \equiv \sigma_{F_2}(\sigma_{F_1}(E))$

3. Αντιμετάθεση επιλογών και προβολών (commuting selections and projections)

Αν η συνθήκη F περιέχει μόνο τις ιδιότητες $A_1,...,A_n$ τότε

$$\pi_{A_1,...,A_n}(\sigma_F(E)) \equiv \sigma_F(\pi_{A_1,...,A_n}(E))$$

Πιο γενικά, αν η συνθήκη F περιέχει ακόμα τις ιδιότητες $B_1,...,B_m$ οι οποίες δεν είναι ανάμεσα στις $A_1,...,A_n$ τότε

$$\pi_{A_1,...,A_n}(\sigma_F(E)) \equiv \pi_{A_1,...,A_n}(\sigma_F(\pi_{A_1,...,A_n,B_1,...,B_m}(E)))$$

4. Αντιμετάθεση επιλογής με καρτεσιανό γινόμενο (commuting selection with Cartesian product). Αν όλες οι ιδιότητες που αναφέρονται στην F είναι ιδιότητες της E_1, τότε

$$\sigma_F(E_1 \times E_2) \equiv \sigma_F(E_1) \times E_2$$

Ακόμα αν η F είναι της μορφής $F_1 \wedge F_2$ όπου η F_1 περιέχει ιδιότητες από τη σχέση E_1 και η F_2 περιέχει ιδιότητες μόνο από τη σχέση E_2 τότε χρησιμοποιώντας τους κανόνες (1), (2) και (4) μπορούμε να καταλήξουμε στο $\sigma_F(E_1 \times E_2) \equiv \sigma_{F_1}(E_1) \times \sigma_{F_2}(E_2)$.

Ακόμα αν η F_1 περιέχει ιδιότητες μόνο από την E_1 αλλά η F_2 περιέχει ιδιότητες και από την E_1 και την E_2 τότε μπορούμε να καταλήξουμε στο $\sigma_F(E_1 \times E_2) \equiv \sigma_{F_2}(\sigma_{F_1}(E_1) \times E_2)$

5. Αντιμετάθεση προβολής με καρτεσιανό γινόμενο (commuting a projection with a Cartesian product)

Έστω ότι E_1 και E_2 είναι δύο σχεσιακές εκφράσεις. Έστω $A_1,...,A_n$ ένα σύνολο από ιδιότητες όπου οι $B_1,...,B_m$ είναι ιδιότητες της E_1 και $C_1,...,C_m$ ιδιότητες της E_2. Τότε

$$\pi_{A_1,...,A_n}(E_1 \times E_2) \equiv \pi_{B_1,...,B_m}(E_1) \times \pi_{C_1,...,C_k}(E_2)$$

2.2.2.3 Ευριστικός Αλγόριθμος

Η βασική ιδέα πίσω από τα βήματα του συγκεκριμένου αλγορίθμου είναι το ότι η πληροφορία που δεν μας είναι χρήσιμη και μπορεί να εξαλειφτεί στα χαμηλότερα επίπεδα του δέντρου να εξαλείφεται εκεί. Με τον συγκεκριμένο τρόπο αποφεύγουμε να μεταφέρουμε πλεονάζουσα πληροφορία από επίπεδο σε επίπεδο, κάτι που σημαίνει λιγότερο κόστος για τις ενδιάμεσες εγγραφές - διαγραφές.

Αλγόριθμοι οι οποίοι βασίζονται στην συγκεκριμένη ιδέα παρουσιάζονται από τον Hall [2], τον Minker [3], τον Pecherer [4] και τους Smith και Chang [5]. Κοινό στοιχείο όλων αυτών των αλγορίθμων είναι το ότι προσπαθούν να μετακινήσουν τις επιλογές όσο γίνεται

πιο χαμηλά στο σχεσιακό δένδρο έτσι ώστε να πετύχουν όσο γίνεται πιο γρήγορα απαλοιφή περιττών πλειάδων.

Ακολουθεί ο αλγόριθμος που χρησιμοποιούμε στην συγκεκριμένη εργασία όπου ακολουθεί και αυτός το ίδιο σκεπτικό:

Αλγόριθμος βελτιστοποίησης σχεσιακών εκφράσεων

Είσοδος: Ένα δένδρο το οποίο αναπαριστά μια έκφραση σχεσιακής άλγεβρας

Έξοδος: Ένα βελτιστοποιημένο δένδρο εκτέλεσης της συγκεκριμένης έκφρασης

Βήματα:

1. Χρησιμοποίησε τον κανόνα 2 για να ξεχωρίσεις κάθε επιλογή τύπου $\sigma_{F_1 \wedge ... \wedge F_n}(E)$ σε $\sigma_{F_1}(...(\sigma_{F_n}(E))...)$

2. Για κάθε επιλογή χρησιμοποίησε τους κανόνες 2 έως και 4 για να για να μετακινήσεις τις επιλογές όσο γίνεται πιο χαμηλά. (Αφορά τις επιλογές σύγκρισης στήλης με τιμή καθώς και επιλογές ζεύξης)

3. Αν βρεις δύο ή περισσότερες επιλογές μαζί χρησιμοποίησε τον κανόνα 2 για να μεταφέρεις όλους τους περιορισμούς σε μια επιλογή.

4. Σάρωσε το δένδρο κατά πλάτος. Αν συναντήσεις προβολή που ακολουθείται από επιλογή, χρησιμοποίησε τον κανόνα 3 για να δημιουργήσεις τις κατάλληλες προβολές κάτω από κόμβους επιλογής και ταυτόχρονα τον κανόνα 5 για να μετακινήσεις τις συγκεκριμένες προβολές στην κατάλληλη θέση μετά τον κόμβο καρτεσιανού γινομένου που μπορεί να ακολουθεί. Αν συναντήσεις μόνο κόμβο προβολής και μετά κατευθείαν κόμβο καρτεσιανού γινομένου τότε απλώς χρησιμοποίησε τον κανόνα 5 για να μετακινήσεις την προβολή κάτω από τον συγκεκριμένο κόμβο καρτεσιανού γινομένου.

5. Σε περίπτωση που υπάρχουν προβολές που είναι περιττές, δηλαδή προβάλουν όλες τις στήλες που βρίσκονται από κάτω τους τότε τις διαγράφουμε. Επιπλέον αν

στο ίδιο τμήμα κλαδιού ενός δένδρου βρεθούν δύο ίδιες προβολές τότε διαγράφεται αυτή που είναι πιο ψηλά στο συγκεκριμένο κλαδί. Ένα τέτοιο τμήμα ορίζεται από τους δύο πιο κοντινούς από τους εξής κόμβους, ρίζα και Χ κόμβοι και Τ κόμβοι.

2.2.2.4 Τρόποι βελτιστοποίησης (μερική και πλήρης)

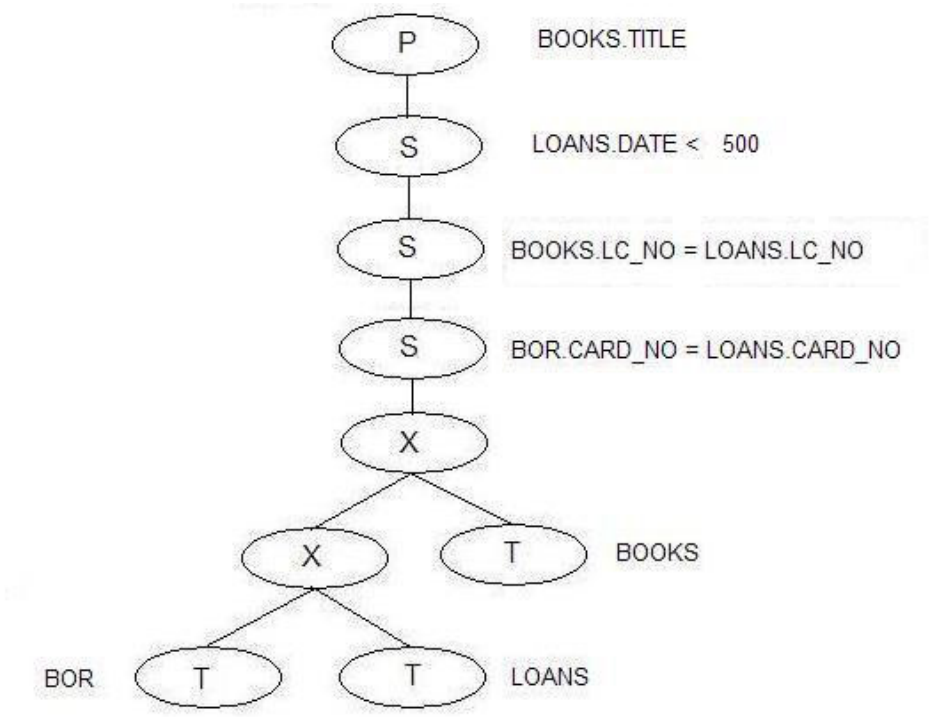
Ουσιαστικά ένας είναι ο τρόπος βελτιστοποίησης και είναι αυτός που περιγράφεται από τον αλγόριθμο της προηγούμενης παραγράφου. Όταν αναφέρουμε μερική βελτιστοποίηση εννοούμε ότι θα εφαρμόσουμε μόνο τα βήματα 1 έως 3 του παραπάνω αλγορίθμου ενώ όταν αναφέρουμε πλήρη βελτιστοποίηση εννοούμε ότι θα εφαρμόσουμε και τα 5 βήματα.

Κατά την μερική βελτιστοποίηση αυτό που ουσιαστικά πραγματοποιούμε είναι η προώθηση των επιλογών αλλά και των ζεύξεων όσο πιο χαμηλά γίνεται. Αυτό έχει ως αποτέλεσμα οι επιλογές που περιορίζουν την τιμή μιας στήλης να καταλήξουν ακριβώς πάνω από τον πίνακα της συγκεκριμένης στήλης και οι επιλογές που περιέχουν ζεύξεις να καταλήξουν ακριβώς πάνω από κόμβο καρτεσιανού γινομένου που στο αριστερό του παιδί θα έχει τον ένα πίνακα της ζεύξης και στο δεξί του παιδί θα έχει τον άλλο πίνακα της ζεύξης αυτής. Αποτέλεσμα της παραπάνω διαδικασίας είναι να έχουμε πια κόμβους S που περιέχουν ζεύξεις πάνω από κόμβους Χ. Οι συγκεκριμένοι κόμβοι αντιμετωπίζονται πια από τον εκτιμητή κόστους ως ζεύξεις και εφαρμόζεται ο κατάλληλος αλγόριθμος ζεύξης. Αν δεν πραγματοποιούσαμε αυτή την μερική βελτιστοποίηση και στέλναμε την επερώτηση όπως ήταν στο αρχικό της στάδιο στον εκτιμητή κόστους θα αντιμετώπιζε τους πιο πολλούς κόμβους Χ σαν καρτεσιανά γινόμενα (γιατί δεν θα υπήρχε ακριβώς από πάνω τους ο κόμβος S που θα περιείχε τις ζεύξεις). Κάτι τέτοιο θα ήταν καταστροφικό όμως για το κόστος της συγκεκριμένης επερώτησης και σίγουρα άδικη η σύγκρισή της με την αντίστοιχη πλήρως βελτιστοποιημένη επερώτηση και το κόστος αυτής.

Κατά την πλήρη βελτιστοποίηση πέρα από αυτά που κάνουμε κατά την μερική βελτιστοποίηση αναλαμβάνουμε να τοποθετήσουμε και προβολές στα κατάλληλα σημεία για την προφύλαξη των ενδιάμεσων αποτελεσμάτων από περιττές στήλες. Παράδειγμα πλήρης βελτιστοποίησης παρουσιάζεται στην παράγραφο που ακολουθεί.

2.2.2.5 Εφαρμογή Αλγορίθμου (Πλήρης Βελτιστοποίηση)

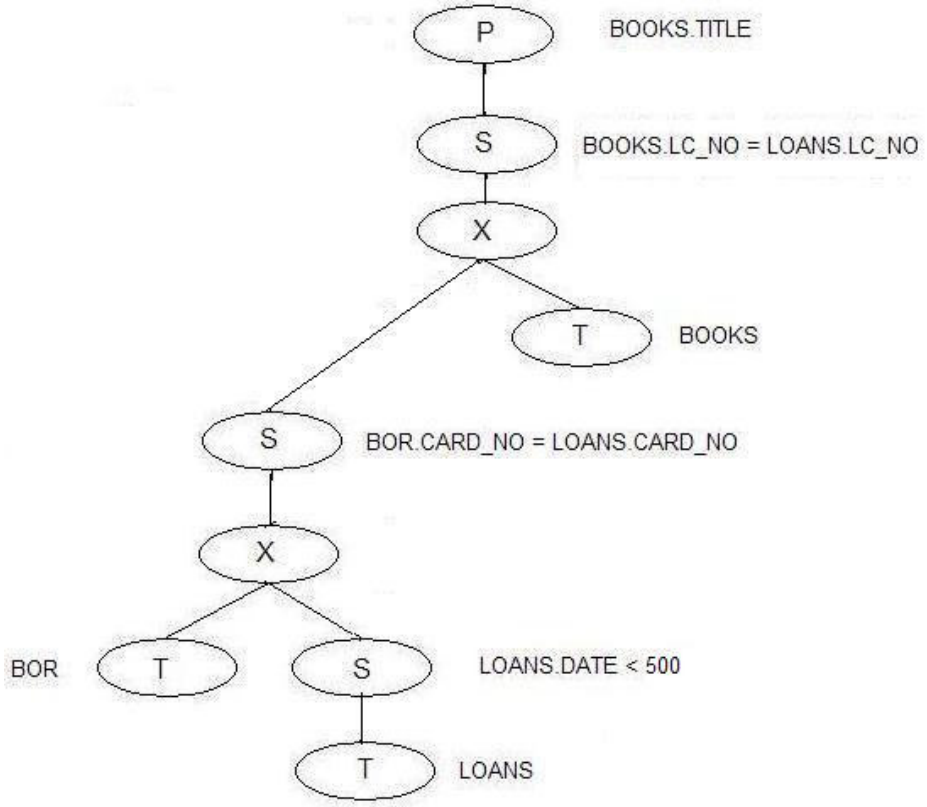
Στο συγκεκριμένο σημείο θα παρουσιάσουμε τα βήματα του αλγορίθμου στο δένδρο της σχεσιακής άλγεβρας που έχουμε στην Εικόνα 2. Πρώτο μας μέλημα είναι να χρησιμοποιήσουμε τον κανόνα 2 για να διαχωρίσουμε την δεύτερη επιλογή σε δύο ξεχωριστές επιλογές που η μια θα έχει ως όρισμα το (BOOKS.LC_NO = LOANS.LC_NO) και η δεύτερη το (BOR.CARD_NO = LOANS.CARD_NO). Το αποτέλεσμα της πράξης μας φαίνεται στην εικόνα που ακολουθεί.



Εικόνα 3: Σχεσιακό αλγεβρικό δένδρο επερώτησης χρήστη μετά το πρώτο βήμα του ευριστικού αλγορίθμου βελτιστοποίησης

Στην συνέχεια θα μεταφέρουμε κάθε κόμβο επιλογής όσο πιο χαμηλά γίνεται. Πιο συγκεκριμένα για τον κόμβο επιλογής με όρισμα (LOANS.DATE < 500) θα χρησιμοποιήσουμε τους κανόνες 2 και 4 και θα τον τοποθετήσουμε πάνω από τον κόμβο

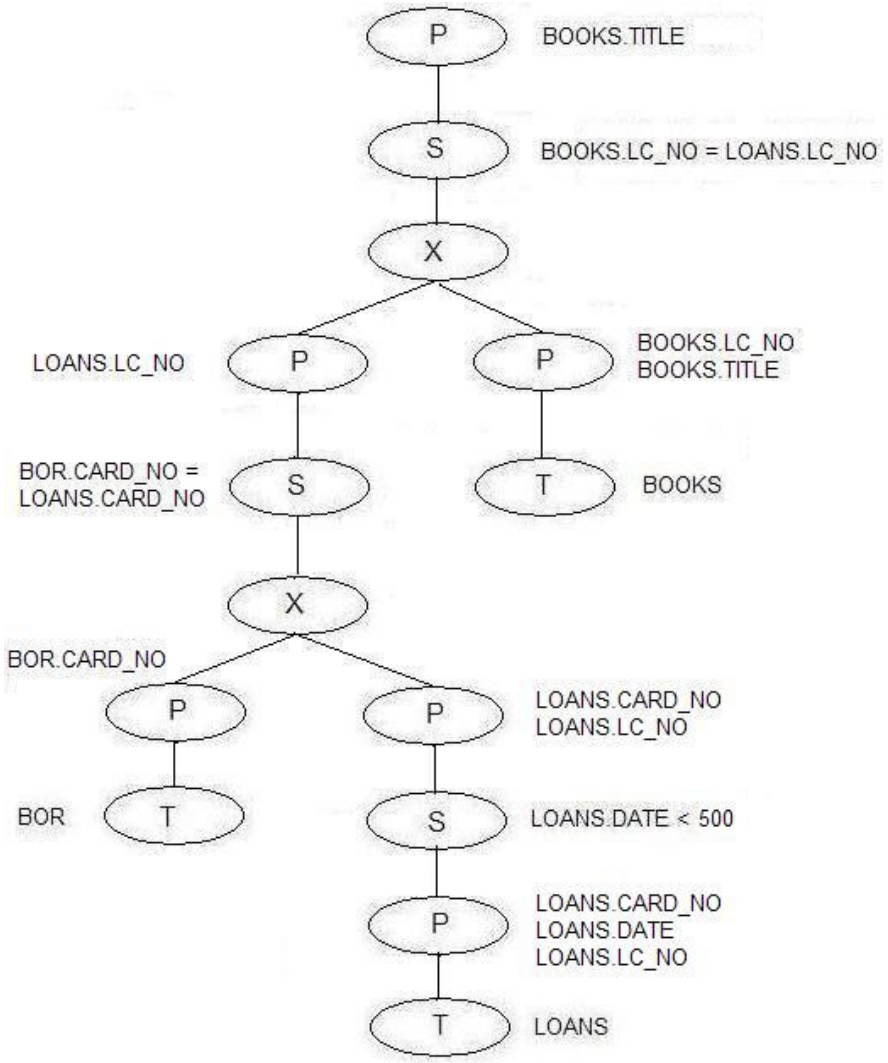
LOANS. Εφαρμόζοντας τους ίδιους κανόνες θα μεταφέρουμε τις υπόλοιπες δύο επιλογές πάνω ακριβώς από τα καρτεσιανά γινόμενα που τις αφορούν. Τα αποτελέσματα φαίνονται στην επόμενη εικόνα.



Εικόνα 4: Σχεσιακό αλγεβρικό δένδρο επερώτησης χρήστη μετά το τρίτο βήμα του ευριστικού αλγορίθμου βελτιστοποίησης

Επόμενο βήμα είναι να δημιουργήσουμε τις κατάλληλες προβολές στα σωστά σημεία. Χρησιμοποιώντας τον κανόνα 3 δημιουργούμε κάτω από την επιλογή S(BOOKS.LC_NO = LOANS.LC_NO) μια νέα προβολή η οποία θα προβάλει τις στήλες BOOKS.TITLE, BOOKS.LC_NO, LOANS.LC_NO. Στην συνέχεια χρησιμοποιούμε τον κανόνα 5 για να διαχωρίσουμε την συγκεκριμένη προβολή σε δύο μικρότερες και να βάλουμε ως αριστερό παιδί του καρτεσιανού γινομένου την προβολή με ορίσματα LOANS.LC_NO ενώ ως δεξί παιδί την προβολή με ορίσματα BOOKS.TITLE, BOOKS.LC_NO. Αφού πραγματοποιήσουμε αυτή την συγκεκριμένη αλλαγή για την προβολή και την επιλογή της ρίζας, την πραγματοποιούμε και για όλους τους άλλους

κόμβους του δένδρου σαρώνοντάς το κατά πλάτος. Τέλος πρέπει να δούμε αν στο δένδρο μας υπάρχουν περιττές προβολές. Αν βρεθεί προβολή η οποία προβάλει όλες τις στήλες οι οποίες είναι από κάτω της τότε δεν προσφέρει τίποτα και πρέπει να την διαγράψουμε. Το ίδιο θα κάνουμε και αν βρούμε δύο εντελώς ίδιες προβολές στο ίδιο τμήμα ενός κλαδιού (δες 5^{ο} βήμα του αλγορίθμου). Το τελικό δένδρο που προκύπτει από όλες τις παραπάνω πράξεις φαίνεται στην εικόνα που ακολουθεί.



Εικόνα 5: Σχεσιακό αλγεβρικό δένδρο επερώτησης χρήστη μετά το πέμπτο βήμα του ευριστικού αλγορίθμου βελτιστοποίησης

2.2.3 Εκτιμητής κόστους (cost estimator)

2.2.3.1 Εισαγωγικά

Κεντρικός στόχος του εκτιμητή κόστους είναι να κάνει μια εκτίμηση για το κόστος εκτέλεσης μιας αλγεβρικής έκφρασης που του δίνεται. Η συγκεκριμένη αλγεβρική έκφραση μπορεί να περιέχει πίνακες και όψεις αλλά και πράξεις πάνω σε αυτούς-αυτές. Και στις δύο περιπτώσεις πρέπει να επικοινωνήσει με το αντικείμενο που διαχειρίζεται τα στατιστικά της βάσης (database statistics manager) έτσι ώστε να λάβει τις κατάλληλες πληροφορίες για να κάνει την εκτίμηση αυτή. Αν η έκφραση περιέχει και υλοποιημένες όψεις τότε πρέπει να επικοινωνήσει και με το αντικείμενο που διαχειρίζεται τις υλοποιημένες όψεις αυτές (view manager) ώστε να λάβει στατιστικά που τις αφορούν.

Οι βασικές λειτουργίες του εκτιμητή κόστους είναι:

- η εκτίμηση του αριθμού των γραμμών και το μέγεθος της γραμμής σε κάθε ενδιάμεσο αλλά και στο τελικό αποτέλεσμα.
- η εκτίμηση του κόστους που χρειάζεται σε διάβασμα και γράψιμο (I/O) για να φτάσουμε στο κάθε ενδιάμεσο αλλά και στο τελικό αποτέλεσμα.

Οι συγκεκριμένες λειτουργίες πρέπει να πραγματοποιηθούν και με την συγκεκριμένη σειρά μια που η δεύτερη χρειάζεται τα αποτελέσματα της πρώτης.

2.2.3.2 Εκτίμηση αριθμού γραμμών και μέγεθος γραμμής

2.2.3.2.1 Περιγραφή αλγορίθμου

Η συγκεκριμένη λειτουργία δέχεται ως είσοδο μια αλγεβρική έκφραση σε μορφή δένδρου όπως είναι αυτή που παρουσιάζεται στην τελευταία εικόνα και στόχος της είναι να

βρει τον αριθμό γραμμών αλλά και το μέγεθος της γραμμής των αποτελεσμάτων σε κάθε κόμβο. Για να το πραγματοποιήσει αυτό αρχικά βρίσκει όλους τους κόμβους T (που συμβολίζουν πίνακες) , όλους τους κόμβους V (που συμβολίζουν υλοποιημένες όψεις) και όλους τους κόμβους X (που συμβολίζουν καρτεσιανό γινόμενο ή ζεύξη).

Αρχίζει από τους κόμβους T και V και βρίσκει τον αριθμό των γραμμών αλλά και το μέγεθος των γραμμών τους. Στην περίπτωση που συναντήσει πίνακα επικοινωνεί με το αντικείμενο το οποίο αποθηκεύει – διαχειρίζεται τα στατιστικά που αφορούν την βάση δεδομένων μας (δηλαδή τον database statistics manager) από το οποίο θα τα λάβει. Ο αριθμός των γραμμών είναι δεδομένος ενώ το μέγεθος των γραμμών (ουσιαστικά το μήκος της κάθε γραμμής σε bytes) είναι το άθροισμα του μήκους των στηλών οι οποίες προβάλλονται. Στην συγκεκριμένη περίπτωση μιλάμε ουσιαστικά για όλες τις στήλες. Στην περίπτωση που συναντήσει κάποια υλοποιημένη όψη τότε επικοινωνεί με τον view manager από τον οποίο θα λάβει τις αντίστοιχες πληροφορίες.

Αφού συμπληρωθούν τα στατιστικά στον συγκεκριμένο κόμβο τότε προχωράμε στον πατέρα του και συμπληρώνουμε και εκεί τα συγκεκριμένα στατιστικά.

- Αν ο κόμβος είναι προβολή (P κόμβος) τότε το ο αριθμός των γραμμών παραμένει σταθερός. Αυτό που αλλάζει (και για την ακρίβεια μειώνεται) είναι το μέγεθος των γραμμών. Δηλαδή κοιτάζουμε ποιες είναι οι στήλες οι οποίες προβάλλονται και στην συνέχεια ζητάμε για κάθε μια από αυτές το μήκος της. Το άθροισμα είναι και το μέγεθος της γραμμής της συγκεκριμένης προβολής.

- Αν ο κόμβος είναι επιλογή ή ζεύξη (S κόμβος) τότε υπολογίζουμε το ποσοστό των γραμμών που ικανοποιούν τα κριτήρια του συγκεκριμένου κόμβου. Ο αριθμός των γραμμών που περνάνε στο πιο πάνω επίπεδο είναι ίσος με το ποσοστό αυτό επί τον αριθμό των γραμμών που έρχονται από τον αμέσως πιο κάτω κόμβο. Στην συνέχεια θα παρουσιαστεί και αναλυτικά πως υπολογίζεται το συγκεκριμένο ποσοστό. Τέλος σημειώνουμε ότι το μέγεθος των γραμμών παραμένει σταθερό.

- Αν ο κόμβος είναι τύπου X τότε ανεξάρτητα τι συμβολίζει πραγματικά (καρτεσιανό γινόμενο ή ζεύξη αν έχει ως πατέρα κόμβο S ζεύξης) υπολογίζουμε τον αριθμό των γραμμών και το μήκος γραμμών που θα είχε αν ήταν καρτεσιανό γινόμενο. Ο αριθμός των γραμμών είναι το γινόμενο του αριθμού γραμμών των δύο παιδιών του

ενώ το μήκος των γραμμών είναι το άθροισμα των μηκών των γραμμών των δύο παιδιών του.

Όταν τελειώνουμε την επεξεργασία κάθε κόμβου κοιτάζουμε αν ο πατέρας του είναι κόμβος τύπου Χ. Στην περίπτωση που ισχύει κάτι τέτοιο σημειώνουμε στον συγκεκριμένο κόμβο Χ ότι το αντίστοιχο παιδί του έχει τα συγκεκριμένα στατιστικά συμπληρωμένα. Βέβαια αν βρεθεί η ρίζα του δένδρου τότε σταματάει η διαδικασία και θεωρούμε ότι ολοκληρώθηκε. (σε αυτή την περίπτωση το δένδρο θα είναι ουσιαστικά μια αλυσίδα από κόμβους χωρίς διακλάδωση που θα καταλήγει στον συγκεκριμένο κόμβο Τ)

Όταν ολοκληρώσουμε την παραπάνω διαδικασία για όλους τους κόμβους Τ του δένδρου μας τότε προχωράμε στην επεξεργασία κόμβων τύπου Χ. Για να μπορέσουμε να επεξεργαστούμε έναν τέτοιο κόμβο πρέπει να έχουν συμπληρωμένο τον αριθμό αλλά και το μέγεθος γραμμών και τα δύο παιδιά του. Αυτό όμως δεν συμβαίνει για όλους τους κόμβους Χ διότι δεν έχουν όλοι οι κόμβοι παιδιά που καταλήγουν άμεσα σε κόμβο Τ ή V. Αυτό μπορούμε να το δούμε και στην εικόνα 5. Αν έχουμε ολοκληρώσει την επεξεργασία όλων των κλαδιών που καταλήγουν άμεσα σε κόμβο Τ (χωρίς να παρεμβληθεί κάποιος Χ κόμβος) τότε ο κόμβος Χ που βρίσκεται πιο χαμηλά στο δένδρο είναι έτοιμος προς επεξεργασία σε αντίθεση με τον άλλο κόμβο Χ που περιμένει ακόμα στοιχεία για το αριστερό του παιδί.

Σε κάθε περίπτωση πάντως υπάρχει σίγουρα ένας κόμβος Χ έτοιμος προς επεξεργασία. Καθώς ολοκληρώνουμε την επεξεργασία κλαδιών που έχουν ως αφετηρία έτοιμους προς επεξεργασία κόμβους τύπου Χ τότε συμπληρώνονται και τα στατιστικά στους κόμβους Χ που ήταν σε κατάσταση μη συμπληρωμένη (όπως ο κόμβος Χ που βρίσκεται πιο ψηλά στο δένδρο). Εφαρμόζοντας την συγκεκριμένη διαδικασία θα καταλήξουμε στην ρίζα του δένδρου όπου και ολοκληρώνεται ο στόχος της συγκεκριμένης συνάρτησης.

2.2.3.2.2 Υπολογισμός ποσοστού γραμμών που ικανοποιούν μια επιλογή

Το σημείο με το οποίο δεν ασχοληθήκαμε στον παραπάνω αλγόριθμο είναι το πως υπολογίζουμε το ποσοστό των γραμμών που ικανοποιούν έναν κόμβο S. Ένας τέτοιος

κόμβος επιλογής μπορεί να έχει έναν ή περισσότερους περιορισμούς. Οι συγκεκριμένοι περιορισμοί μπορεί να συμβολίζουν είτε ζεύξη ανάμεσα σε στήλες δύο διαφορετικών πινάκων είτε περιορισμό της τιμής κάποιας στήλης με βάση κάποια σταθερά. Όποια και να είναι η περίπτωση υπολογίζεται το αντίστοιχο ποσοστό και στο τέλος ως αποτέλεσμα λαμβάνουμε το γινόμενο των ποσοστών αυτών (για την ακρίβεια γινόμενο των επιμέρους κλασμάτων) μια και κάνουμε την παραδοχή ότι υπάρχει ανεξαρτησία ανάμεσα στους περιορισμούς αυτούς. Αναλυτική περιγραφή των εκτιμήσεων που ακολουθούν μπορεί να βρεθεί και στο [6] αλλά ας δούμε πιο προσεκτικά τι κάνουμε σε κάθε μια από τις δύο περιπτώσεις.

2.2.3.2.2.1 Περίπτωση σύγκρισης τιμής στήλης πίνακα με μια σταθερά

1. Στην περίπτωση που θέλουμε η τιμή της στήλης να είναι ίση με μια σταθερά τότε ο αριθμός των εγγραφών που ικανοποιούν την συγκεκριμένη συνθήκη είναι $\frac{n_r}{V(A,r)}$ όπου n_r είναι ο αριθμός των εγγραφών ενώ $V(A,r)$ είναι ο αριθμός των διαφορετικών τιμών που εμφανίζονται στην στήλη A του πίνακα r. Το ποσοστό που χρειαζόμαστε είναι το $\frac{1}{V(A,r)} * 100\%$. Μια λεπτομέρεια που δεν πρέπει να μας διαφύγει είναι ότι το παραπάνω αποτέλεσμα θα το επιστρέψουμε μόνο στην περίπτωση που η σταθερά είναι μεγαλύτερη ή ίση από την ελάχιστη τιμή και μικρότερη ή ίση από την μέγιστη τιμή της συγκεκριμένης στήλης.

2. Στην περίπτωση που θέλουμε να πραγματοποιήσουμε κάποιου είδους σύγκριση (< ή >) με κάποια σταθερά βρίσκουμε τον αριθμό των εγγραφών του αποτελέσματος από τον ακόλουθο τύπο $n_r * \frac{v - \min(A,r)}{\max(A,r) - \min(A,r)}$ και όλα αυτά αν η τιμή v βρίσκεται ανάμεσα στη ελάχιστη και στην μέγιστη τιμή της στήλης A. Συνεπώς το ποσοστό στην συγκεκριμένη περίπτωση θα είναι :

   - 0% αν $v < \min(A,r)$ ή $v > \max(A,r)$.

- $\frac{v - \min(A,r)}{\max(A,r) - \min(A,r)} * 100\%$ στις υπόλοιπες περιπτώσεις.

2.2.3.2.2.2 Περίπτωση ζεύξης

Στην περίπτωση που ο κόμβος S συμβολίζει κάποια ζεύξη ανάμεσα στους πίνακες R και S τότε ισχύουν τα εξής :

- Αν $R \cap S = \{A\}$ δεν είναι κλειδί είτε για τον R είτε για τον S τότε ο αριθμός των εγγραφών που θα προκύψουν από την ζεύξη των R και S θα είναι είτε $\frac{n_r * n_s}{V(A,s)}$ είτε $\frac{n_r * n_s}{V(A,r)}$. Το πιο μικρό από τα δύο κλάσματα είναι και πιθανότατα και η πιο ακριβής εκτίμηση. Συνεπώς το ποσοστό που αναζητούμε θα είναι :
  - $\frac{1}{V(A,s)} * 100\%$, αν $V(A,s) > V(A,r)$
  - $\frac{1}{V(A,r)} * 100\%$, αν $V(A,s) \le V(A,r)$
- Αν $R \cap S = \{A\}$ είναι κλειδί για το R τότε μια γραμμή από το S θα συζευχθεί το πολύ με μια γραμμή από το R. Συνεπώς ο αριθμός των γραμμών της ζεύξης μεταξύ R και S δεν θα είναι μεγαλύτερος από τον αριθμό των γραμμών του πίνακα S. Οπότε το ποσοστό θα είναι $\le \frac{1}{n_s} * 100\%$.
- Όμοια θα είναι και αν $R \cap S = \{A\}$ είναι κλειδί για το S. Το ποσοστό σε αυτή την περίπτωση θα είναι $\le \frac{1}{n_r} * 100\%$.
- Αν $R \cap S = \{A\}$ είναι ξένο κλειδί στον S το οποίο αναφέρεται στον R τότε ο αριθμός των εγγραφών της ζεύξης μεταξύ R και S θα είναι ακριβώς ίσος με τον αριθμό των εγγραφών στο S. Στο σημείο αυτό δεν πρέπει να θυμηθούμε ότι από την στιγμή

που το A είναι ξένο κλειδί στον S το οποίο αναφέρεται στον R τότε οι τιμές που λαμβάνει η συγκεκριμένη στήλη του S πρέπει να βρίσκονται και στον R. Οπότε το ποσοστό στην συγκεκριμένη περίπτωση θα είναι ίσο με $\frac{1}{n_s} * 100\%$

- Όμοια αν $R \cap S = \{A\}$ είναι ξένο κλειδί στον R το οποίο αναφέρεται στον S τότε το ποσοστό θα είναι ίσο με $\frac{1}{n_r} * 100\%$

- Τέλος, αν $R \cap S = \varnothing$ τότε ο αριθμός των γραμμών της ζεύξης ανάμεσα σε R και S θα είναι ίσος με $n_r * n_s$ (δηλαδή πραγματοποιούμε πράξη καρτεσιανού γινομένου) και το ποσοστό θα είναι 100%.

2.2.3.3 Εκτίμηση κόστους σε διάβασμα – γράψιμο

Στην συγκεκριμένη ενότητα θα περιγράψουμε τον τρόπο με τον οποίο υπολογίζουμε το συνολικό κόστος που χρειάζεται να δαπανήσουμε σε διάβασμα και γράψιμο στον δίσκο για να φτάσουμε σε κάθε κόμβο του δένδρου που μας δόθηκε (όπως είναι αυτό της εικόνας 5). Έχουμε υπολογίσει προηγουμένως το μέγεθος (σε αριθμό και μέγεθος γραμμών) που θα έχουν τα ενδιάμεσα αποτελέσματα σε κάθε κόμβο και τα στοιχεία αυτά θα μας χρησιμεύσουν και στον υπολογισμό του κόστους που θέλουμε να πραγματοποιήσουμε. Στην συνέχεια ακολουθεί η περιγραφή του αλγορίθμου εκτίμησης κόστους και ακολουθούν αναλύσεις για το κόστος εφαρμογής του αλγόριθμου ζεύξης εμφωλιασμένων βρόχων καθώς και συγχώνευσης σάρωσης.

2.2.3.3.1 Περιγραφή αλγορίθμου εκτίμησης κόστους

Κατά την συγκεκριμένη λειτουργία ο εκτιμητής κόστους αφού έχει υπολογίσει τον αριθμό των γραμμών αλλά και το μέγεθός τους σε κάθε στάδιο αναλαμβάνει να βρει και το κόστος που χρειάζεται να δαπανήσει ένα ΣΔΒΔ σε διάβασμα και γράψιμο για να φτάσει στο συγκεκριμένο στάδιο. Για να το πετύχει αυτό ακολουθεί τον τρόπο εξερεύνησης του δένδρου που ακολουθήθηκε και στον παραπάνω αλγόριθμο. Θα αρχίσει από τους κόμβους

Τ που συμβολίζουν πίνακες και τους κόμβους V που συμβολίζουν όψεις. Διαλέγοντας κάθε φορά έναν από αυτούς θα συμπληρώνει τα στατιστικά σε όλους τους κόμβους ανάμεσα στον συγκεκριμένο κόμβο Τ ή V και στον πρώτο κόμβο Χ που θα βρει. Στην περίπτωση που δεν υπάρχει κόμβος Χ για να τερματίσει τότε θα τερματίσει μόλις συναντήσει την ρίζα του δένδρου. Μόλις συμπληρωθούν όλα τα στατιστικά στα κλαδιά που έχουν ως κατώτερο κόμβο έναν κόμβο Τ ή V τότε θα διαλέξει έναν Χ κόμβο ο οποίος θα έχει συμπληρωμένα με στατιστικά κόστους και τα δυο του παιδιά. Θα ακολουθηθεί η ίδια διαδικασία που ακολουθήθηκε και προηγουμένως και θα τερματιστεί μόλις συναντήσουμε την ρίζα του δένδρου.

Αξίζει σε αυτό το σημείο να δούμε πιο προσεκτικά το πως υπολογίζει ο αλγόριθμος το κόστος ανάλογα με τον κόμβο που συναντά.

- Αν συναντήσει κόμβο Τ (ουσιαστικά αρχίζει από τους συγκεκριμένους κόμβους) τότε το κόστος επεξεργασίας τους είναι ουσιαστικά το κόστος ανάγνωσης των δεδομένων τους. Αυτό θα είναι ίσο με το γινόμενο του αριθμού των γραμμών των πινάκων επί το μέγεθός τους. Όπως αναφέραμε και σε προηγούμενη παράγραφο, το μέγεθος της γραμμής ενός πίνακα είναι ίσο με το άθροισμα των μεγεθών των επιμέρους στηλών του και οι πληροφορίες αυτές είναι ήδη διαθέσιμες μια που έχουν υπολογιστεί στην αντίστοιχη λειτουργία του εκτιμητή κόστους που έχει ήδη προηγηθεί. Ακόμα πρέπει να επισημάνουμε ότι μια που ο κόμβος Τ (όπως και ο Χ) είναι ο πρώτος κόμβος προς επεξεργασία (δεν προηγούνται άλλοι κόμβοι κάτω από αυτόν) δεν υπάρχει κάποιο επιπλέον κόστος για να φτάσουμε σε αυτόν πέρα από το κόστος του ανάγνωσης του ίδιου του πίνακα.

- Αν συναντήσουμε κόμβο V τότε ο συγκεκριμένος κόμβος αναπαριστά μια υλοποιημένη όψη. Ο τρόπος αντιμετώπισης μιας υλοποιημένης όψης είναι όπως και αυτή ενός πίνακα. Το κόστος επεξεργασίας του συγκεκριμένου κόμβου είναι ουσιαστικά το κόστος ανάγνωσης των δεδομένων της υλοποιημένης όψης το οποίο θα είναι ίσο με το γινόμενο του αριθμού των γραμμών της επί τον μέγεθος των γραμμών αυτών. Επειδή ο κόμβος V (όπως και ο Τ) είναι ο πρώτος κόμβος προς επεξεργασία πρέπει να επισημάνουμε ότι το συνολικό κόστος για να φτάσουμε στον συγκεκριμένο κόμβο θα είναι ίσο με το κόστος του κόμβου αυτού και μόνο, μια που δεν προηγείται κάποιος άλλος κόμβος.

- Αν συναντήσουμε κόμβο P τότε θα έχουμε προβολή. Το κόστος της προβολής είναι μηδενικό (από άποψη διαβάσματος – γραψίματος στον δίσκο) μια που όπως έρχονται τα δεδομένα από το πιο κάτω επίπεδο πραγματοποιείται η προβολή και προωθούνται άμεσα στο επόμενο επίπεδο. Το συνολικό κόστος για να φτάσουμε στον κόμβο της προβολής είναι ακριβώς όσο και το συνολικό κόστος του να φτάσουμε στον κόμβο που βρίσκεται ακριβώς κάτω από τον κόμβο της προβολής.

- Αν συναντήσουμε κόμβο S τότε θα υπάρχουν δύο ενδεχόμενα. Είτε ο συγκεκριμένος κόμβος να συμβολίζει κάποια – κάποιες ζεύξεις, είτε να συμβολίζει απλό περιορισμό τιμής κάποιας στήλης – κάποιων στηλών ενός πίνακα με βάση μια σταθερά. Και τα δύο αυτά ενδεχόμενα δεν μπορούν να συμβούν ταυτόχρονα διότι δεν θα το επιτρέψει ο ευριστικός βελτιστοποιητής. Ο απλός περιορισμός τιμής θα τοποθετηθεί μόνος του σε έναν κόμβο S ή μαζί με άλλους περιορισμούς τιμής στον ίδιο κόμβο S αν αναφέρονται στον πίνακα που είναι ακριβώς από κάτω τους στον αντίστοιχο κόμβο T. Οι ζεύξεις όμως αναφέρονται σε δύο διαφορετικούς πίνακες οπότε θα βρίσκονται αναγκαστικά πάνω από έναν κόμβο X ο οποίος θα έχει στο αριστερό του παιδί τον ένα πίνακα και στο δεξί του παιδί τον άλλο πίνακα της ζεύξης.

    - Στην περίπτωση που ο κόμβος S έχει απλούς περιορισμούς τιμής με βάση μια σταθερά τότε θεωρούμε ότι το κόστος εφαρμογής του περιορισμού είναι μηδενικό (από άποψη διαβάσματος – γραψίματος στον δίσκο). Τα δεδομένα έρχονται από το πιο κάτω επίπεδο και ανάλογα αν πληρούν τα κριτήρια προωθούνται στο αμέσως επόμενο επίπεδο ή όχι. Το συνολικό κόστος μέχρι τον συγκεκριμένο κόμβο ισούται με το συνολικό κόστος που χρειαζόμαστε να δαπανήσουμε για να φτάσουμε στον ακριβώς από κάτω κόμβο.

    - Στην περίπτωση που ο κόμβος S περιέχει μια ή περισσότερες ζεύξεις τότε θα χρησιμοποιήσουμε κάποιον αλγόριθμο ζεύξης και θα υπολογίσουμε με βάση αυτόν το συνολικό κόστος που χρειάζεται για να πραγματοποιηθεί η ζεύξη αυτή. Οι αλγόριθμοι που χρησιμοποιούνται στην συγκεκριμένη εργασία είναι οι εμφωλιασμένοι βρόχοι καθώς και η συγχώνευση – σάρωση. Θα εξετάσουμε στην συνέχεια αναλυτικά το συνολικό κόστος που προκύπτει με την εφαρμογή του κάθε αλγορίθμου.

- Αν συναντήσουμε κόμβο Χ τότε ο συγκεκριμένος κόμβος μπορεί να συμβολίζει είτε καρτεσιανό γινόμενο είτε ζεύξη. Αν πατέρας του κόμβου Χ είναι κόμβος S που περιέχει ζεύξεις τότε θεωρούμε ότι οι δύο αυτοί κόμβοι (S και Χ) συμβολίζουν μια ζεύξη. Το συνολικό κόστος στην περίπτωση αυτή θα αναγράφεται στον κόμβο S ενώ ο κόμβος Χ θα περιέχει στο πεδίο του κόστους 0. Στην περίπτωση που ως πατέρας του Χ κόμβου είναι ένας οποιοδήποτε άλλος κόμβος τότε θεωρούμε ότι ο συγκεκριμένος κόμβος Χ είναι κόμβος καρτεσιανού γινομένου και το κόστος υπολογισμού του είναι ίσο με το κόστος εφαρμογής του αλγορίθμου ζεύξης εμφωλιασμένων βρόχων.

2.2.3.3.2 Αλγόριθμος ζεύξης εμφωλιασμένων βρόχων

Το κόστος του αλγορίθμου ζεύξης εμφωλιασμένων βρόχων δίνεται από την παρακάτω σχέση:

$$|R| + \left\lceil \frac{|R|}{|M|-2} \right\rceil * |S| \qquad (1)$$

όπου $|M|$ τα συνολικά block που χωράει η μνήμη του συστήματός μας, $|R|$ τα συνολικά block της σχέσης R, $|S|$ τα συνολικά block της σχέσης S και $|R| \leq |S|$.

Το παραπάνω κόστος προκύπτει με βάση το ακόλουθο συλλογισμό: Προσπαθούμε να φέρουμε όσο γίνεται περισσότερα blocks της μικρότερης σχέσης στην μνήμη (δηλαδή της R). Ο μέγιστος αριθμός blocks που μπορούμε να φέρουμε είναι $|M|-2$ και αυτό γιατί χρειάζεται να κρατήσουμε ένα block για να φέρνουμε blocks της S και ένα block για το αποτέλεσμα της ζεύξης (μέχρι να γεμίσει και να το στείλουμε στον δίσκο). Αφού φέρουμε τα πρώτα $|M|-2$ block της R στην μνήμη θα φέρουμε ένα-ένα τα block της S και θα εξετάζουμε αν πραγματοποιείται ζεύξη. Στην συνέχεια θα φέρουμε τα επόμενα $|M|-2$ block της R και θα πραγματοποιήσουμε την ίδια διαδικασία. Το παραπάνω σκεπτικό θα επαναληφθεί μέχρι να περάσουν όλα τα block της R από την μνήμη, δηλαδή θα επαναληφθεί για $\left\lceil \frac{|R|}{|M|-2} \right\rceil$ φορές. Τόσες φορές θα χρειαστεί δηλαδή να φέρουμε τα block

του S στην μνήμη. Στο συγκεκριμένο κόστος πρέπει να προσθέσουμε και τον συνολικό αριθμό blocks της R τα οποία θα τα φέρουμε μια φορά στην μνήμη. Συνεπώς καταλήγουμε στην παραπάνω εξίσωση.

Αυτό που πρέπει να προσέξουμε είναι ότι η συγκεκριμένη εξίσωση μας δίνει το κόστος του αλγορίθμου σε blocks. Επειδή εμείς θέλουμε το κόστος σε bytes πρέπει να πολλαπλασιάσουμε τον αριθμό αυτό με το μέγεθος του block. Κάτι ακόμα που δεν πρέπει να μας διαφύγει είναι ότι ο συγκεκριμένος αλγόριθμος θεωρεί ότι τα δεδομένα είναι αποθηκευμένα στον δίσκο και τα διαβάζει από εκεί. Στην περίπτωση που τα αποτελέσματα έρχονται από άλλες πράξεις όπως προβολές ή επιλογές θα πρέπει να αποθηκευτούν πρώτα στον δίσκο και μετά να αρχίσει να επεξεργάζεται τα δεδομένα αυτά ο αλγόριθμος. Οπότε πρέπει να λάβουμε και αυτό το κόστος υπ' όψιν μας πέρα από το κόστος του αλγορίθμου. Στην περίπτωση που τα δεδομένα έρχονται άμεσα από πίνακα ή κάποια υλοποιημένη όψη χωρίς την μεσολάβηση κάποιας πράξης τότε προφανώς δεν έχει νόημα να διαβαστούν από τον αντίστοιχο κόμβο T ή V και στην συνέχεια να αποθηκευτούν αυτούσια για να τα επεξεργαστεί ο συγκεκριμένος αλγόριθμος. Αυτό που κάνουμε είναι να θεωρούμε ότι ο αλγόριθμος τα διαβάζει κατευθείαν από το δίσκο μη λαμβάνοντας υπ' όψιν μας το κόστος διαβάσματος που έχει ο κόμβος T ή V.

Τέλος στον συγκεκριμένο αλγόριθμο έχει ληφθεί μέριμνα για την περίπτωση που για κάποια από τις δύο σχέσεις που συμμετέχουν στην ζεύξη υπάρχει ευρετήριο (η σχέση αυτή πρέπει να είναι πίνακας). Πιο συγκεκριμένα αν υπάρχει για παράδειγμα ευρετήριο στις στήλες της σχέσης R που συμμετέχουν στην ζεύξη τότε θα χρησιμοποιηθεί για την προσπέλαση της και το συνολικό κόστος στην περίπτωση αυτή θα είναι :

$$|S| + \{S\} * (IndexCost + 1) \quad , \qquad (2)$$

όπου $|S|$ τα συνολικά block της σχέσης S, $\{S\}$ οι συνολικές πλειάδες της σχέσης S και $IndexCost$ το κόστος προσπέλασης του ευρετηρίου.

Το σημείο που πρέπει να επισημάνουμε στην παραπάνω σχέση είναι ότι το

$IndexCost$ είναι ο αριθμός των blocks του ευρετηρίου που πρέπει να προσπελάσουμε μέχρι να βρούμε τον δείκτη προς το block δεδομένων της R που μας ενδιαφέρει. Φυσικά στο συγκεκριμένο κόστος πρέπει να προσθέσουμε και την μονάδα μια που πρέπει να ανακτήσουμε και το συγκεκριμένο block της R.Τέλος δεν πρέπει να ξεχάσουμε ότι όπως και η σχέση 1 έτσι και η σχέση 2 μας δίνει το κόστος του αλγορίθμου εμφωλιασμένων βρόχων σε blocks. Για να βρούμε το κόστος σε bytes πρέπει να πολλαπλασιάσουμε το συγκεκριμένο αποτέλεσμα με το μέγεθος του block.

Συνοψίζοντας, το κόστος του συγκεκριμένου αλγορίθμου δίνεται από την σχέση 1. Στην περίπτωση που έχουμε και ευρετήριο που μπορεί να χρησιμοποιηθεί εξετάζουμε το κόστος που προκύπτει αν το χρησιμοποιήσουμε (το οποίο μας δίνεται από την σχέση 2) και αν αυτό είναι μικρότερο από αυτό της σχέσης 1 τότε ακολουθούμε αυτή την μέθοδο του ευρετηρίου.

2.2.3.3.3 Αλγόριθμος ζεύξης συγχώνευσης σάρωσης

Σύμφωνα με τον απλό αλγόριθμο συγχώνευσης σάρωσης [7] πρέπει αρχικά να ταξινομήσουμε την κάθε σχέση με βάση την στήλη που λαμβάνει μέρος στην ζεύξη και στην συνέχεια να σαρώσουμε και τις δύο ταξινομημένες σχέσεις μαζί έτσι ώστε να πραγματοποιήσουμε την ζεύξη. Το κόστος του αλγορίθμου ζεύξης συγχώνευσης σάρωσης στην περίπτωση που καμία από τις δύο σχέσεις που λαμβάνουν μέρος στην ζεύξη δεν είναι ήδη ταξινομημένη είναι το εξής:

$$2*|R|*\left(1+\log_{|M|-1}\frac{|R|}{|M|}\right)+2*|S|*\left(1+\log_{|M|-1}\frac{|S|}{|M|}\right)+|R|+|S| \quad , \qquad (3)$$

όπου $|R|$ είναι ο αριθμός των blocks της σχέσης R, $|S|$ είναι ο αριθμός των block της σχέσης S και $|M|$ είναι ο αριθμός των block που χωράνε στην μνήμη. Το κόστος ταξινόμησης της πρώτης σχέσης είναι $2*|R|*\left(1+\log_{|M|-1}\frac{|R|}{|M|}\right)$ ενώ το κόστος ταξινόμησης

της δεύτερης σχέσης είναι $2*|S|*\left(1+\log_{|M|-1}\frac{|S|}{|M|}\right)$. Το συγκεκριμένο κόστος περιλαμβάνει το κόστος διαβάσματος από τον δίσκο της αρχικής μη ταξινομημένης σχέσης, το κόστος σε διάβασμα και γράψιμο που χρειάζεται κατά την ταξινόμηση και τέλος το κόστος της αποθήκευσης του ταξινομημένου αποτελέσματος στον δίσκο. Αναλυτικά ο αλγόριθμος της ταξινόμησης μπορεί να βρεθεί στο [8]. Στην συνέχεια θα χρειαστούν να διαβαστούν μια φορά οι σχέσεις R και S για να πραγματοποιηθεί η ζεύξη.

Το κόστος που μας δίνεται από την σχέση 3 είναι το κόστος σε blocks του συγκεκριμένου αλγορίθμου. Επειδή εμείς επιθυμούμε να λάβουμε το κόστος σε bytes πρέπει να πολλαπλασιάσουμε το συγκεκριμένο κόστος με το μέγεθος του block.

Στην περίπτωση που κάποια από τις σχέσεις είναι ήδη ταξινομημένη τότε προφανώς δεν χρειάζεται να ταξινομηθεί εκ νέου. Οπότε το κόστος σε blocks που χρειάζεται ο απλός αλγόριθμος της συγχώνευσης σάρωσης διαμορφώνεται όπως φαίνεται παρακάτω.

- Αν οι σχέσεις R και S δεν είναι ήδη ταξινομημένες τότε το κόστος σε blocks θα είναι :

$$2*|R|*\left(1+\log_{|M|-1}\frac{|R|}{|M|}\right)+2*|S|*\left(1+\log_{|M|-1}\frac{|S|}{|M|}\right)+|R|+|S| \qquad (3)$$

- Αν η σχέση R είναι ήδη ταξινομημένη και η S δεν είναι τότε το κόστος σε blocks θα είναι :

$$2*|S|*\left(1+\log_{|M|-1}\frac{|S|}{|M|}\right)+|R|+|S| \qquad (4)$$

- Αν η σχέση S είναι ήδη ταξινομημένη και η σχέση R δεν είναι τότε το κόστος σε blocks θα είναι :

$$2*|R|*\left(1+\log_{|M|-1}\frac{|R|}{|M|}\right)+|R|+|S| \qquad (5)$$

- Αν και οι δύο σχέσεις είναι ήδη ταξινομημένες τότε το κόστος σε blocks θα είναι :

$$|R|+|S| \qquad (6)$$

Στην περίπτωση που η μνήμη που διαθέτουμε είναι αρκετή και πιο συγκεκριμένα $|M|>\sqrt{|S|}$ (όπου $|S|\geq|R|$) τότε μπορούμε να χρησιμοποιήσουμε μια παραλλαγή του παραπάνω αλγορίθμου που παρουσιάστηκε στο [9]. Με βάση την συγκεκριμένη παραλλαγή ο αλγόριθμος εκτελείται σε δύο φάσεις. Κατά τη πρώτη φάση σωληνώνουμε το διάβασμα με την ταξινόμηση ενώ στην δεύτερη φάση πραγματοποιείται παράλληλη συγχώνευση και ζεύξη των σχέσεων R και S.

Ας δούμε όμως πιο αναλυτικά τι συμβαίνει. Κατά την πρώτη φάση φέρνουμε όσα περισσότερα blocks από μια σχέση μπορούμε στην μνήμη και δημιουργούμε «ταξινομημένα τρεξίματα». Δηλαδή αρχίζοντας θα φέρουμε $M-1$ blocks της S και θα κρατήσουμε ένα block για να τοποθετούμε τα αποτελέσματα που θα στείλουμε προς αποθήκευση. Από τα συγκεκριμένα blocks που είναι στην μνήμη διαλέγουμε τις μικρότερες τιμές και τις τοποθετούμε στο block που θα στείλουμε στον δίσκο. Μόλις γεμίσει το block αυτό το αποθηκεύουμε ενώ όταν αδειάσει κάποιο block από αυτά της S που είναι στην μνήμη τότε φέρνουμε ένα νέο. Μόνος περιορισμός στην παραπάνω διαδικασία είναι ότι θέλουμε τα blocks που στέλνουμε στον δίσκο να είναι ταξινομημένα, δηλαδή και οι τιμές μέσα στο ίδιο το block να είναι ταξινομημένες αλλά και οι τιμές που θα έχει ένα block πρέπει να είναι μεγαλύτερες ή ίσες από το ακριβώς προηγούμενο block που αποθηκεύτηκε. Η συγκεκριμένη διαδικασία μας οδηγεί στην δημιουργία σειρών από block τα οποία θα είναι ταξινομημένα (ταξινομημένα τρεξίματα). Στην χειρότερη περίπτωση ένα τέτοιο τρέξιμο θα αποτελείται από $M-1$ blocks ενώ κατά μέσο όρο θα αποτελείται από περίπου $2*M$ blocks, όπου $2M>2\sqrt{S}$ blocks. Συνεπώς για την περίπτωση που δημιουργούμε ταξινομημένα τρεξίματα για την σχέση S θα δημιουργηθούν λιγότερα από $\frac{|S|}{2\sqrt{|S|}}=\frac{1}{2}\sqrt{|S|}$ ξεχωριστά τρεξίματα της S. Και επειδή έχουμε θεωρήσει ότι $|S|\geq|R|$ τότε και για την σχέση R θα έχουμε λιγότερα από $\frac{1}{2}\sqrt{|S|}$ τρεξίματα.

Κατά την δεύτερη φάση φέρνουμε το πρώτο block από κάθε τρέξιμο της R αλλά και της S στην μνήμη. Στο σημείο αυτό είμαστε σε θέση πια να δημιουργήσουμε ταξινομημένα

τρεξίματα της R και της S με μήκη $|R|$ και $|S|$ αντίστοιχα. Οπότε καθώς μετατρέπονται οι σχέσεις R και S σε ταξινομημένες μπορούμε να εφαρμόσουμε την ζεύξη που επιθυμούμε και τα αποτελέσματα της να τα στέλνουμε στον δίσκο ή όπου αλλού θέλουμε.

Το συνολικό κόστος για την παραπάνω διαδικασία σε blocks είναι ίσο με $3|R|+3|S|$.

Στην περίπτωση που μια από τις δύο σχέσεις είναι ήδη ταξινομημένες τότε μπορούμε να εφαρμόσουμε πάλι την παραπάνω διαδικασία. Μόνο που τα blocks της μνήμης σε αυτή την περίπτωση πρέπει να είναι $|M|>\frac{\sqrt{|S|}}{2}$ (αν για παράδειγμα η σχέση S είναι η μη ταξινομημένη). Συνεπώς μετά το πέρας της πρώτης φάσης θα έχουμε ταξινομημένα τρεξίματα με μήκος $2|M|$ blocks όπου $2|M|>\sqrt{|S|}$. Στην δεύτερη φάση θα έχουμε λιγότερα από $\frac{S}{\sqrt{|S|}}=\sqrt{|S|}$ ταξινομημένα τρεξίματα. Αφού ο αριθμός των ταξινομημένων τρεξιμάτων θα είναι μικρότερος από $\sqrt{|S|}$ θα υπάρχει χώρος να φέρουμε το πρώτο μαζί με το πρώτο block του κάθε ταξινομημένου τρεξίματος και το πρώτο block της ταξινομημένης σχέσης R. Οπότε πια μπορούμε να έχουμε ταξινομημένη την S και να πραγματοποιήσουμε την ζεύξη που επιθυμούμε με την R. Το κόστος της παραπάνω διαδικασίας σε διάβασμα αλλά και γράψιμο blocks είναι ίσο με $3|S|+|R|$

Στην περίπτωση που η σχέση S είναι ταξινομημένη θα ακολουθηθεί ακριβώς η συμμετρική διαδικασία και το κόστος θα είναι ίσο με $|S|+|3R|$.

Συνοψίζοντας όλα τα παραπάνω έχουμε τα εξής όσον αφορά το κόστος σε blocks για διάβασμα και γράψιμο στην περίπτωση που χρησιμοποιήσουμε τον αλγόριθμο του [9] για να πραγματοποιήσουμε συγχώνευση σάρωση:

- Αν οι σχέσεις R και S δεν είναι ήδη ταξινομημένες και για την μνήμη ισχύει ότι $|M|>\sqrt{|S|}$ με $|S|\geq|R|$ τότε το κόστος είναι $3|R|+3|S|$ (7)
- Αν η σχέση R είναι ήδη ταξινομημένη ενώ η σχέση S δεν είναι και για την μνήμη ισχύει ότι $|M|>\frac{\sqrt{|S|}}{2}$ τότε το κόστος είναι $|R|+3|S|$ (8)

- Αν η σχέση S είναι ήδη ταξινομημένη ενώ η σχέση R δεν είναι και για την μνήμη ισχύει ότι $|M| > \frac{\sqrt{|R|}}{2}$ τότε το κόστος είναι $3|R| + |S|$ (9)

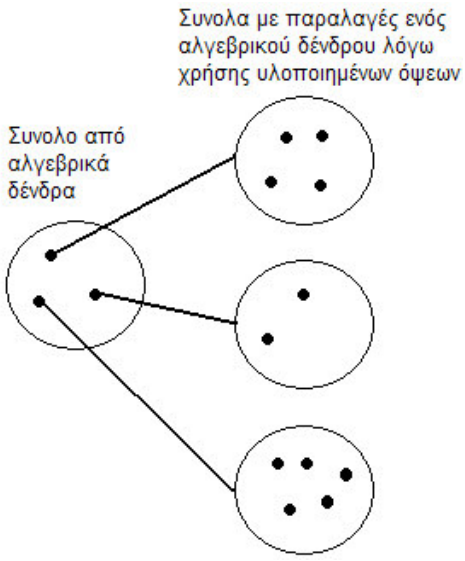
Το σημείο που δεν έχουμε εξετάσει στην συγκεκριμένη παράγραφο είναι αν μπορούμε να πραγματοποιήσουμε κάποιου είδους σωλήνωση με τα αποτελέσματα που μας έρχονται από τα πιο χαμηλά επίπεδα. Οι δύο εκδοχές του αλγορίθμου συγχώνευσης σάρωσης που αναλύθηκαν παραπάνω λαμβάνουν ως δεδομένο ότι οι σχέσεις που θα συμμετάσχουν στην ζεύξη (δηλαδή οι σχέσεις R και S στην συγκεκριμένη περίπτωση) είναι αποθηκευμένες στον δίσκο και θα πρέπει να διαβαστούν. Η ουσία είναι ότι είτε οι σχέσεις αυτές είναι ταξινομημένες είτε όχι μπορούν να χρησιμοποιηθούν άμεσα από τους παραπάνω αλγορίθμους χωρίς να χρειαστεί προηγούμενη αποθήκευσή τους πρώτα. Στην περίπτωση που μας έρχονται από κάποιον κόμβο Τ (κόμβος ανάγνωσης πίνακα) και είναι ήδη ταξινομημένες τότε σίγουρα μπορεί να χρησιμοποιηθούν άμεσα για την πραγματοποίηση της ζεύξης. Στην περίπτωση που έρχονται από οποιοδήποτε άλλο κόμβο ή δεν έχουμε καμία πληροφορία για το αν είναι ταξινομημένα πάλι μπορούν να χρησιμοποιηθούν άμεσα έτσι ώστε να παραληφθεί η πρώτη ανάγνωση που πραγματοποιείται κατά την φάση της ταξινόμησης.

2.2.4 View Matcher

2.2.4.1 Γενικό σκεπτικό

Το συγκεκριμένο αντικείμενο λαμβάνει ως είσοδο ένα σύνολο από βελτιστοποιημένα πλάνα και (τα οποία τα έχουμε λάβει προηγουμένως από τον ευριστικό βελτιστοποιητή) και αναλαμβάνει να βρει τις υλοποιημένες όψεις που μπορούν να χρησιμοποιηθούν έτσι ώστε να αντικαταστήσουν τμήματα του δένδρου (άρα και μέρος της επερώτησης). Αφού βρει τις συγκεκριμένες υλοποιημένες όψεις τις προσαρμόζει κατάλληλα στα σωστά σημεία των αντίστοιχων δένδρων. Για να επιτευχθεί η συγκεκριμένη προσαρμογή μπορεί να χρειαστεί να προστεθούν κάποιες πράξεις πάνω από έναν κόμβο υλοποιημένης όψης έτσι ώστε να περιοριστούν τα αρχικά αποτελέσματα της όψης σε αυτά που μας είναι πραγματικά

απαραίτητα. Τέλος επιστρέφει τροποποιημένα δένδρα με τους καλύτερους από τους διαθέσιμους συνδυασμούς τοποθέτησης υλοποιημένων όψεων για κάθε ένα από τα δένδρα του συνόλου που του δόθηκαν. Στην εικόνα που ακολουθεί μπορούμε να δούμε μακροσκοπικά την λειτουργία του view matcher.



Εικόνα 6: Λειτουργία view matcher

2.2.4.2 Πότε μια υλοποιημένη όψη είναι χρήσιμη

Πριν προχωρήσουμε στην ανάλυση της λειτουργίας του view matcher θα ασχοληθούμε πρώτα με το πότε μια υλοποιημένη όψη μπορεί να είναι χρήσιμη. Στο [10] και στο [11] εξετάζονται οι προϋποθέσεις για το πότε μια όψη μπορεί να χρησιμοποιηθεί και αν ναι πως μπορεί να πραγματοποιηθεί κάτι τέτοιο. Τα αποτελέσματα μιας υλοποιημένης όψης θα μας είναι χρήσιμα για την απάντηση μιας επερώτησης αν:

- Η όψη αλλά και η επερώτηση χρησιμοποιούν τους ίδιους πίνακες.
- Η όψη έχει τους ίδιους, λιγότερους ή πιο χαλαρούς περιορισμούς τιμής κάποιας στήλης ενός πίνακα με την επερώτηση. Δηλαδή σε καμία περίπτωση δεν πρέπει η

όψη να περιέχει έστω και έναν περιορισμό ο οποίος είναι πιο αυστηρός από τον αντίστοιχο της επερώτησης ή να μην υπάρχει στην επερώτηση.

- Η όψη πραγματοποιεί τις ίδιες ή λιγότερες ζεύξεις σε σχέση με την επερώτηση.

- Η όψη πραγματοποιεί προβολές σε όλες τις στήλες που προβάλει και η επερώτηση συν αυτές που πρέπει να χρησιμοποιηθούν για να της επιβληθούν εκ των υστέρων εξωτερικά αυστηρότεροι περιορισμοί ή ζεύξεις ώστε να συμβαδίζει με τους περιορισμούς της επερώτησης. Στην περίπτωση που προβάλει ακόμα περισσότερες στήλες τότε δεν μας πειράζει.

Το ιδανικό είναι η όψη να έχει ακριβώς τους ίδιους πίνακες, περιορισμούς, ζεύξεις αλλά και προβολές σε σχέση με την επερώτηση που μας ενδιαφέρει (και αν συμμετέχει και σε ζεύξη να είναι και προταξινομημένη με βάση μια στήλη της που χρησιμοποιείται στην ζεύξη αυτή). Αν οι πίνακες που συμμετέχουν στην όψη δεν είναι ακριβώς οι ίδιοι με την επερώτηση τότε η όψη απορρίπτεται. Υπάρχουν βέβαια και περιπτώσεις που ακόμα και αν η όψη έχει περισσότερους πίνακες από αυτούς της επερώτησης μπορεί να χρησιμοποιηθεί [12] αλλά δεν θα ασχοληθούμε εμείς με την περίπτωση αυτή στην συγκεκριμένη εργασία.

2.2.4.3 Πράξεις που πρέπει να εφαρμοστούν εκ των υστέρων σε μια υλοποιημένη όψη

Αν η υλοποιημένη όψη που διαλέξαμε δεν ταυτίζεται με την επερώτηση που θέλουμε να αντικαταστήσουμε τότε χρειάζεται να πραγματοποιήσουμε εξωτερικά κάποιες πράξεις στην όψη έτσι ώστε να είναι ισοδύναμη με την επερώτηση μας. Πιο συγκεκριμένα αν οι ζεύξεις της όψης είναι υποσύνολο αυτών της επερώτησης τότε πρέπει να προσθέσουμε εξωτερικά τις συγκεκριμένες ζεύξεις. Δηλαδή να προσθέσουμε έναν κόμβο S ακριβώς πάνω από τον κόμβο V της όψης με τις ζεύξεις που λείπουν. Ακριβώς το ίδιο θα πραγματοποιήσουμε αν οι περιορισμοί τιμής μιας στήλης που υπάρχουν σε μια επερώτηση είναι πιο αυστηροί από αυτούς που υπάρχουν στην όψη που μας ενδιαφέρει ή ακόμα περισσότερο αν δεν υπάρχουν. Τέλος πρέπει να βάλουμε πάνω από τους δύο κόμβους S που αναφέρθηκαν προηγουμένως και έναν κόμβο P έτσι ώστε να προβάλλουμε ακριβώς τις ίδιες στήλες με αυτές που προβάλει η επερώτηση και όχι παραπάνω.

2.2.4.4 Ποια τμήματα μιας επερώτησης μπορούν να αντικατασταθούν από μια υλοποιημένη όψη

Πέρα από την προφανή απάντηση που είναι όλη η επερώτηση αξίζει να δούμε αν μπορούμε να πραγματοποιήσουμε άλλες πιο μικρές αντικαταστάσεις. Σίγουρα το να βρεθεί υλοποιημένη όψη που να μπορεί να αντικαταστήσει όλη την επερώτηση είναι η ιδανική περίπτωση από άποψη κέρδους σε διάβασμα και γράψιμο αλλά οι πιθανότητες να πετύχουμε κάτι τέτοιο δεν είναι και οι μεγαλύτερες. Επόμενο βήμα είναι να δούμε αν μπορούμε να αντικαταστήσουμε τμήματα μιας επερώτησης. Για να το καταφέρουμε πρέπει να δούμε τα υποτμήματα μιας επερώτησης σαν μικρότερες επερωτήσεις. Στην συνέχεια μπορούμε να ψάξουμε για να δούμε αν μπορούμε να αντικαταστήσουμε τα υποτμήματα αυτά με υλοποιημένες όψεις. Η συγκεκριμένη πρακτική έχει ως αποτέλεσμα μεγαλύτερο αριθμό αναζητήσεων ανά επερώτηση αλλά μας δίνει και περισσότερες πιθανές λύσεις. Αν υπάρχει η βέλτιστη θα μας την επιστρέψει αλλά ακόμα και αν δεν υπάρχει θα μας επιστρέψει πλάνα που θα είναι σίγουρα καλύτερα από την εκτέλεση της απλής βελτιστοποιημένης επερώτησης χωρίς αντικαταστάσεις.

Τα σημεία που θα κοιτάξουμε για το αν μπορούν να αντικατασταθούν είναι τα εξής:

- Όλη η επερώτηση από το σημείο της ρίζας και πιο χαμηλά.
- Κάθε παιδί ενός κόμβου X. Ο συγκεκριμένος κόμβος μπορεί να συμβολίζει είτε ζεύξη αν έχει ως πατέρα κόμβο S που περιέχει ζεύξη είτε καρτεσιανό γινόμενο σε όλες τις άλλες περιπτώσεις.

Για παράδειγμα για την βελτιστοποιημένη επερώτηση που παρουσιάζεται στην εικόνα 5 θα πραγματοποιήσουμε πέντε αναζητήσεις. Μια αναζήτηση για την ρίζα και μια για κάθε παιδί κάθε κόμβου X που περιέχει το δένδρο.

2.2.4.5 Μείωση των αναζητήσεων για υλοποιημένες όψεις

Ακολουθώντας το σκεπτικό της παραπάνω παραγράφου καταλήγουμε στο ότι

πρέπει να πραγματοποιούμε για κάθε επερώτηση αρκετές παραπάνω αναζητήσεις για υλοποιημένες όψεις σε αντίθεση με την περίπτωση που επιθυμούσαμε να βρούμε μόνο μια γενική αντικατάσταση όλης της επερώτησης. Ο αριθμός των αναζητήσεων που θα πραγματοποιηθεί εξαρτάται άμεσα από τον αριθμό των ζεύξεων – καρτεσιανών γινομένων που έχει η επερώτησή μας. Πιο συγκεκριμένα αν έχουμε k ζεύξείς ή καρτεσιανά γινόμενα τότε ο αριθμός των αναζητήσεων που θα πρέπει να πραγματοποιήσουμε είναι $2k+1$. Δηλαδή σε μια σχέση με 2 ζεύξης πραγματοποιούμε 5 φορές περισσότερες αναζητήσεις. Σε αυτό το σημείο δεν πρέπει να ξεχνάμε ότι ο αριθμός των υλοποιημένων όψεων είναι μεγάλος οπότε κάθε τέτοια αναζήτηση κοστίζει και εμείς πρέπει πάση θυσία να αποφύγουμε τις περιττές αναζητήσεις.

Το σκεπτικό μείωσης των αναζητήσεων κρύβεται πίσω από μια απλή παρατήρηση. Στον view matcher δίνονται ένα σύνολο από βελτιστοποιημένα πλάνα τα οποία είναι ισοδύναμα μεταξύ τους και προήλθαν από παραλλαγές της ίδιας επερώτησης. Οι παραλλαγές αυτές διαφέρουν στην σειρά με την οποία πραγματοποιούν τις ζεύξεις και τα καρτεσιανά γινόμενα. Αυτό έχει ως αποτέλεσμα τα τελικά βελτιστοποιημένα πλάνα (αλγεβρικά δένδρα) να έχουν κοινά τμήματα μεταξύ τους (για παράδειγμα παιδιά Χ κόμβων). Αυτό που κάνουμε είναι για κάθε αναζήτηση που πραγματοποιείται μέσα στα πλαίσια του ίδιου συνόλου βελτιστοποιημένων επερωτήσεων να κρατάμε το αποτέλεσμά της (είτε μπορέσαμε να βρούμε όψη για να πραγματοποιήσουμε αντικατάσταση είτε όχι) οπότε σε επόμενη αναζήτηση να αποφύγουμε αναζητήσεις που έχουμε ήδη πραγματοποιήσει.

2.2.4.6 Δημιουργία νέων τροποποιημένων επερωτήσεων με χρήση των κατάλληλων υλοποιημένων όψεων.

Στο σημείο αυτό θεωρούμε ότι έχουμε πραγματοποιήσει μια αναζήτηση για υλοποιημένες όψεις για κάθε τμήμα της επερώτησης που είναι δυνατόν να αντικατασταθεί. Κάθε τέτοια αναζήτηση θα μας επιστρέψει ένα σύνολο με τις δυνατές όψεις που μπορούν να χρησιμοποιηθούν για αντικατάσταση του συγκεκριμένου τμήματος της επερώτησης. Οι συγκεκριμένες όψεις μπορούν να χρησιμοποιηθούν αλλά προφανώς μια είναι αυτή που θα μας δώσει τα καλύτερα αποτελέσματα.

Υπάρχουν δύο κριτήρια που θα μας οδηγήσουν στην επιλογή της πιο κατάλληλης όψης για ένα συγκεκριμένο σημείο.

1. Επιλογή της συνολικά πιο μικρής σε μέγεθος όψης.

2. Στην περίπτωση που η όψη λαμβάνει μέρος σε κάποια ζεύξη τότε για κάθε στήλη των πινάκων που λαμβάνουν μέρος στην ζεύξη κρατάμε την πιο μικρή όψη που είναι προταξινομημένη με βάση την συγκεκριμένη στήλη.

Επιλέγοντας την πιο μικρή όψη που είναι διαθέσιμη (και μπορεί να χρησιμοποιηθεί) εξασφαλίζουμε ότι για την συγκεκριμένη όψη το κόστος για να την διαβάσουμε θα είναι το πιο μικρό. Στην περίπτωση που δεν συμμετέχει σε ζεύξη είναι η ιδανική όψη. Ιδανική είναι και στην περίπτωση που θέλουμε να εφαρμόσουμε ζεύξη με βάση τον αλγόριθμο των εμφωλιασμένων βρόχων. Η μόνη περίπτωση που η συγκεκριμένη επιλογή μπορεί να μην είναι η βέλτιστη είναι αν έχουμε ζεύξη, χρησιμοποιηθεί ο αλγόριθμος συγχώνευσης σάρωσης, δεν είναι προταξινομημένη με βάση κάποια στήλη που συμμετέχει στην ζεύξη και υπάρχει μια άλλη όψη που είναι προταξινομημένη με βάση μια τέτοια στήλη. Σε μια τέτοια περίπτωση μπορεί η προταξινομημένη όψη να μην είναι η μικρότερη αλλά να αποτελεί την βέλτιστη επιλογή επειδή δεν θα υπάρχει το κόστος της αρχικής ταξινόμησης που θα χρειαστεί για την περίπτωση της πιο μικρής αλλά μη προταξινομημένης όψης.

Συνεπώς εφαρμόζοντας τα παραπάνω κριτήρια θα καταλήξουμε σε ένα σύνολο από δύο τρεις όψεις για κάποιο τμήμα της επερώτησης που μπορεί να αντικατασταθεί σε αντίθεση με το αρχικό σύνολο των είκοσι όψεων που μπορεί να είχαμε ως δυνατές αντικαταστάσεις για το συγκεκριμένο τμήμα. Αν αναλογιστούμε ότι μπορεί να είναι δυνατόν να αντικατασταθούν αρκετά διαφορετικά μέρη μιας επερώτησης από όψεις τότε είναι μεγάλο προτέρημα να ψάχνουμε την βέλτιστη λύση ανάμεσα σε συνδυασμούς διαφορετικών συνόλων μεγέθους δύο και διαφορετικών συνόλων μεγέθους είκοσι. Για να γίνει κατανοητή η διαφορά και στην πράξη αξίζει να αναφέρουμε ότι χωρίς την χρήση της παραπάνω βελτιστοποίησης, το πρόγραμμά μας για την ίδια βελτίωση της ίδιας επερώτησης μπορούσε να επεξεργαστεί το μέγιστο χίλιες με δύο χιλιάδες υλοποιημένες όψεις ενώ τώρα μπορεί να επεξεργαστεί τουλάχιστον πενήντα χιλιάδες υλοποιημένες όψεις.

Αφού καταλήξουμε με βάση το παραπάνω σκεπτικό στα κατάλληλα σύνολα όψεων

για αντικατάσταση συγκεκριμένων τμημάτων της επερώτησης πρέπει να βρούμε και όλους τους δυνατούς συνδυασμούς τοποθέτησης των συγκεκριμένων όψεων στην επερώτηση. Στο σημείο αυτό πρέπει να προσέξουμε επειδή η χρήση μιας όψης σε πιο υψηλό σημείο της επερώτησης μπορεί να αποκλείει την χρήση μιας άλλης όψης σε χαμηλότερο επίπεδο. Δηλαδή αν βρούμε μια όψη που μπορεί να αντικαταστήσει την ρίζα της επερώτησης της εικόνας 5 τότε προφανώς δεν μπορεί να συνδυαστεί με μια άλλη η οποία θα αντικαταστήσει το αριστερό παιδί ενός X κόμβου. Τους συγκεκριμένους περιορισμούς πρέπει να τους λάβουμε υπ' όψιν μας κατά την εύρεση των δυνατών συνδυασμών αλλά παρόλο που υπάρχουν περιορισμοί δεν παύει να προκύπτει ένας μεγάλος αριθμός τροποποιημένων επερωτήσεων για τις οποίες πρέπει στην συνέχεια να γίνει εκτίμηση κόστους.

2.2.5 Αντικείμενο το οποίο διαχειρίζεται τα στατιστικά της βάσης (database statistics manager)

2.2.5.1 Στόχος αντικειμένου

Το συγκεκριμένο αντικείμενο έχει ως στόχο να διαχειρίζεται και να προσφέρει στατιστικές πληροφορίες που αφορούν τα δεδομένα που είναι αποθηκευμένα στην βάση δεδομένων μας στα αντικείμενα που τις ζητάνε. Σύνδεση με βάση δεδομένων δεν υπάρχει στην εφαρμογή μας και τα συγκεκριμένα στατιστικά είναι τυχαία. Παρόλα αυτά έχουν λογική έτσι ώστε να μπορούμε να βασιστούμε και να βγάλουμε σωστά συμπεράσματα.

2.2.5.2 Δομή αρχείου που περιέχει στατιστικές πληροφορίες για τα δεδομένα της βάσης

Οι συγκεκριμένες πληροφορίες είναι αποθηκευμένες σε ένα αρχείο .xml έτσι ώστε να είναι πιο εύκολη η επεξεργασία τους. Στην συνέχεια ακολουθούν οι ετικέτες (tags) που

έχουν οι πληροφορίες αυτές στο xml αρχείο καθώς και μια περιγραφή γι' αυτές.

- <dbinfo> – Με την συγκεκριμένη ετικέτα αλλά και με την </dbinfo> σηματοδοτείται η έναρξη και η λήξη των πληροφοριών που είναι αποθηκευμένες στο αρχείο.
    - <sizeOfBlock> </sizeOfBlock> – Στο συγκεκριμένο σημείο αποθηκεύουμε το μέγεθος του block που έχει το σύστημά μας.
    - <sizeOfBuffer> </ sizeOfBuffer> – Στο συγκεκριμένο σημείο αποθηκεύουμε το μέγεθος του buffer που έχει το σύστημά μας. Προφανώς πρέπει να είναι μεγαλύτερο από το μέγεθος του block.
    - <table> – Η συγκεκριμένη ετικέτα σηματοδοτεί την έναρξη πληροφοριών για έναν συγκεκριμένο πίνακα. Οι πληροφορίες αυτές τελειώνουν με την ετικέτα </table>.
        - <tableName> <tableName> – Στο σημείο αυτό αποθηκεύουμε το όνομα του συγκεκριμένου πίνακα
        - <numberOfTuples> </numberOfTuples> – Στο σημείο αυτό αποθηκεύουμε τον αριθμό των πλειάδων που περιέχει ο πίνακας αυτός.
        - <sizeOfTuple> </sizeOfTuple> – Εδώ αποθηκεύουμε το μέγεθος που έχει μια γραμμή ενός πίνακα.
        - <columns> </columns> – Εδώ θα αποθηκεύσουμε τα ονόματα των στηλών που έχει ο πίνακας αυτός.
        - <collumnStatisticsNode> – Η συγκεκριμένη ετικέτα σηματοδοτεί την έναρξη πληροφοριών για μια στήλη του συγκεκριμένου πίνακα στον οποίο βρισκόμαστε την στιγμή αυτή. Οι πληροφορίες αυτές τελειώνουν με την ετικέτα </collumnStatisticsNode>.
            - <collumnName> </collumnName> – Στο συγκεκριμένο σημείο θα αποθηκεύσουμε το όνομα της συγκεκριμένη στήλης.

- <collumnSize> </collumnSize> – Στο συγκεκριμένο σημείο θα αποθηκεύσουμε πληροφορίες που αφορούν το μέγεθος μιας εγγραφής της στήλης αυτής.

- <numberOfDistinctValues> </numberOfDistinctValues> – Στο συγκεκριμένο σημείο θα αποθηκεύσουμε τον αριθμό των διαφορετικών τιμών που έχει συγκεκριμένη στήλη

- <minValue> </minValue> – Στο συγκεκριμένο σημείο θα αποθηκεύσουμε την ελάχιστη τιμή που μπορεί να έχει η συγκεκριμένη στήλη.

- <maxValue> </maxValue> – Στο συγκεκριμένο σημείο θα αποθηκεύσουμε την μέγιστη τιμή που μπορεί να έχει η συγκεκριμένη στήλη.

▪ <primaryKey> </primaryKey> – Στο συγκεκριμένο σημείο καταγράφουμε ποια στήλη του πίνακα είναι πρωτεύον κλειδί.

▪ <foreignKey> – Στο σημείο αυτό θα καταγράψουμε πληροφορίες για ξένο κλειδί προς έναν άλλο πίνακα. Οι πληροφορίες για το συγκεκριμένο ξένο κλειδί θα λήξουν όταν θα συναντήσουμε την ετικέτα </foreignKey>. Αν υπάρχει και άλλο ξένο κλειδί θα ακολουθήσει και νέα ετικέτα <foreignKey> μετά την </foreignKey>.

- <referencingTable> </referencingTable> – Εδώ καταγράφουμε το όνομα του πίνακα στον οποίο παραπέμπει το συγκεκριμένο ξένο κλειδί.

- <foreignKeyCollumn> </foreignKeyCollumn> – Εδώ καταγράφουμε το όνομα της στήλης που είναι ξένο κλειδί. Από σύμβαση θεωρούμε ότι το όνομα είναι κοινό και στο ξένο κλειδί αλλά και στο αντίστοιχο κλειδί στο οποίο παραπέμπει.

▪ <index> – Στο σημείο αυτό θα καταγράψουμε πληροφορίες για ένα ευρετήριο. Οι πληροφορίες αυτές θα λήξουν όταν συναντήσουμε την

ετικέτα </index>.

- <indexingCols> </indexingCols> – Εδώ αναφέρουμε τις στήλες του συγκεκριμένου πίνακα πάνω στις οποίες έχει δημιουργηθεί το συγκεκριμένο ευρετήριο.

- <accessCost> </accessCost> – Εδώ αναφέρουμε το κόστος προσπέλασης του ευρετηρίου. Είναι ουσιαστικά ο αριθμός σε blocks που πρέπει να προσπελάσει το ευρετήριο μέχρι να μας δώσει τον δείκτη προς το block που περιέχει τα δεδομένα που μας ενδιαφέρουν.

- <joiningCollumns> </joiningCollumns> - Εδώ αναφέρουμε ανάμεσα σε ποιες στήλες ποιων πινάκων μπορεί να πραγματοποιηθεί ζεύξη.

2.2.6 Αντικείμενο το οποίο διαχειρίζεται τις υλοποιημένες όψεις (view manager)

2.2.6.1 Στόχος αντικειμένου και τρόπος λειτουργίας

Το συγκεκριμένο αντικείμενο έχει ως στόχο να διαχειριστεί τις υλοποιημένες όψεις που υπάρχουν διαθέσιμες στο σύστημα μας. Οι περιγραφές των συγκεκριμένων όψεων είναι αποθηκευμένες σε αρχείο .xml το οποίο πρέπει να διαβαστεί σε πρώτη φάση. Όταν ολοκληρώνεται το διάβασμα της περιγραφής μιας υλοποιημένης όψης από το συγκεκριμένο αρχείο τότε επικοινωνεί με τον εκτιμητή κόστους έτσι ώστε να λάβει μια εκτίμηση για τον αριθμό αλλά και το μέγεθος των αποτελεσμάτων όψης αυτής. Στην συνέχεια με βάση τους πίνακες που συμμετέχουν στις συγκεκριμένες όψεις τις στέλνει στους αντίστοιχους κουβάδες κατακερματισμού χρησιμοποιώντας μια συνάρτηση κατακερματισμού. Αναζητήσεις πραγματοποιούνται πάλι μέσω της συνάρτησης κατακερματισμού και χρησιμοποιούνται ως κλειδί τα ονόματα των πινάκων που θέλουμε να λαμβάνουν μέρος στην όψη. Μαζί με την περιγραφή των υλοποιημένων όψεων επιστρέφεται και το μέγεθος των αποτελεσμάτων τους. Φυσικά, στοιχεία για το μέγεθος των αποτελεσμάτων της κάθε όψης θα μπορούσαν να μην είναι προϋπολογισμένα και

αποθηκευμένα και να υπολογίζονται την στιγμή που χρειάζονται από τον εκτιμητή κόστους. Κάτι τέτοιο όμως θα επιβάρυνε το έργο του κατά την φάση εκτίμησης των ισοδύναμων εκφράσεων μιας επερώτησης και επιλέξαμε το συγκεκριμένο έργο να πραγματοποιηθεί για όλες τις όψεις πριν ληφθεί μια επερώτηση για επεξεργασία.

2.2.6.2 Δομή αρχείου που περιέχει περιγραφές για τις υλοποιημένες όψεις

Και το συγκεκριμένο αρχείο είναι ένα xml αρχείο έτσι ώστε να είναι πιο εύκολη η επεξεργασία του και η ανάγνωσή του. Ένα παράδειγμα ενός τέτοιου αρχείου που περιέχει μόνο μια υλοποιημένη όψη είναι το ακόλουθο.

```
<viewinfo>
      <view>
            <tables>
                  BOOKS
                  BOR
                  LOANS
                  PUB
            </tables>
            <joins>
                  BOOKS.PNAME e PUB.PNAME
                  BOOKS.LC_NO e LOANS.LC_NO
                  BOR.CARD_NO e LOANS.CARD_NO
            </joins>
            <constrains>
                  BOOKS.TITLE g 852
                  BOOKS.PNAME g 329
                  PUB.PCITY l 264
            </constrains>
            <projections>
                  BOR.ADDR
                  BOR.NAME
                  BOR.CITY
                  LOANS.L_ID
                  LOANS.DATE
            </projections>
      </view>
</viewinfo>
```

Πίνακας 2: Δείγμα αρχείου που περιέχει περιγραφές υλοποιημένων όψεων

Στην συνέχεια θα παρουσιάσουμε τις ετικέτες που εμφανίζονται σε ένα τέτοιο αρχείο μαζί με πληροφορίες για τα δεδομένα που συνδέονται με τις ετικέτες αυτές.

- <viewinfo> </viewinfo> – Με τις συγκεκριμένες ετικέτες σηματοδοτείται η έναρξη και η λήξη του αρχείου που περιλαμβάνει τις περιγραφές για τις υλοποιημένες όψεις.
    - <view> </view> – Με τις συγκεκριμένες ετικέτες σηματοδοτείται η έναρξη και η λήξη της περιγραφής μιας όψης.
        - <tables> και </tables> – Ανάμεσα στις ετικέτες αυτές αναφέρουμε τους πίνακες πάνω στους οποίους βασίζεται η όψη.
        - <joins> </joins> – Ανάμεσα στις ετικέτες αυτές καταγράφουμε τις ζεύξεις της όψης.
        - <constrains> </constrains> – Ανάμεσα στις ετικέτες αυτές καταγράφουμε περιορισμούς τιμής στηλών ορισμένων πινάκων με βάση κάποιες σταθερές.
        - <projections> </projections> – Ανάμεσα στις ετικέτες αυτές καταγράφουμε τις προβολές που πραγματοποιεί η όψη μας.

Το σημείο που πρέπει να προσέξουμε (σε περίπτωση που θέλουμε να επεξεργαστούμε το συγκεκριμένο αρχείο) είναι ότι αντί για το σύμβολο της ισότητας ‘=’ ανάμεσα στις δύο στήλες που συμμετέχουν στην ζεύξη υπάρχει το γράμμα ‘e’. Τα ίδια ισχύουν και στην περίπτωση των περιορισμών τιμής αν θέλουμε να έχουμε ισότητα. Στην περίπτωση που θέλουμε να δηλώσουμε ανισότητα και πιο συγκεκριμένα θέλουμε να γράψουμε ‘<’ τότε θα τοποθετήσουμε το γράμμα ‘l’ ενώ αν θέλουμε να γράψουμε ‘>’ τότε θα τοποθετήσουμε το γράμμα ‘g’.

2.3 Περιβάλλον γραφικής παρουσίασης αποτελεσμάτων βελτιστοποιητή

2.3.1 Στόχος εφαρμογής

Η συγκεκριμένη γραφική εφαρμογή σχεδιάστηκε έτσι ώστε να μπορούμε να παρατηρούμε γραφικά τα αποτελέσματα του βελτιστοποιητή και στόχος της είναι να μας βοηθήσει να τα κατανοήσουμε καλύτερα. Ο βελτιστοποιητής κάνει έναν μεγάλο αριθμό από υπολογισμούς αλλά και τροποποιήσεις στην αρχική μας επερώτηση οι οποίοι αν δεν αναπαρασταθούν γραφικά θα είναι δύσκολο να γίνουν κατανοητοί. Για τον λόγο αυτό δίνονται στη συγκεκριμένη εφαρμογή αλγεβρικά δένδρα από διαφορετικά στάδια της βελτιστοποίησης και αυτή αναλαμβάνει να τα σχεδιάσει. Ακόμα αναγράφει στα συγκεκριμένα δένδρα και το κόστος που έχει υπολογίσει ο εκτιμητής κόστους για κάθε κόμβο ενός δένδρου

2.3.2 Αρχείο εισόδου

Σαν είσοδο η γραφική εφαρμογή λαμβάνει σύνολα από τριάδες αλγεβρικών δένδρων και αναλαμβάνει να τα απεικονίσει ανά τρία κάθε φορά. Μια τριάδα αναφέρεται στο ίδιο δένδρο το οποίο καταγράφεται αρχικά ημι-βελτιστοποιημένο, στην συνέχεια πλήρως βελτιστοποιημένο και τέλος με χρήση υλοποιημένων όψεων. Στην περίπτωση που δεν υπάρχουν υλοποιημένες όψεις για να γίνουν αντικαταστάσεις τμημάτων του πλήρως βελτιστοποιημένου δένδρου η τρίτη θέση παραμένει κενή. Αυτό που πρέπει να τονίσουμε πριν δούμε την δομή του αρχείου είναι ότι όλες οι τριάδες αφορούν την ίδια επερώτηση και είναι αυτή που δώσαμε στον βελτιστοποιητή. Στον επόμενο πίνακα παρουσιάζεται ένα πολύ απλό αρχείο εισόδου του το οποίο περιέχει μια μόνο τριάδα από αλγεβρικά δένδρα.

```
<-- Next -->
<-- Partially Optimized Plan -->
P[BOOKS.TITLE|(882106,10,674)](S[BOOKS.LC_NO =
LOANS.LC_NO|(882106,136,674)|Nested Loops Join with Index on table
BOOKS]((S[BOR.CARD_NO = LOANS.CARD_NO|(758528,102,485)|Simple Merge
Join]((T[BOR|(64000,80,800)]) X[(0,102,388000)] (S[LOANS.DATE <
"500"|(17664,22,485)](T[LOANS|(17664,22,800)]))))) X[(0,136,485000)]
(T[BOOKS|(34048,34,1000)])))
<-- Fully Optimized Plan -->
P[BOOKS.TITLE|(172986,10,674)](S[BOOKS.LC_NO =
LOANS.LC_NO|(172986,18,674)|Nested Loops Join with Index on table
BOOKS]((P[LOANS.LC_NO|(96512,4,485)](S[BOR.CARD_NO =
LOANS.CARD_NO|(96512,12,485)|Sapiro Merge
Join]((P[BOR.CARD_NO|(64000,4,800)](T[BOR|(64000,80,800)])) X[(0,12,388000)]
(P[LOANS.CARD_NO , LOANS.LC_NO|(17664,8,485)](S[LOANS.DATE <
"500"|(17664,22,485)](T[LOANS|(17664,22,800)])))))) X[(0,18,485000)]
(P[BOOKS.LC_NO , BOOKS.TITLE|(34048,14,1000)](T[BOOKS|(34048,34,1000)])))))
<-- Fully Optimized Plan With Views -->
P[BOOKS.TITLE|(26880,10,674)](S[LOANS.DATE <
"500"|(26880,24,674)](V[P(BOOKS.TITLE , LOANS.CARD_NO , LOANS.DATE) C()
J(BOR.CARD_NO = LOANS.CARD_NO , BOOKS.LC_NO = LOANS.LC_NO) T(BOOKS , BOR ,
LOANS)|(26880,24,1112)]))
```

Πίνακας 3: Δείγμα αρχείου εισόδου του περιβάλλοντος γραφικής παρουσίασης των αποτελεσμάτων του βελτιστοποιητή

Το συγκεκριμένο αρχείο δεν είναι δομημένο σε xml μορφή μια που είναι αρχείο για εσωτερική χρήση ανάμεσα στο βελτιστοποιητή και το περιβάλλον γραφικής παρουσίασης και όχι για να το επεξεργάζεται ο χρήστης. Παρόλα αυτά τα δεδομένα του μπορούν να γίνουν εύκολα κατανοητά από κάποιον τρίτο που θέλει να τα διαβάσει.

Στον παραπάνω πίνακα όταν αρχίζει μια νέα τριάδα από πλάνα τότε αυτό δηλώνεται με την ετικέτα '`<-- Next -->`'. Αμέσως μετά ακολουθεί η ετικέτα ΄`<-- Partially Optimized Plan -->`΄ η οποία δηλώνει ότι στην επόμενη γραμμή ακολουθεί μια έκφραση σχεσιακής άλγεβρας η οποία αντιστοιχεί σε ένα μη πλήρως βελτιστοποιημένο αλγεβρικό δένδρο. Μετά την συγκεκριμένη σχέση θα ακολουθήσει στην επόμενη γραμμή η ετικέτα '`<-- Fully Optimized Plan -->`' που μας δηλώνει ότι στην αμέσως επόμενη γραμμή ακολουθεί έκφραση σχεσιακής άλγεβρας του πλήρως βελτιστοποιημένου πλάνου εκτέλεσης της επερώτησης που είχαμε δώσει στον βελτιστοποιητή. Κλείνοντας τελευταία ετικέτα είναι η '`<-- Fully Optimized Plan With Views -->`' που μας πληροφορεί ότι στην γραμμή που ακολουθεί θα βρούμε έκφραση σχεσιακής άλγεβρας που αντιστοιχεί σε πλάνο βελτιστοποιημένου δένδρου με χρήση όψεων. Στην περίπτωση που δεν υπάρχουν όψεις τέτοιες ώστε να μπορούν να χρησιμοποιηθούν για την κατασκευή ενός τέτοιου δένδρου η

συγκεκριμένη γραμμή θα παραμείνει κενή.

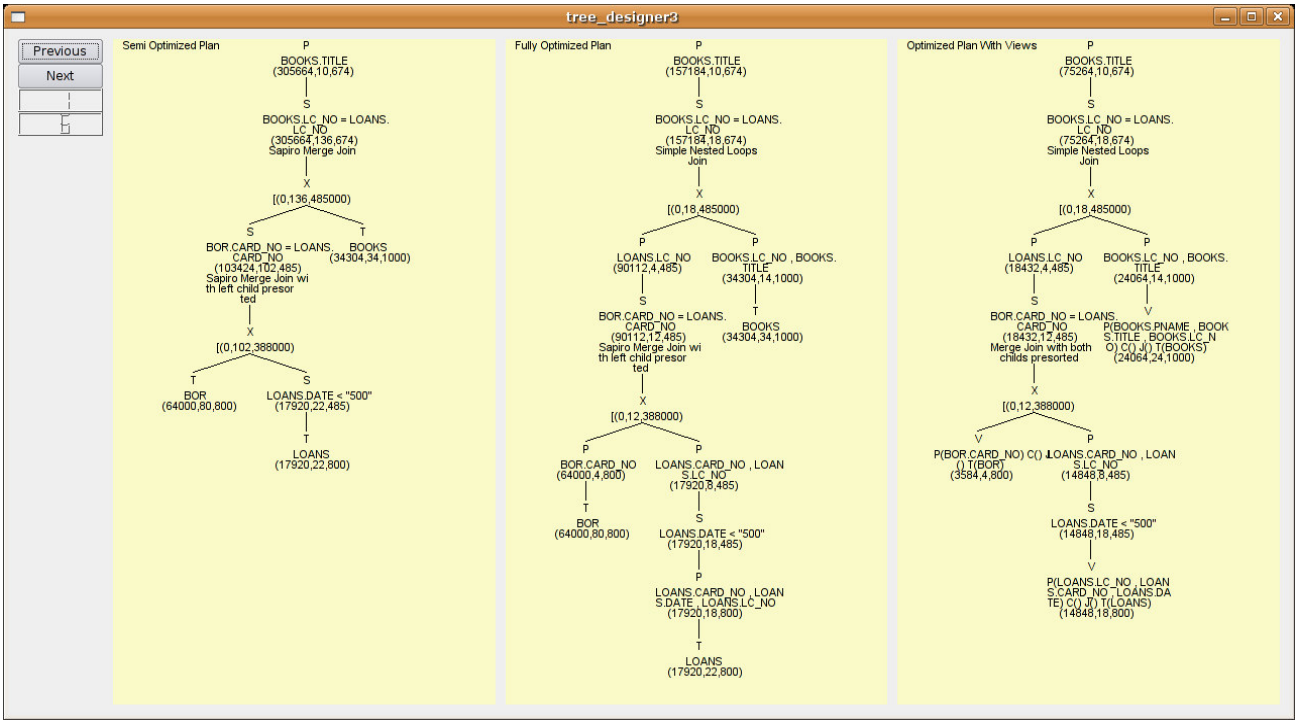
Ολοκληρώνοντας με την επεξήγηση του συγκεκριμένου αρχείου εισόδου αξίζει να προσθέσουμε κάτι ακόμα. Στις σχεσιακές εκφράσεις που αναφέρθηκαν παραπάνω υπάρχουν προβολές, επιλογές, καρτεσιανά γινόμενα, πίνακες, όψεις κλπ. Οι συγκεκριμένες πράξεις αλλά και αντικείμενα συνοδεύονται από μια τριάδα αριθμών που συμβολίζουν:

- το συνολικό κόστος (σε bytes) που πρέπει να δαπανηθεί για να φτάσουμε στο συγκεκριμένο σημείο.
- το μήκος της γραμμής (σε bytes) του αποτελέσματος μέχρι εκείνο το σημείο.
- τον αριθμό των γραμμών του αποτελέσματος στο σημείο εκείνο.

Δεν πρέπει να ξεχνάμε ότι λόγω σύμβασης, μια πράξη καρτεσιανού γινομένου την θεωρούμε ζεύξη αν συνοδεύεται (έχει σαν πατέρα της στο σχεσιακό δένδρο) από κόμβο επιλογής που περιέχει ζεύξη. Στην συγκεκριμένη περίπτωση το συνολικό κόστος μέχρι το σημείο αυτό θα αναγραφεί στην επιλογή ενώ θα υπάρχει και σχόλιο σχετικά με τον αλγόριθμο ζεύξης που ακολουθήθηκε.

2.3.3 Εμφάνιση αποτελεσμάτων

Αφού η εφαρμογή διαβάσει το αρχείο εισόδου θα παρουσιάσει στην συνέχεια γραφικά τα αποτελέσματα τα οποία αντιστοιχούν στο συγκεκριμένο αρχείο. Η οθόνη που θα μας εμφανιστεί είναι η παρακάτω.



Εικόνα 7: Δείγμα γραφικής αναπαράστασης των αποτελεσμάτων του βελτιστοποιητή

Στο αριστερό τμήμα του παραθύρου υπάρχουν 2 κουμπιά που μας παραπέμπουν στην επόμενη και προηγούμενη τριάδα πλάνων που είναι διαθέσιμη (Next και Previous). Ακριβώς κάτω από τα συγκεκριμένα πλάνα υπάρχουν δύο αριθμοί. Ο πρώτος είναι ο αριθμός της τριάδας που παρατηρούμε την συγκεκριμένη στιγμή ενώ ο δεύτερος είναι ο συνολικός αριθμός των τριάδων που φορτώθηκαν από την εφαρμογή.

Το κεντρικό παράθυρο της εφαρμογής χωρίζεται σε τρία τμήματα. Στο πρώτο παρουσιάζεται ένα από τα μη πλήρη βελτιστοποιημένα δένδρα που προέκυψαν από την επερώτησή μας ενώ στο δεύτερο το αντίστοιχο πλήρως βελτιστοποιημένο δένδρο. Στο τρίτο τμήμα εμφανίζεται πλάνο μόνο στην περίπτωση που υπάρχουν όψεις έτσι ώστε να μπορούν να χρησιμοποιηθούν για την αντικατάσταση τμημάτων του πλήρως βελτιστοποιημένου δένδρου.

2.3.4 Επεξήγηση αποτελεσμάτων

Στο συγκεκριμένο σημείο είναι καλό να ασχοληθούμε με τα αποτελέσματα που

παρουσιάζονται από την συγκεκριμένη εφαρμογή μια που θα μας βοηθήσει να κατανοήσουμε καλύτερα την λειτουργικότητα των πιο σημαντικών αντικειμένων με τα οποία συνεργάζεται ο βελτιστοποιητής.

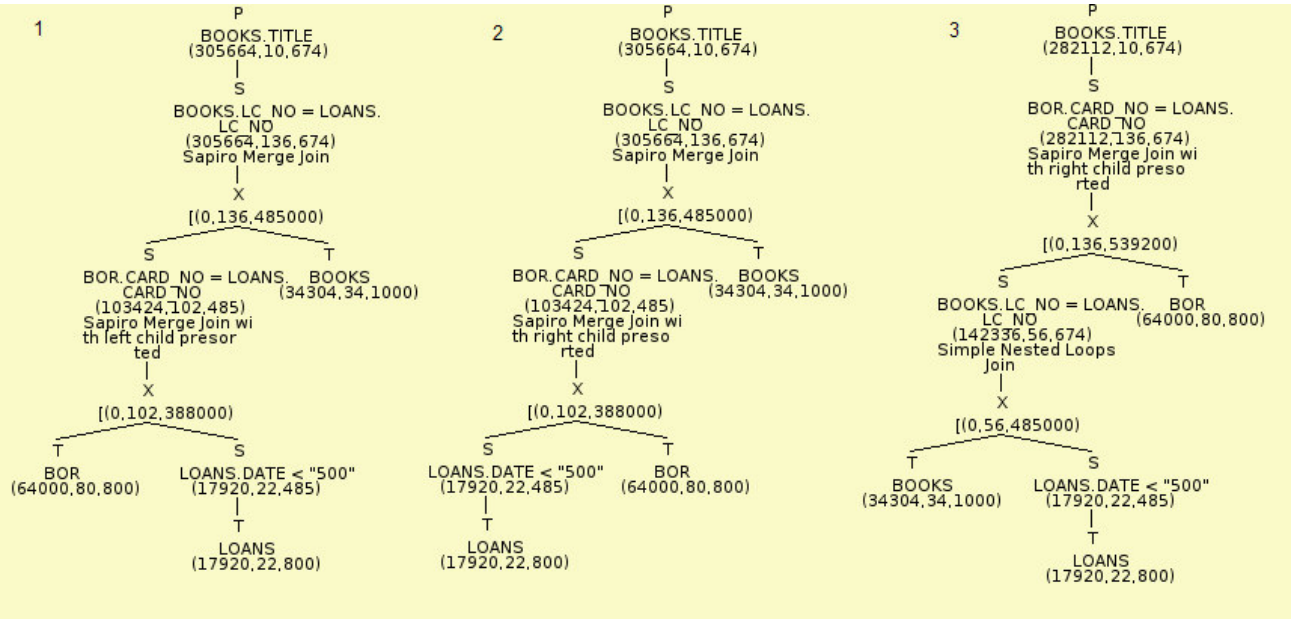
Ας εξετάσουμε την περίπτωση που έχουμε ζητήσει από το κεντρικό πρόγραμμα να μας βελτιστοποιήσει την επερώτηση η οποία παρουσιάζεται στον πίνακα 1. Ακόμα του ζητάμε να μας δώσει τα αποτελέσματα στην μορφή του πίνακα 3 έτσι ώστε να μπορέσουμε να τα χρησιμοποιήσουμε ως είσοδο στο συγκεκριμένο περιβάλλον αναπαράστασης. Τέλος ζητάμε για κάθε βελτιστοποιημένη παραλλαγή της αρχικής επερώτησης να έχουμε μόνο το βέλτιστο ισοδύναμο πλάνο που χρησιμοποιεί υλοποιημένες όψεις. Μπορούμε βέβαια να ζητήσουμε να μας επιστρέφεται ένα σύνολο από καλά πλάνα που χρησιμοποιούν υλοποιημένες όψεις και αντιστοιχούν σε μια βελτιστοποιημένη παραλλαγή της αρχικής μας επερώτησης και όχι μόνο το βέλτιστο ανά περίπτωση αλλά όπως καταλαβαίνουμε έτσι θα αυξηθεί σημαντικά το μέγεθος των αποτελεσμάτων.

Μια παραλλαγή της επερώτησης του πίνακα 1 η οποία είναι μη βελτιστοποιημένη παρουσιάζεται στην εικόνα 2. Στην εικόνα 7 μπορούμε να παρατηρήσουμε το ημι-βελτιστοποιημένο πλάνο της, το πλήρες βελτιστοποιημένο πλάνο της καθώς και την πιο οικονομική παραλλαγή του βελτιστοποιημένου πλάνου με χρήση υλοποιημένων όψεων.

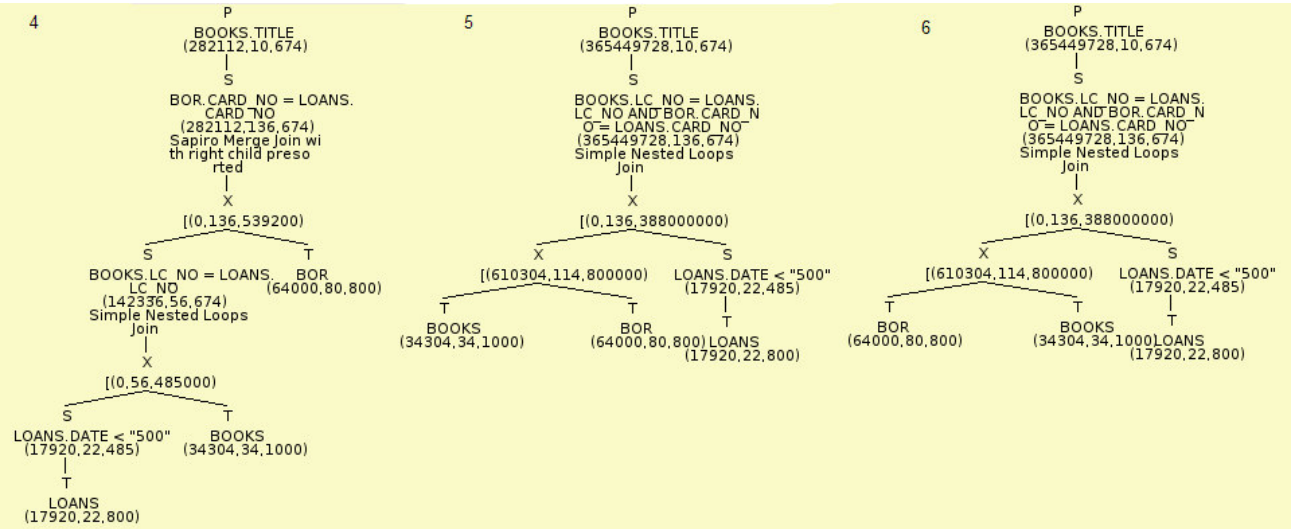
Η παραλλαγή που βλέπουμε στην εικόνα 2 είναι μια από τις 6 συνολικά παραλλαγές της συγκεκριμένης επερώτησης. Όλες οι παραλλαγές είναι αριστεροβαθιά δένδρα, μια που αυτά είναι πιο καλά από θέμα βελτιστοποίησης και διαφέρουν μόνο στην διάταξη των πινάκων που συμμετέχουν στα καρτεσιανά γινόμενα. Όλες οι δυνατές διατάξεις n πινάκων ανά n είναι $n!$ και από εκεί προκύπτει και ότι έχουμε 6 παραλλαγές της συγκεκριμένης επερώτησης. Τα συγκεκριμένα μη βελτιστοποιημένα πλάνα δεν παρουσιάζονται στην εφαρμογή μας και δεν υπολογίζεται κόστος εκτέλεσης γι’ αυτά γιατί όπως έχουμε αναφέρει είναι πολύ ακριβά. Αντίθετα χρησιμοποιούμε τα αντίστοιχα ημι-βελτιστοποιημένα πλάνα τα οποία παρουσιάζονται στις εικόνες που ακολουθούν.

Παρατηρώντας τα συγκεκριμένα πλάνα μπορούμε να διαπιστώσουμε ότι αυτά που περιλαμβάνουν μόνο ζεύξεις και όχι πράξεις καρτεσιανού γινομένου είναι πολύ πιο φθηνά από αυτά που περιέχουν έστω και μια πράξη καρτεσιανού γινομένου. Ακόμα παρόλο που έχουν βελτιστοποιηθεί σε έναν βαθμό μπορούμε να παρατηρήσουμε ότι η βασική διαφορά

τους είναι η διάταξη των πινάκων που συμμετέχουν. Όλες οι υπόλοιπες διαφορές που μπορεί να παρατηρήσουμε (για παράδειγμα θέση των κόμβων S) είναι άμεσα συσχετισμένες με τις διαφορές στις θέσεις των πινάκων.

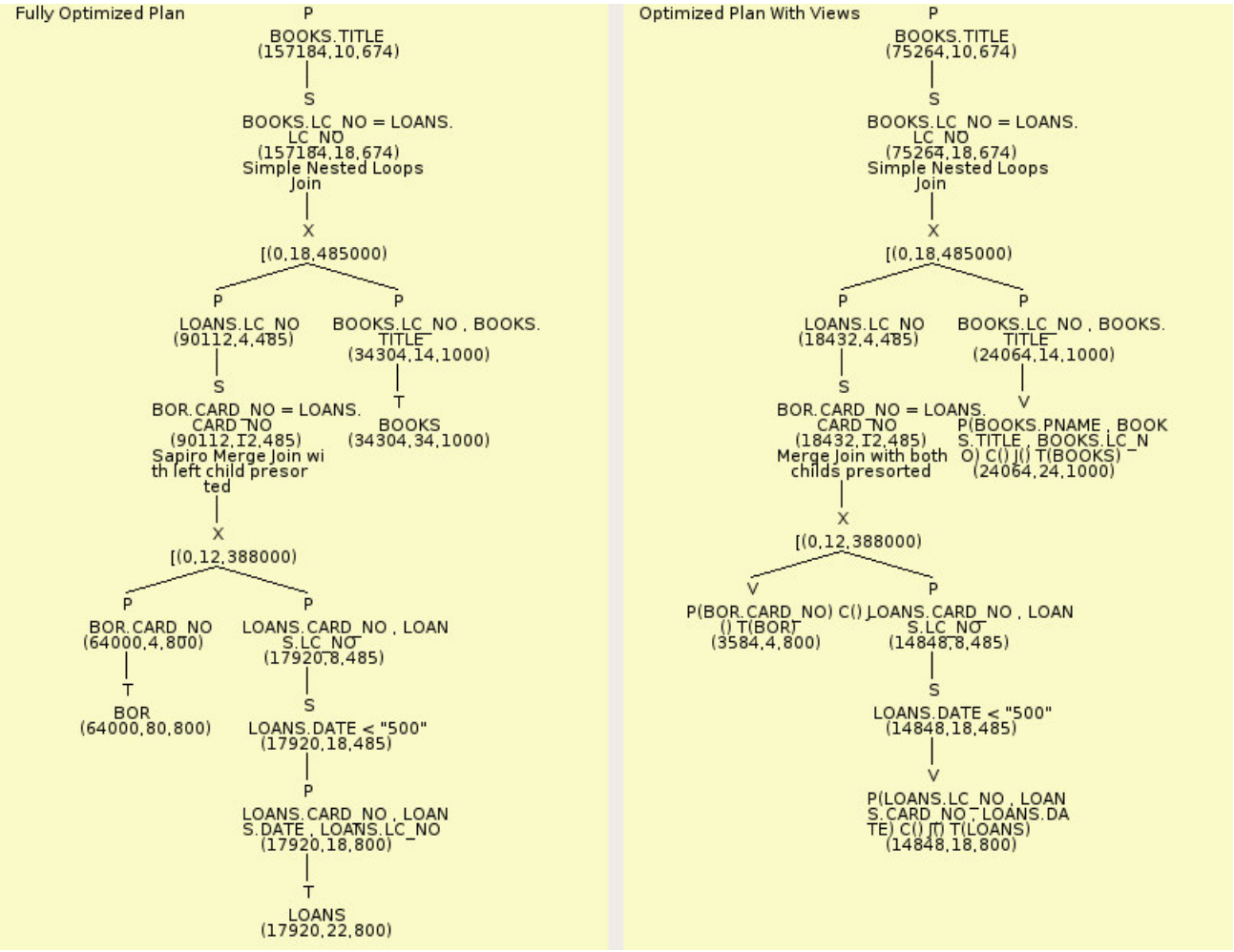


Εικόνα 8: Πρώτο σύνολο από μη πλήρως βελτιστοποιημένες παραλλαγές της επερώτησης του πίνακα 1



Εικόνα 9: Δεύτερο σύνολο από μη πλήρως βελτιστοποιημένες παραλλαγές της επερώτησης του πίνακα 1

Στην συνέχεια θα ασχοληθούμε με το πλήρως βελτιστοποιημένο πλάνο αλλά και το βέλτιστο πλάνο με χρήση όψεων που προκύπτουν από το πρώτο ημι-βελτιστοποιημένο πλάνο της εικόνας 8. Τα συγκεκριμένα αυτά πλάνα παρουσιάζονται στην εικόνα που ακολουθεί.



Εικόνα 10: Πλήρως βελτιστοποιημενο πλάνο και πλήρως βελτιστοποιημενο πλάνο με χρήση υλοποιημένων όψεων της παραλλαγής 1 της εικόνας 8

Αν συγκρίνουμε το ημι-βελτιστοποιημένο πλάνο με το αντίστοιχο βελτιστοποιημένο της παραπάνω εικόνας θα δούμε ότι διαφέρουν στον αριθμό των προβολών που πραγματοποιούνται. Ουσιαστικά οι προβολές προέκυψαν εφαρμόζοντας τα βήματα 4 και 5 του αλγόριθμου ευριστικής βελτιστοποίησης. Σε κάθε κόμβο των δένδρων αυτών

αναγράφονται 3 αριθμοί οι οποίοι είναι:

- το κόστος σε διάβασμα και γράψιμο μέχρι και εκείνο το σημείο (σε bytes)
- το μέγεθος την γραμμής του (ενδιάμεσου) αποτελέσματος σε εκείνο το σημείο (σε bytes)
- καθώς και ο αριθμός των γραμμών του (ενδιάμεσου) αποτελέσματος σε εκείνο το σημείο

και εμφανίζονται με την συγκεκριμένη σειρά που αναφέρθησαν. Μπορούμε να παρατηρήσουμε ότι η χρήση των προβολών στο πλήρες βελτιστοποιημένο δένδρο έχει ως αποτέλεσμα να φτάνουν μόνο τα χρήσιμα αποτελέσματα στους αλγορίθμους ζεύξης και το κέρδος από αυτό αντικατοπτρίζεται στο κόστος των συγκεκριμένων αλγορίθμων αλλά φυσικά και στο συνολικό κόστος υπολογισμού του αντίστοιχου πλάνου. Αν συγκρίνουμε το συνολικό κόστος του ημι-βελτιστοποιημένου με το πλήρως βελτιστοποιημένο πλάνο θα παρατηρήσουμε ότι υπάρχει μια βελτίωση της τάξης του 49% για την πρώτη παραλλαγή της αρχικής επερώτησης.

Στην εικόνα 10 παρουσιάζεται η πρώτη παραλλαγή της επερώτησης βελτιστοποιημένη ευριστικά καθώς και η καλύτερη, από άποψη κόστους, εκδοχή της με χρήση υλοποιημένων όψεων. Μπορούμε να παρατηρήσουμε ότι η πιο αριστερή όψη μπορεί να χρησιμοποιηθεί άμεσα μια που προβάλει τις στήλες που χρειαζόμαστε δεν συμβαίνει όμως το ίδιο και με τις άλλες δύο όψεις. Στην επόμενη όψη που συναντάμε, πηγαίνοντας από τα αριστερά προς τα δεξιά, πρέπει να περιορίσουμε και τον αριθμό των γραμμών εφαρμόζοντας τον περιορισμό LOANS.DATE < 500 αλλά και το μέγεθος των γραμμών προβάλλοντας τις κατάλληλες στήλες. Τέλος για την τρίτη όψη το μόνο που πρέπει να κάνουμε είναι να περιορίσουμε το μέγεθος των γραμμών εφαρμόζοντας τον κατάλληλο περιορισμό. Η βελτίωση του που παρατηρείται στην συγκεκριμένη περίπτωση λόγω χρήσης όψεων σε σχέση με το πλήρες βελτιστοποιημένο πλάνο είναι 52%. Πρέπει να σημειώσουμε στο σημείο αυτό ότι η συγκεκριμένη βελτίωση αφορά μόνο την συγκεκριμένη παραλλαγή (δηλαδή την πρώτη) της επερώτησης που εξετάσαμε. Βελτιώσεις ανάμεσα στο βέλτιστο βελτιστοποιημένο πλάνο και στο βέλτιστο πλάνο με χρήση υλοποιημένων όψεων θα παρουσιαστούν στην ενότητα των μετρήσεων.

2.4 Πρόγραμμα μαζικής δημιουργίας όψεων

Στόχος του συγκεκριμένου προγράμματος είναι η δημιουργία μεγάλου αριθμού από περιγραφές όψεων έτσι ώστε να μπορέσουμε να τις χρησιμοποιήσουμε στο πρόγραμμα του βελτιστοποιητή και να πραγματοποιήσουμε τα πειράματα που μας ενδιαφέρουν. Για να δημιουργηθούν οι συγκεκριμένες όψεις πρέπει να έχουμε και στατιστικά στοιχεία που αφορούν τα δεδομένα που είναι αποθηκευμένα στην βάση δεδομένων του συστήματός μας. Τα συγκεκριμένα στατιστικά στοιχεία πρέπει να είναι τα ίδια με αυτά που θα χρησιμοποιήσει ο βελτιστοποιητής στην συνέχεια για να βελτιστοποιήσει κάποια επερώτησή μας.

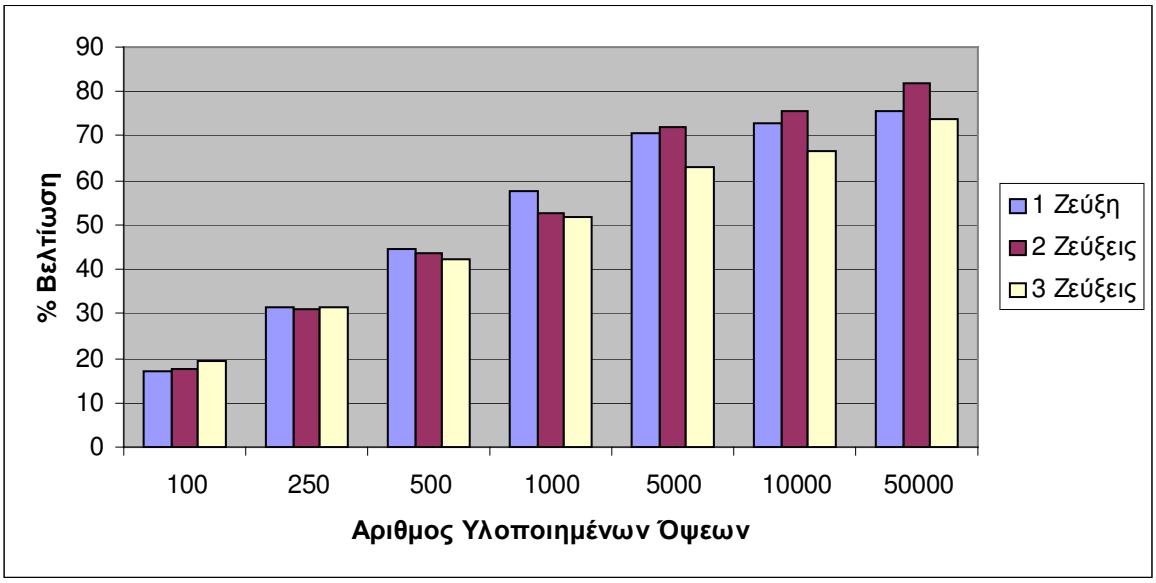
Ουσιαστικά αυτό που κάνει το συγκεκριμένο πρόγραμμα είναι να διαβάζει το αρχείο με τα στατιστικά που αφορούν τα δεδομένα και μέχρι να δημιουργήσει τον αριθμό των όψεων που έχουμε αποφασίσει να κάνει τα παρακάτω:

- να διαλέγει τυχαία ποιοι πίνακες από αυτούς που είναι διαθέσιμοι θα συμμετέχουν στην όψη αυτή.
- να βρίσκει σε ποιους πίνακες από αυτούς που διάλεξε μπορεί να δημιουργήσει ζεύξεις και ανάμεσα σε ποίες στήλες των πινάκων αυτών.
- να διαλέξει ανάμεσα στις ζεύξεις που μπορεί να πραγματοποιήσει κάποιες έτσι ώστε να πραγματοποιήσει ζεύξεις ανάμεσα σε όλους τους πίνακες που έχει διαλέξει προηγουμένως.
- να διαλέγει και να δημιουργεί τυχαία περιορισμούς πάνω στις στήλες των πινάκων που έχει διαλέξει.
- να αποφασίζει πάνω σε ποιες από τις διαθέσιμες στήλες θα πραγματοποιήσει προβολή.
- να αποφασίσει αν η όψη θα είναι ήδη ταξινομημένη με βάση κάποια στήλη της και αν ναι τότε να διαλέξει μια τέτοια στήλη που θα είναι μία από τις στήλες της προβολής.

Ακόμα το συγκεκριμένο πρόγραμμα μας δίνει την δυνατότητα να προωθήσουμε την δημιουργία περιγραφών όψεων που βασίζονται σε έναν συγκεκριμένο αριθμό από πίνακες έναντι των υπολοίπων όψεων. Πιο συγκεκριμένα μπορούμε να ζητήσουμε για παράδειγμα ότι θέλουμε οι όψεις που χρησιμοποιούν δύο πίνακες (και υπάρχει ζεύξη ανάμεσα στους πίνακες αυτούς) να καταλαμβάνουν το 80% των συνολικών όψεων. Το ίδιο μπορούμε να κάνουμε για οποιονδήποτε αριθμό πινάκων (αρκεί να υπάρχει αυτός ο αριθμός και να μπορεί να δημιουργηθεί ζεύξη ανάμεσα σε αυτούς) και για οποιαδήποτε ποσοστό επιθυμούμε. Η συγκεκριμένη ιδιότητα θα μας φανεί πολύ χρήσιμη στην πραγματοποίηση των πειραμάτων μας όπως θα δούμε και στην συνέχεια.

3. Πειράματα

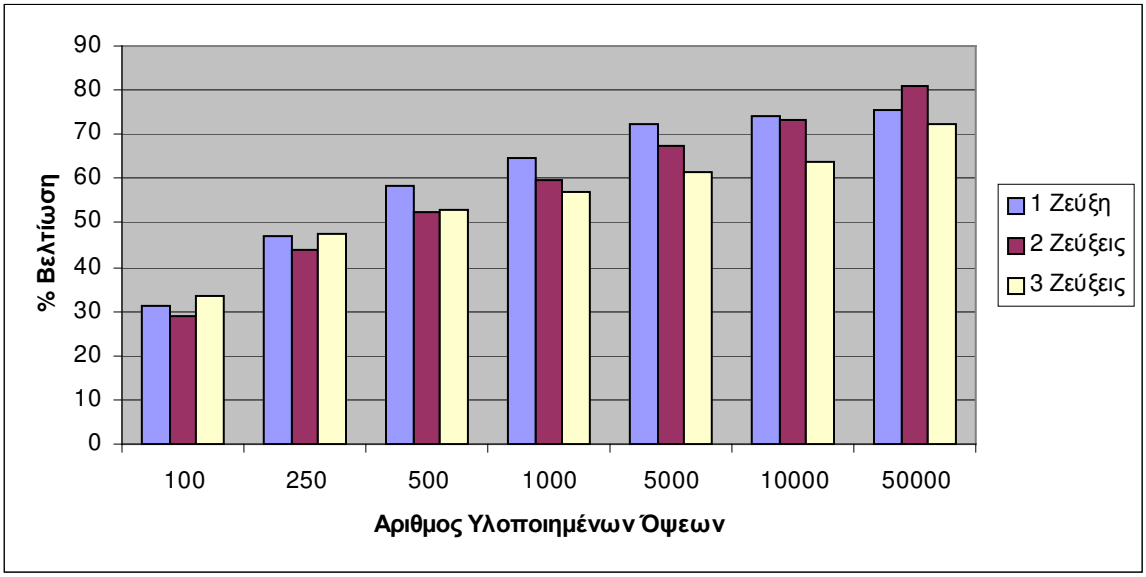
Στην παράγραφο αυτή θα περιγράψουμε την εκτέλεση και τα αποτελέσματα από μια σειρά πειραμάτων έτσι ώστε να δούμε το μέγεθος της βελτίωσης που έχουμε με χρήση όψεων καθώς και την σχέση που έχει το μέγεθος της βελτίωσης με τον αριθμό των χρησιμοποιούμενων υλοποιημένων όψεων. Όταν αναφέρουμε τον όρο βελτίωση εννοούμε βελτίωση σε συνολικό διάβασμα και γράψιμο στον δίσκο του συστήματός μας που θα έχουμε αν χρησιμοποιήσουμε το βέλτιστο πλάνο που χρησιμοποιεί όψεις σε αντίθεση με το βέλτιστο πλάνο το οποίο θα μας δώσει ο ευριστικός βελτιστοποιητής. Στα πειράματα που ακολουθούν προσπαθήσαμε να βελτιστοποιήσουμε επερωτήσεις που πραγματοποιούν μια, δύο και τρεις ζεύξεις ανάμεσα σε δύο, τρεις και τέσσερις πίνακες αντίστοιχα. Για να έχουμε πιο αξιόπιστα αποτελέσματα δημιουργήσαμε 5 επερωτήσεις για κάθε μια από τις τρεις αυτές κατηγορίες και επαναλάβαμε τα πειράματα για έναν μεγάλο αριθμό διαφορετικών συνόλων από περιγραφές υλοποιημένων όψεων. Ο αριθμός των υλοποιημένων όψεων που αναγράφεται στα γραφήματα που ακολουθούν αφορά όψεις που μπορεί να βασίζονται σε οποιονδήποτε από τους τέσσερις πίνακες της βάσης μας και φυσικά μπορούν να συμμετέχουν πάνω του ενός πίνακες σε κάθε μια τέτοια όψη (η όψη να περιέχει ζεύξη ανάμεσα στους πίνακες αυτούς).



Σχήμα 1: Επί τις εκατό βελτίωση επερωτήσεων με χρήση υλοποιημένων όψεων

Στο παραπάνω σχήμα παρουσιάζεται η επί τις εκατό βελτίωση των τριών συνόλων επερωτήσεων χρησιμοποιώντας διαφορετικό αριθμό από όψεις κάθε φορά. Τα σύνολα των υλοποιημένων όψεων που χρησιμοποιήθηκαν στο πείραμα αυτό περιέχουν σε ίση αναλογία όψεις με έναν δύο, τρεις και τέσσερις πίνακες. Το συμπέρασμα στο οποίο μπορούμε να καταλήξουμε από το συγκεκριμένο σχήμα είναι ότι ακόμα και με λίγες όψεις (100) έχουμε μια μικρή βελτίωση. Χρησιμοποιώντας 50.000 όψεις παρατηρούμε βελτίωση περίπου 70% με 80% που είναι πολύ μεγάλη αλλά θα θυσιάσουμε αρκετό χρόνο για να καταλήξουμε στο αποτέλεσμα. Σε γενικές γραμμές το παραπάνω γράφημα μας δείχνει ότι δεν έχει νόημα να χρησιμοποιούμε πάνω από 5000 όψεις μια που η βελτίωση που πραγματοποιείται από το σημείο αυτό και μετά είναι αρκετά μικρή.

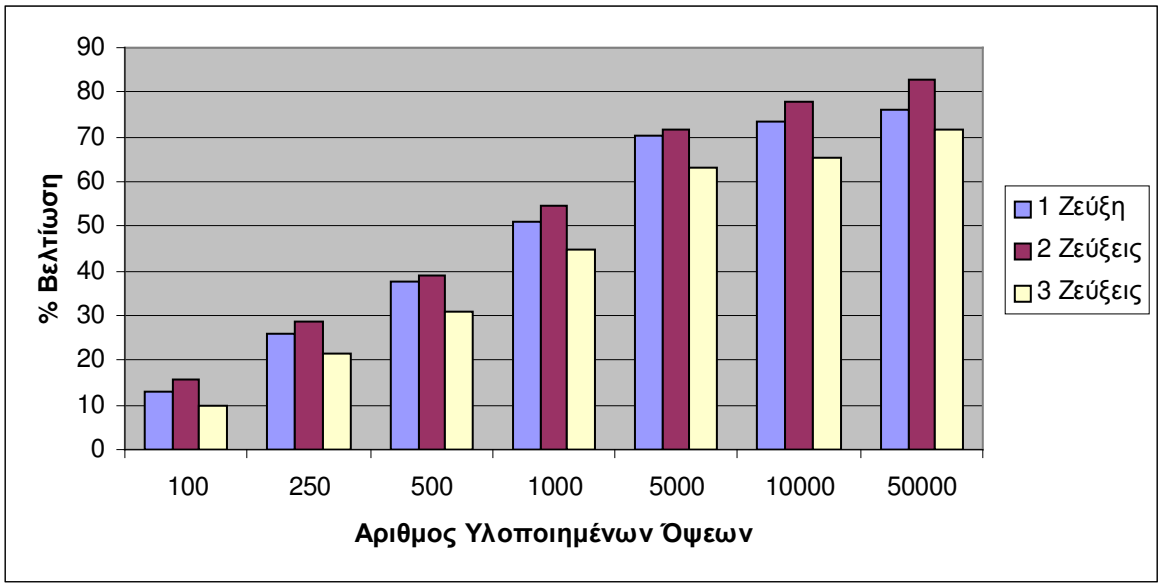
Στο σχήμα που ακολουθεί έχουμε ζητήσει οι όψεις που θα χρησιμοποιηθούν να είναι βασισμένες κατά κύριο λόγο (80%) σε σύνολα του ενός πίνακα και να μην αποτελούν ζεύξης δύο, τριών ή τεσσάρων πινάκων.



Σχήμα 2: Επί τις εκατό βελτίωση επερωτήσεων με χρήση υλοποιημένων όψεων που βασίζονται κυρίως σε έναν (μη συγκεκριμένο) πίνακα

Συγκρίνοντας το σχήμα 2 με το σχήμα 1 βλέπουμε καθαρά ότι οι όψεις που βασίζονται σε έναν μόνο πίνακα είναι πραγματικά χρήσιμες. Αυτό μπορούμε να το δούμε στα σύνολα που περιέχουν μικρό αριθμό από υλοποιημένες όψεις. Αν συγκρίνουμε τα αντίστοιχα σύνολα από τα σχήματα 1 και 2 θα παρατηρήσουμε ότι στο σχήμα 2 έχουμε μεγαλύτερη βελτίωση. Κατά την γνώμη μας είναι πολύ σημαντικό το ότι με 500 μόλις όψεις μπορούμε να πετύχουμε 50% με 60% βελτίωση και στα τρία σύνολα επερωτήσεων που έχουμε. Την ίδια ακριβώς βελτίωση την πετυχαίναμε στο σχήμα 1 με 1000 όψεις. Κλείνοντας τις παρατηρήσεις μας για το συγκεκριμένο σχήμα πρέπει να κρατήσουμε ότι με 125 όψεις ανά πίνακα μπορούμε να μειώσουμε το κόστος της επερώτησής μας κάτω από το μισό σε σχέση με το βέλτιστο ευριστικό πλάνο κάτι που είναι πολύ σημαντικό.

Το γράφημα που παρουσιάζει αποτελέσματα από πειράματα στα οποία προωθούμε κατά κύριο λόγο την χρήση υλοποιημένων όψεων οι οποίες βασίζονται σε (αποτελούν ζεύξη) δύο πινάκων ακολουθεί.

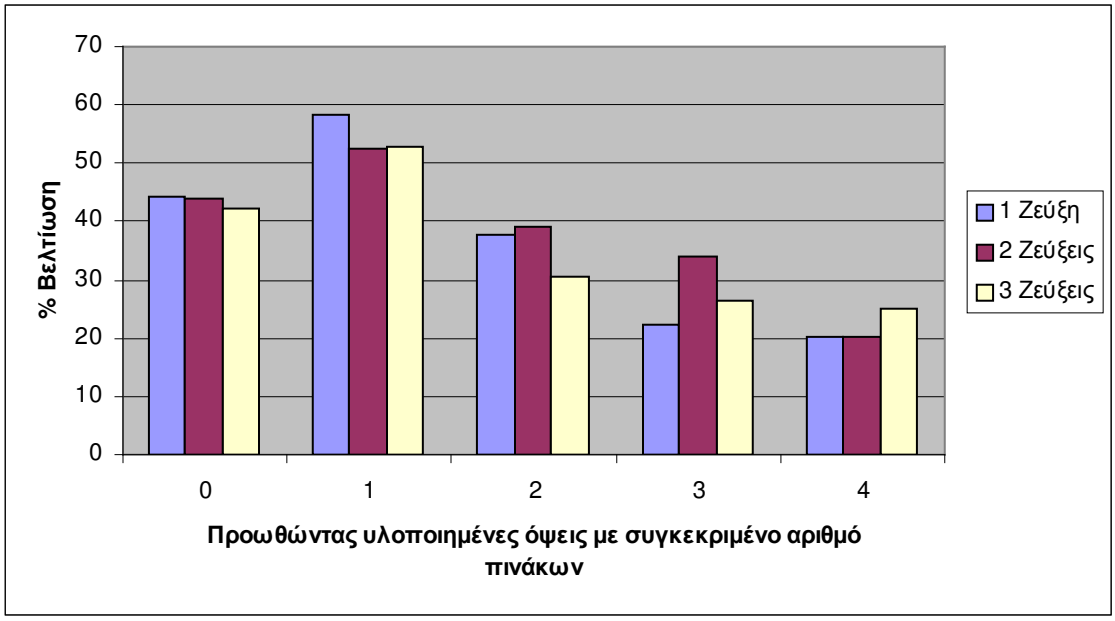


Σχήμα 3: Επί τις εκατό βελτίωση επερωτήσεων με χρήση υλοποιημένων όψεων που βασίζονται κυρίως σε δύο (μη συγκεκριμένους) πίνακες

Αυτό που μπορούμε να πούμε για τα παραπάνω αποτελέσματα είναι ότι με χαμηλό αριθμό όψεων (≤ 500) έχουμε μικρότερη απόδοση σε σχέση με τα αποτελέσματα των

σχημάτων 1 και 2. Παρατηρούμε ακόμα ότι σε μεγάλο αριθμό όψεων (≥ 5000) δεν έχουμε σημαντικές διαφοροποιήσεις σε απόδοση ανάμεσα στα τρία αυτά σύνολα αποτελεσμάτων. Ακόμα μικρότερη απόδοση με χαμηλό αριθμό όψεων θα έχουμε και στην περίπτωση που προωθούμε όψεις με ζεύξεις ανάμεσα σε τρεις αλλά και τέσσερις πίνακες. Αυτό είναι και κάτι το φυσιολογικό μια που οι πιθανότητες να αποβεί χρήσιμη σε μια επερώτηση μια όψη η οποία αποτελεί ζεύξη τεσσάρων πινάκων είναι σαφώς μικρότερη από μια όψη που βασίζεται σε μόνο έναν πίνακα.

Στο επόμενο σχήμα παρουσιάζεται ακριβώς αυτό. Έχουμε σύνολα από 500 υλοποιημένες όψεις και κάθε φορά προωθούμε όψεις βασισμένες σε συγκεκριμένο αριθμό από πίνακες. Το 0 στις πρώτες τρεις στήλες συμβολίζει ότι όλες οι όψεις εμφανίζονται με την ίδια πιθανότητα ανεξαρτήτως αριθμού πινάκων που χρησιμοποιούν ενώ οι αριθμοί 1, 2, 3 και 4 μας δείχνουν ότι έχουμε «ευνοήσει» κατά 80% την δημιουργία όψεων που βασίζονται σε σύνολα του ενός, δύο, τριών αλλά και τεσσάρων πινάκων αντίστοιχα.



Σχήμα 4 : Επί τις εκατό βελτίωση επερωτήσεων με χρήση διαφορετικών ειδών συνόλων από 500 υλοποιημένες όψεις.

Παρατηρούμε ότι την μεγαλύτερη απόδοση την έχουμε όταν προωθούμε όψεις που βασίζονται σε σύνολα του ενός πίνακα ενώ την μικρότερη όταν προωθούμε όψεις που αποτελούν ζεύξεις τεσσάρων πινάκων. Με το συγκεκριμένο σχήμα θα κλείσουμε και την ενότητα της παρουσίασης των αποτελεσμάτων των πειραμάτων κρατώντας ότι με ένα σωστά επιλεγμένο σύνολο από 500 υλοποιημένες όψεις μπορούμε να έχουμε μειώσουμε το κόστος κάτω από το μισό.

4. Σχετική ερευνητική δουλειά

Στο σημείο αυτό θα εξετάσουμε ερευνητικές προσπάθειες που έχουν γίνει στον τομέα της βελτιστοποίησης επερωτήσεων με χρήση υλοποιημένων όψεων.

Στο [12] οι συγγραφείς περιγράφουν έναν αλγόριθμο με τον οποίο μπορούν να μετασχηματιστούν οι επερωτήσεις που δίνονται στο σύστημά τους έτσι ώστε να χρησιμοποιούν και υλοποιημένες όψεις. Ο συγκεκριμένος αλγόριθμος δίνει ουσιαστικά μια επιπλέον επιλογή (το πλάνο με χρήση υλοποιημένων όψεων) στον βελτιστοποιητή του Microsoft SQL Server και ο τελευταίος αποφασίζει αν θα την χρησιμοποιήσει. Η αντικατάσταση πραγματοποιείται σε select-project-join-group-by (SPJG) εκφράσεις που προσπαθεί να τις αντικαταστήσει με μια υλοποιημένη όψη. Στο σημείο αυτό διαφέρει από την εργασία μας μια που πραγματοποιεί μια αντικατάσταση ανά επερώτηση και όχι περισσότερες όμως πραγματοποιεί αντικαταστάσεις σε πιο γενικές εκφράσεις σε αντίθεση με τις select-project-join (SPJ) εκφράσεις που επεξεργαζόμαστε εμείς. Αυτό που είναι πραγματικά καινοτόμο στην συγκεκριμένη δουλειά είναι ένα πολυ-επίπεδο ευρετήριο που χρησιμοποιείται έτσι ώστε να μπορεί να βρίσκει γρήγορα ποιες όψεις είναι σχετικές με μια SPJG έκφραση. Το συγκεκριμένο ευρετήριο αποτελείται από επιμέρους υπο-ευρετήρια το οποίο σχετίζεται με μια συγκεκριμένη ιδιότητα των όψεων (π.χ. το σύνολο των πινάκων της όψης, το σύνολο των στηλών που προβάλλονται κλπ.). Τα συγκεκριμένα υπο-ευρετήρια οργανώνονται σε ιεραρχική δομή και το καθ' ένα χωρίζει το σύνολο των όψεων με βάση κάποια διαφορετική ιδιότητα. Τέλος οι συγγραφείς περιγράφουν έναν αριθμό πειραμάτων στα οποία παρουσιάζεται ότι ο συγκεκριμένος αλγόριθμος αυξάνει πολύ λίγο τον συνολικό χρόνο βελτιστοποίησης, ακόμα και αν ο αριθμός των υλοποιημένων όψεων φτάνει τις 1000.

Στο [13] οι συγγραφείς περιγράφουν πως πραγματοποιείται ο μετασχηματισμός μιας επερώτησης έτσι ώστε να χρησιμοποιούνται υλοποιημένες όψεις από τον βελτιστοποιητή IBM DB2 UDB. Ο αλγόριθμός τους χρησιμοποιεί το μοντέλο γράφου επερώτησης (Query Graph Model) για να αναπαραστήσει μια επερώτηση [14]. Με βάση το συγκεκριμένο μοντέλο μια επερώτηση μπορεί να χωριστεί σε πολλαπλά QGM κουτιά που το κάθε ένα αντιστοιχεί σε ένα select-project-join τμήμα. Ο αλγόριθμος προσπαθεί να ταιριάξει ζευγάρια

από κουτιά QGM των όψεων με αυτά της επερώτησης. Για να το καταφέρει αυτό διατρέχει το QGM από κάτω προς τα πάνω αρχίζοντας από τα φύλλα. Η αντιστοίχηση ανάμεσα σε κουτιά της επερώτησης και μιας όψης μπορεί είναι άμεση, αν τα δύο κουτιά αναπαριστούν ισοδύναμες επερωτήσεις ή μπορεί να χρειάζεται εξισορρόπηση (compensation). Η εξισορρόπηση είναι ένα σύνολο από επιπλέον πράξεις που πρέπει να πραγματοποιηθούν εξωτερικά της όψης έτσι ώστε το αποτέλεσμα που θα προκύπτει να είναι ισοδύναμο με αυτό του αντίστοιχου κουτιού της επερώτησης. Ο αλγόριθμος λαμβάνει υπ' όψιν του ένα ζευγάρι από κουτιά μόνο αν έχει εφαρμοστεί ο αλγόριθμος της αντιστοίχησης ανάμεσα σε όλα τα δυνατά ζευγάρια των παιδιών τους. Για τον λόγο αυτό η αντιστοίχηση (και η εξισορρόπηση που θα προκύψει) μπορεί να πραγματοποιηθεί χωρίς να κοιτάζουμε τα υποδένδρα των παιδιών. Ο αλγόριθμος τερματίζει όταν βρεθεί αντιστοίχηση ανάμεσα μεταξύ της ρίζας της όψης και κάποιου κουτιού στο γράφο QGM της επερώτησης.

Στο [15] οι συγγραφείς περιγράφουν το πως πραγματοποιείται η βελτιστοποίηση επερωτήσεων με χρήση όψεων στο Oracle 8i DBMS. Ο αλγόριθμός τους αποτελείται από δύο φάσεις. Κατά την πρώτη φάση προσπαθεί να αντικαταστήσει τμήματα της επερώτησης με αντιστοιχίσεις σε ήδη υπάρχουσες υλοποιημένες όψεις ενώ κατά την δεύτερη φάση ο αλγόριθμος συγκρίνει το εκτιμώμενο κόστος των δύο πλάνων. Δηλαδή του αποτελέσματος της πρώτης φάσης καθώς και του βέλτιστου πλάνου χωρίς την χρήση υλοποιημένων που θα δώσει ο βελτιστοποιητής. Το προτέρημα της συγκεκριμένης προσέγγισης είναι η ευκολία της υλοποίησης μια που η δυνατότητα χρήσης υλοποιημένων όψεων προστίθεται στο βελτιστοποιητή χωρίς να χρειάζεται να αλλαχτεί η σειρά με την οποία επεξεργάζεται και πραγματοποιεί τις ζεύξεις μιας επερώτησης. Το μειονέκτημα είναι ότι λαμβάνει υπ όψιν της μόνο ένα πλάνο που χρησιμοποιεί όψεις με αποτέλεσμα να υπάρχει η περίπτωση να χάσει την πιο μικρό σε κόστος πλάνο που χρησιμοποιεί υλοποιημένες όψεις.

Τέλος στο [16] οι συγγραφείς περιγράφουν έναν αλγόριθμο ο οποίος προσπαθεί να ενσωματώσει ομοιόμορφα την χρήση υλοποιημένων όψεων, εξειδικυμένων ευρετηρίων και σημασιολογικών περιορισμών ακεραιότητας (semantic integrity constrains). Και οι τρεις αυτές περιπτώσεις αναπαριστούνται ως περιορισμοί. Ο αλγόριθμός τους λειτουργεί σε δύο φάσεις. Κατά την πρώτη φάση (η οποία ονομάζεται "chase") η επερώτηση επεκτείνεται έτσι ώστε να συμπεριλάβει όποια άλλη δομή είναι σχετική, όπως για παράδειγμα υλοποιημένες όψεις. Με τον τρόπο αυτό δημιουργείται ένα καθολικό πλάνο το οποίο κατά την δεύτερη

φάση (την “back-chase”) προσπαθεί ο βελτιστοποιητής να μειώσει το κόστος του στο ελάχιστο δυνατό αφαιρώντας δομές (άρα και ζεύξεις) από αυτό. Στο σημείο αυτό πρέπει να αναφέρουμε ότι η πρώτη φάση (chase) είναι μια γενίκευση του αλγόριθμου που εμφανίστηκε στο [17] έτσι ώστε να μπορεί να χειρίζεται και αντικειμενοστραφείς επερωτήσεις. Περιγραφή της υλοποίησης του συγκεκριμένου αλγορίθμου αλλά και πειράματα που παρουσιάζουν την απόδοσή του μπορούν να βρεθούν στο [18].

5. Στόχοι – Περαιτέρω έρευνα

Πρέπει να ομολογήσουμε ότι τα αποτελέσματα των πειραμάτων είναι αρκετά εντυπωσιακά. Πετυχαίνουμε με χρήση 50.000 υλοποιημένων όψεων μείωση κόστους επερωτήσεων που ξεπερνά σε ορισμένες περιπτώσεις και το 80%. Ακόμα και με ένα όχι τόσο προσεκτικά επιλεγμένο σύνολο από 500 όψεις μπορούμε να μειώσουμε το κόστος κάτω από το μισό. Αυτό που δεν πρέπει να ξεχνάμε είναι ότι στην παραπάνω μείωση κόστους δεν έχει συμπεριληφθεί το κόστος που χρειάζεται το δικό μας πρόγραμμα για να βρει την βέλτιστη λύση, κάτι που θα μειώσει την συνολική βελτίωση. Παρόλα αυτά πιστεύουμε ότι με ένα σωστά επιλεγμένο μικρό σύνολο από υλοποιημένες όψεις θα έχουμε πολύ θετικά αποτελέσματα.

Οι κατευθύνσεις που πρέπει να κινηθούμε πια είναι:

- Η βελτίωση του προγράμματος έτσι ώστε να υποστηρίζει και πιο γενικές επερωτήσεις που χρησιμοποιούν και διαζεύξεις πέρα από συζεύξεις ανάμεσα στους περιορισμούς που δίνουμε.

- Η σύνδεση των υλοποιημένων όψεων που αποθηκεύονται με τις επερωτήσεις που υποβάλλονται στο σύστημα. Ήδη έχουμε δείξει ότι αν επιλεχθεί κατάλληλα έστω και ένα μικρό σύνολο υλοποιημένων όψεων η βελτιστοποίηση που πραγματοποιείται με αυτές είναι αξιόλογη. Αν γίνει μια ακόμα καλύτερη επιλογή του συνόλου θα είναι ενδιαφέρον να εξετάσουμε το μέγεθος της συγκεκριμένης βελτίωσης.

- Δοκιμή όλων των παραπάνω σε πραγματικές συνθήκες. Να συνδέσουμε το σύστημά μας με ένα ΣΔΒΔ έτσι ώστε να λαμβάνουμε πραγματικά στατιστικά στοιχεία για τα δεδομένα της βάσης και να μετρήσουμε πραγματικούς χρόνους εκτέλεσης της βέλτιστης επερώτησης που δίνει το σύστημά μας χωρίς χρήση υλοποιημένων όψεων και της βέλτιστης που δίνει με χρήση υλοποιημένων όψεων. Στον τελευταίο χρόνο πρέπει να προσθέσουμε και τον χρόνο που χρειάζεται το πρόγραμμά μας για να καταλήξει στην βέλτιστη λύση με χρήση υλοποιημένων όψεων.

6. Βιβλιογραφία

[1] Sellis T. K., “Multiple-query optimization”, ACM Transactions on Database Systems, Vol. 13, No. 1 (1988), pp.23-52.

[2] Hall P. A. V., “Optimization of a single relational expression in a relational database”, IBM J. Research and Development (1976), pp. 244-257.

[3] Minker J., “Search strategy and selection function for an inferential relational system”, ACM Transactions on Database Systems 3:1 (1978), pp 1-31.

[4] Pecherer R. M., “Efficient evaluation of expressions in a relational algebra” Proc. of ACM Pacific Conf. (1975), pp. 44-49.

[5] Smith J. M. and Chang P. Y., “Optimizing the performance of a relational algebra database interface”, Comm. ACM 18:10 (1975), pp. 568-579.

[6] Selinger P. G., et al., “Access path selection in a relational database management system”, Proc. of ACM-SIGMOD Int. Conf on Manage of Data (1979).

[7] Blasgen M. W. and Eswaran K. P., “Storage and access in relational databases” IBM Syst. J 16, 4 (1977).

[8] Knuth D., “The art of computer programming: sorting and searching”, Addison-Wesley (1973).

[9] Sapiro L. D., “Join processing in database systems with large main memories”, ACM Transactions on Database Systems, Vol. 11, No. 3 (1986), pp. 239-264.

[10] Larson P.A., Yand H.Z., “Computing queries from derived relations”, VLDB (1985), 259-269.

[11] Yand H.Z., Larson P.A., “Query transformation for PSJ-queries”, Proc. of VLDB (1987) pp. 245-254.

[12] Goldstein J., Larson P. A., “Optimizing queries using materialized views: A Practical, Scalable Solution”, ACM SIGMOD (2001).

[13] Zaharioudakis M., Cochrane R., Liapis G., Pirahesh H., Urata M., “Answering complex SQL queries using automatic summary tables” Proc. of SIGMOD (2000), pp.105-116.

[14] Haas L., Freytag J., Lohman G., Pirahesh H.,”Extensible query processing in starburst”, Proc. of SIGMOD (1989), pp 377-388.

[15] Bello R., Dias K., Downing A., Feenan J., Finnerty J., Norcott W., Sun H., Witkowski A.,Ziauddin M., “Materialized views in oracle”, Proc of VLDB (1998), pp. 659-664.

[16] Deutsch A., Popa L., Tannen V., “Physical data independence, constraints and optimization with universal plans”, Proc. of VLDB (1999), pp. 459-470.

[17] Popa L., Tannen V., “An equational chase for path conjunctive queries, constrains and views”, Proc ICDT (1999).

[18] Popa L., Deutsch A., Sahuguet A., Tannen V. “A chase too far?”, Proc. of SIGMOD (2000), pp.273-284.