Μέλη ομάδας:

Παναγιώτα Αδαμοπούλου 1115201400003

Ανδρέας Αθανασίου 1115201400005

Παναγιώτα Παπαδήμα 1115201400143

Αναφορά παραδοτέου:

Μεταγλώττιση: make

Εκτέλεση: ./radixhash

Αρχεία εισόδου:

1)Είσοδος αρχείων και αποθήκευση στη μνήμη:

Το εκτελέσιμο παίρνει σαν είσοδο τα αρχεία με τις εγγραφές και τα περνάει στη μνήμη. Παράλληλα με το διάβασμα του κάθε αρχείου υπολογίζονται τα στατιστικά και οι πληροφορίες που χρειαζόμαστε από κάθε πίνακα. Η είσοδος των αρχείων σταματάει όταν δοθεί σαν είσοδος η λέξη "Done".

Οι δομές που χρησιμοποιούνται για την αποθήκευση των δεδομένων βρίσκονται στο rhj.h . Η δομή all_data κρατάει έναν πίνακα με δείκτες . Κάθε δείκτης δείχνει σε ένα relation_data . Για κάθε αρχείο που εισάγεται το μέγεθος του πίνακα στην δομή all_data επεκτείνεται κατά ένα και εκεί αποθηκεύονται τα δεδομένα του αρχείου.

2) Queries :

Στην συνέχεια το πρόγραμμα περιμένει να πάρει σαν είσοδο τα queries που θα εκτελέσει. Κάθε ομάδα queries αρχίζει να εκτελείται αφου δοθεί σαν είσοδος το F. Οι δομές που χρησιμοποιούνται για την αποθήκευση των queries βρίσκονται στο predicates.h.

Η δομή batch κρατάει έναν πίνακα με δείκτες σε query και των αριθμό των queries. Η δομή query αποτελείται από το string με τις σχέσεις , string με τις προβολες , εναν πίνακα με δείκτες στα κατηγορήματα , των αριθμό των κατηγορημάτων κάθε query και των αριθμό των σχέσεων που συμμετέχουν στο query. Η επεξεργασία και η αποθήκευση του αρχικού string γίνεται με τις συναρτήσεις seperate_predicate(..,..) και AddToBatch(..,..);

Αφού δοθεί το F εκτελούνται τα query ένα ένα με την σειρά που δόθηκαν με την execute_query(..,..) , το σώμα της βρίσκεται στο predicates.c .

3) Εκτέλεση query :

Για κάθε query δημιουργείται ένας πίνακας με δείκτες σε relation_data και μέγεθος όσο οι σχέσεις που συμμετέχουν σε query με την χρήση της find_corresponding(..) . Με αυτόν τον

πίνακα μπορούμε να έχουμε πρόσβαση στα δεδομένα κάθε πίνακα . Για παράδειγμα αν οι σχέσεις που συμμετέχουν στο query είναι οι 2 , 4 , 6 , ο πίνακας relations περιέχει :

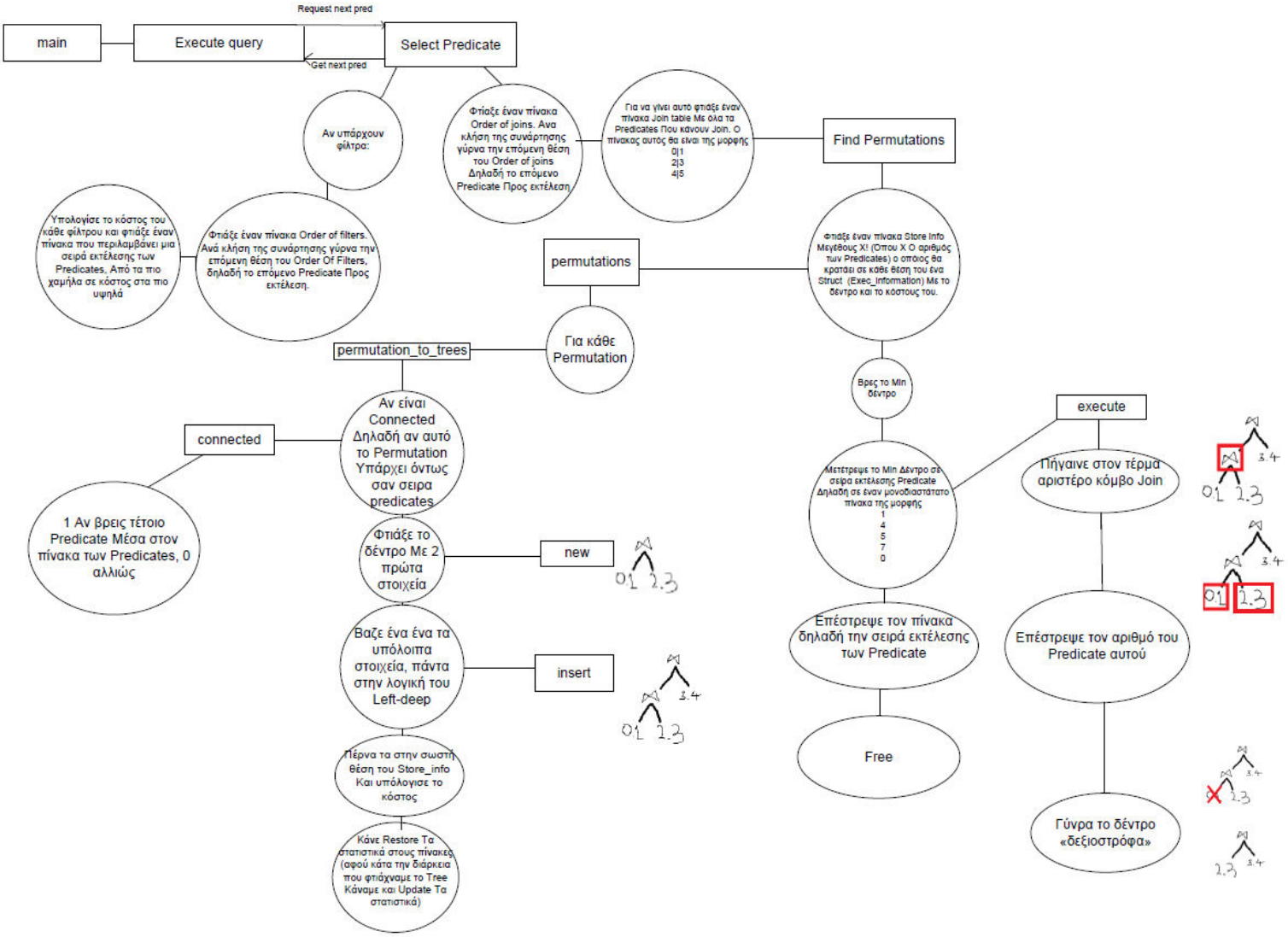
relations[0] = δεδομένα πίνακα 2

relations[1] = δεδομένα πίνακα 4

relations[2] = δεδομένα πίνακα 6

Έπειτα με την χρήση της select_predicate(..) και με βάση τα στατιστικά υπολογίζεται ποιο predicate θα εκτελεστεί .

Η select statistics συνοψίζεται στο ακόλουθο διάγραμμα:



Η restore statistics είναι μια συνάρτηση που μας επιτρέπει να επιστρέψουμε τα στατιστικά σε αυτά που διαβάσαμε στην αρχή, χρησιμοποιώντας μεταβλητές της δομη΄ς που έχουμε χρησιμοποιήσει για να αποθηκεύσουμε τα αρχικά στατιστικά.

Αφού επιλεχθεί το κατηγόρημα , με την χρήση της execute_predicate εκτελείται και ανανεώνονται τα αποτελέσματα.

4) Εκτελεση κατηγορήματος :

Στην execute_predicate(..) εκτελείται ένα κατηγόρημα ανάλογα με το αν είναι φίλτρο , αν είναι join ή αν πρέπει να γίνει σειριακός έλεγχος. Επίσης ελέγχεται αν μία σχέση έχει ενδιάμεσα αποτελέσματα ή οχι . Αν δεν έχει , επιλέγεται το αρχικό relation για να συμμετέχει στην εκτέλεση του κατηγορήματος , αλλιώς με την χρήση της BuildRelation(..) δημιουργείται μία καινούρια σχέση που σαν κλειδιά έχει τα index των κλειδιών μέσα στα ενδιάμεσα και σαν payloads τα αντίστοιχα payloads.

5 ) Εκτέλεση Φίλτρου :

Έχουμε χρησιμοποιήσει threads στην εκτέλεση του φίλτρου. Δημιουργούνται τόσα FilterJobs όσα ο αριθμός των threads , και κάθε νήμα δημιουργεί την δική του δομή αποτελεσμάτων. Αφού τελειώσουν και τα 3 , το γονεϊκό thread συνενώνει τα αποτελέσματα και ανανεώνει την δομή των ενδιάμεσων αποτελεσμάτων.

5) Εκτέλεση σειριακού ελέγχου :

Έχουμε χρησιμοποιήσει threads στην εκτέλεση του σειριακού έλεγχου με τον ίδιο τρόπο όπως στα φίλτρα.

6) Εκτέλεση Join :

Η εκτέλεση του Join έχει παραλληλοποιηθεί όπως ζητήθηκε.

7) Ενδιάμεσα αποτελέσματα:

Για την αποθήκευση των ενδιάμεσων αποτελεσμάτων έχει χρησιμοποιηθεί η δομή inbetween results (inbetween.h). Αποτελείται από έναν δισδιάστατο πίνακα ακεραίων NxM , όπου N ο αριθμός των πινακών που συμμετέχουν στο query και M ο αριθμός των μέχρι τώρα αποτελεσμάτων , των αριθμό των σχέσεων που συμμετέχουν και έναν μονοδιάστατο πίνακα με όνομα joined ο οποίος αρχικοποιείται με -1 και έχει μέγεθος N. Από αυτόν καταλαβαίνουμε αν μία σχέση έχει ενδιάμεσα αποτελέσματα (joined[relId]=-1) ή πόσα αποτελέσματα βρίσκονται

στην ενδιάμεση δομή . Μετά την εκτέλεση κάθε κατηγορήματος η δομή αυτή ανανεώνεται ως εξής:

Αν οι σχέσεις που συμμετέχουν στο κατηγόρημα δεν έχουν ενδιάμεσα αποτελέσματα εισάγονται τα κλειδιά που έχουν επιστραφεί από την εκτέλεση του κατηγορήματος με την δομή inbet_list .

Αν μία απο τις δύο σχέσεις έχουν ενδιάμεσα αποτελέσματα , σημαίνει ότι η δομή inbet_list αντί για κλειδιά της αρχικής σχέσης περιέχει indexes των προηγούμενων αποτελεσμάτων. Έτσι φτιάχνεται καινούρια δομή ενδιάμεσων αποτελεσμάτων , και για κάθε σχέση που έχει ενδιάμεσα αποτελέσματα στην παλιά δομή , εισάγονται στη νέα δομή τα ανανεωμένα αποτελέσματα της σύμφωνα με την δομή inbet_list.

Αν μία απο τις δύο σχέσεις δεν έχει ενδιάμεσα αποτελέσματα , εισάγονται ένα ένα και τα δικά της κλειδιά στην καινούρια δομή.

Όταν τελειώσει η εκτέλεση όλων των κατηγορημάτων , καλείται η show_results η οποία βρίσκει τα ζητούμενα αθροίσματα και τα εμφανίζει.

Όταν τελειώσει η εκτέλεση όλων των query το πρόγραμμα περιμένει σαν είσοδο ένα νέο batch of queries.

8) Threadpool (thpool.c , thpool.h)

Υπάρχουν σχόλια μέσα στα αρχεία που εξηγούν την λειτουργία.