Project (Part 2)

‘Ονομα : Γιαννούκος Ορέστης AM : 1115 2016 00035
‘Ονομα : Λιακόπουλος Γεώργιος AM : 1115 2016 00091
‘Ονομα : Χασιαλής Κωνσταντίνος AM : 1115 2016 00195

Επεξηγήσεις / Παραδοχές στον κώδικα :

Επεξηγησεις απο το part 1 :

Το κομμάτι του sort αποφασίσαμε να το κάνουμε επαναληπτικά και όχι αναδρομικά , καθώς ειπώθηκε στο μάθημα ότι η αναδρομή δεν θα βοηθούσε στην τρίτη φάση του Project. Στην συνάρτηση iterative_sort που βρίσκεται στο αρχείο join.c βρίσκεται η υλοποίηση της ταξινόμησης. Επιλέξαμε να επεξεργαζόμαστε τον πίνακα με τα tuples “κατά πλάτος” , δηλαδή σκανάρουμε όλο τον πίνακα φτιάχουμε το κατά πλάτος” , δηλαδή σκανάρουμε όλο τον πίνακα φτιάχουμε το ιστόγραμμα και το psum και κάθε bucket του νέου reordered πίνακα το βάζουμε σεμια ουρά . Στην συνέχεια κάθε bucket που κάνουμε pop εξετάζεται αν μπορεί να εφαρμοστεί σε αυτό quicksort , αλλιώς ακολουθεί ίδια διαδικασία. Επομένως είναι φανερός ο όρος “κατά πλάτος” , δηλαδή σκανάρουμε όλο τον πίνακα φτιάχουμε το κατά πλάτος” αφού στην αρχή θα επεξεργαστούμε πρώτα το 1ο bucket του reordered πίνακα , μετά το 2ο κλπ.
Στον έλεγχο για quicksort μετράμε τα bytes των tuples + τα bytes του num_tuples να είναι μικρότερα από 64 KB.

Στο κομμάτι του join διατρέχουμε παράλληλα τους δυο πίνακες με τα tuples και ανάλογα με τις τιμές των κλειδιών αποφασίζουμε το index ποιανού πίνακα να αυξήσουμε .

Επεξηγησεις απο το part 2 :

Για την εκτέλεση του κάθε query , αρχικά μεταθέτουμε τα predicates ωστε να μπουν όλα τα φίλτρα μπροστά και στην συνέχεια γκρουπάρουμε join στα οποία κάποιος από τους δύο τελεστέους είναι ίδιος , δηλαδή για παράδειγμα 1.0=2.1 && 2.1 = 3.0.

Για τα ενδιάμεσα αποτελέσματα αποφασίσαμε να κρατάμε όλες τις οντότητες γιατί μπορεί να ξαναχρησιμοποιηθούν. ‘Οπως για παράδειγμα στο query
3.0 < 33199 & 0.2 = 1.0 & 1.0 = 2.1 & 0.1 = 3.2 , θα δημιουργηθεί μια οντότητα ενδιάμεσων αποτελεσμάτων με Rowid 3 από το φίλτρο , μετά θα δημιουργηθεί νέα οντότητα από το πρώτο join με Rowid 0 | Rowid 1 , στο τελευταίο join ομως θα χρειαστούμε τα rowid0 και τα rowid3 , για αυτό δεν τις διαγράφουμε.

Παρατηρήσαμε ότι μετά το join προκύπτουν κάποια duplicates , τα οποία τα αφαιρούμε μέσω hash-table .
Η συνάρτηση join_payloads που βρίσκεται στο αρχείο join.c η οποία συνδιάζει παλιά και καινούρια αποτελέσματα για την διαμόρφωση των υπόλοιπων payloads που δεν συμμετείχαν στην join. Για παράδειγμα στο παραπάνω παράδειγμα
1.0=2.1 && 2.1 = 3.0. όταν γίνεται το join 2.1 =3 .0 η join_payloads θα προσθέσει τα payloads που πρέπει και στην σχέση 0 για να βγάλει τα αποτελέσματα
Rowid 1 | Rowid 2 | Rowid 3 .

Για τις περιπτώσεις queries όπου στις σχέσεις δίνεται κάτι σαν αυτό : 12 1 6 12 , αποφασίσαμε να ξεχωρίζουμε την σχέση 12 με βάση το id που της δόθηκε στο query , δηλαδή θα δημιουργηθούν 2 στήλες στα ενδιάμεσα αποτελέσματα , μια για την σχέση 12 (id = 0) και μια για την σχέση 12 (id =3) αφού και στα predicates αναφέρονται σαν διαφορετικές σχέσεις.