Πληροφορική & Τηλεπικοινωνίες

K23α - Ανάπτυξη Λογισμικού Για Πληροφοριακά Συστήματα

Χειμερινό Εξάμηνο 2019 – 2020

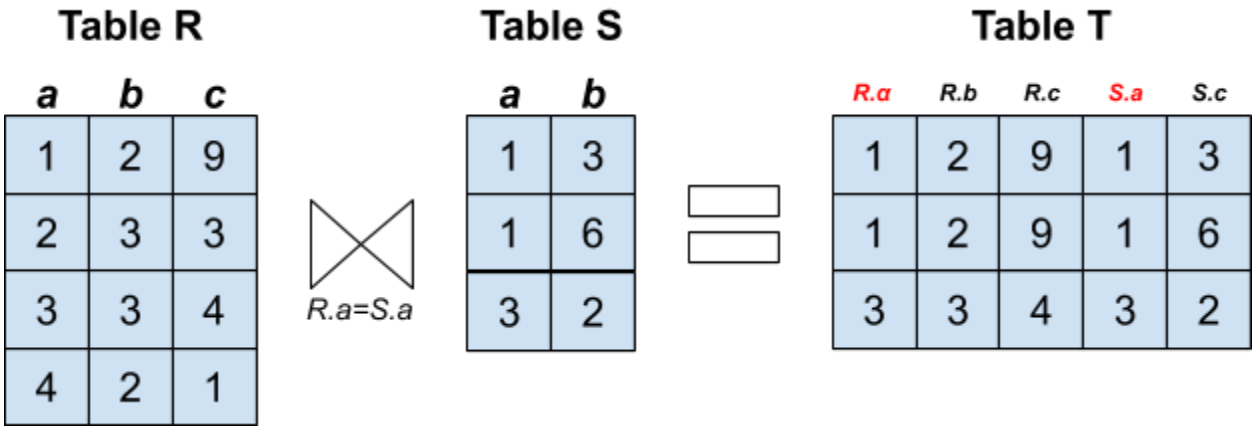
Καθηγητής Ι. Ιωαννίδης

Άσκηση 1 – Παράδοση: Τρίτη 05 Νοεμβρίου 2019

Με την εξέλιξη των υπολογιστών τα τελευταία χρόνια, η σχέση τιμής ανά GB για μνήμες RAM είναι πολύ μικρή ενώ ταυτόχρονα αυξάνονται ολοένα και περισσότερο ο αριθμός των πυρήνων σε κάθε CPU. Σήμερα είναι σύνηθες να συναντάμε Servers με 256 GB RAM και 20 επεξεργαστικούς πυρήνες. Σε αυτά τα νέα δεδομένα hardware προσπαθούν να προσαρμοστούν οι σύγχρονες σχεσιακές βάσεις δεδομένων. Σε αυτο το εξάμηνο θα προσπαθήσουμε να δημιουργήσουμε ένα υποσύνολο μιας βάσης δεδομένων που διαχειρίζεται δεδομένα που βρίσκονται εξ ολοκλήρου στη RAM. Στο πρώτο μέρος θα ασχοληθούμε με την υλοποίηση του βασικό σχεσιακού τελεστή που θα χρειαστούμε για τα μετέπειτα κομμάτια της άσκησης που είναι ο τελεστής ζεύξης ισότητας.

Τελεστής ζεύξης

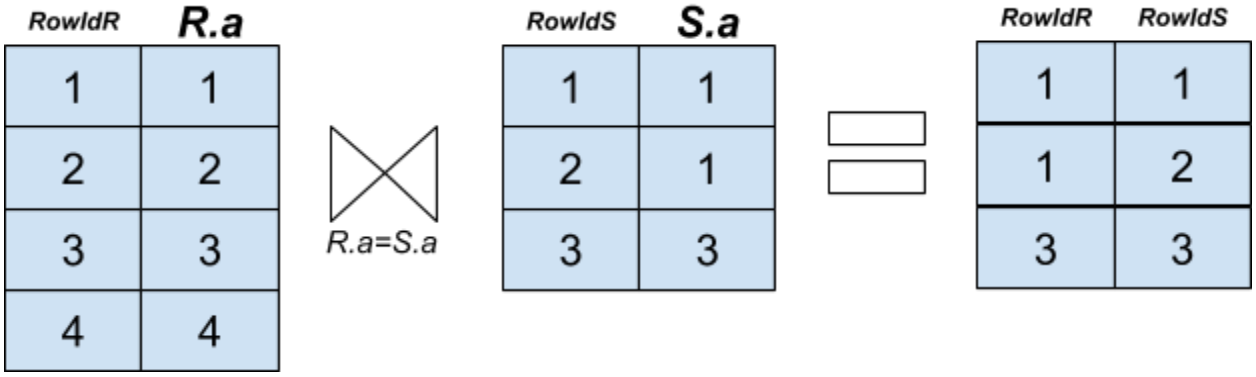
Πρίν προχωρήσουμε στις δομές που θα χρειαστείτε να φτιάξετε για την πρώτη άσκηση θα δούμε ένα παράδειγμα με το πώς δουλεύει ένας τελεστής ζεύξης ισότητας. Στην πιο κάτω εικόνα, φαίνεται το αποτέλεσμα της πράξεις $R.\alpha = S.\alpha$ μεταξύ δύο σχεσιακών πινάκων (R, S):



Η πιο πάνω απεικόνιση θεωρεί ότι οι πίνακες αποθηκεύονται στην μνήμη κατα σειρές (row store). Αυτό πρακτικά σημαίνει, ότι το κάθε στοιχείο μιας στήλης θα απέχει από το επόμενο του, τόσα στοιχεία μακριά όσες είναι και ο αριθμός των στηλών της αντίστοιχης σχέσης. Αυτο σε αρκετες περιπτωσεις είναι χρονοβόρο, γιατί καλούμαστε να διαχειριζόμαστε πολύ περισσότερη

μνήμη, από όση στην πράξη χρειαζόμαστε. Μια άλλη τεχνική, αποθηκεύει τις σχέσεις ανά στήλες. Αυτό σημαίνει, ότι στη μνήμη τα στοιχεία μιας στήλης είναι σειριακά το ένα δίπλα στο άλλο. Με αυτόν τον τρόπο, αν θέλουμε να διαβάσουμε μια στήλη, “ακουμπάμε” πολύ λιγότερη μνήμη.

Για να εκμεταλλευτούμε την αποθήκευση ανά στήλες θα θέλαμε ο τελεστής ζεύξης να χρησιμοποιεί μόνο τις στήλες που γίνεται η πράξη και το αποτέλεσμα να είναι σε τέτοια μορφή ώστε να μπορούμε να ανακτήσουμε όλες τις στήλες από τους πίνακες που παίρνουν μέρος. Για τον λόγο αυτό εισάγουμε την έννοια του rowId. Το rowId είναι μια ένδειξη του αριθμού της γραμμής που ανήκει ένα συγκεκριμένο στοιχείο μιας στήλης.. Με αυτόν τον τρόπο, μπορούμε να αποθηκεύσουμε κάθε στήλη μιας σχέσης, σαν μια ξεχωριστή σχέση με δύο στήλες, όπου η πρώτη στήλη είναι το rowId και η δεύτερη οι πραγματικές τιμές της στήλης. Το αποτέλεσμα της ζεύξης δύο τέτοιων σχέσεων, είναι μια σχέση με δύο στήλες όπου αναφέρονται στα ζευγάρια από rowIds τις πρώτης και της δεύτερης σχέσης που ταιριαζουν με βάση την συνθήκη της ισότητας. Η πιο κάτω εικόνα, απεικονίζει την εκτέλεση του τελεστή της ζεύξης του προηγούμενου παραδείγματος, χρησιμοποιώντας αποθήκευση ανα στήλες.



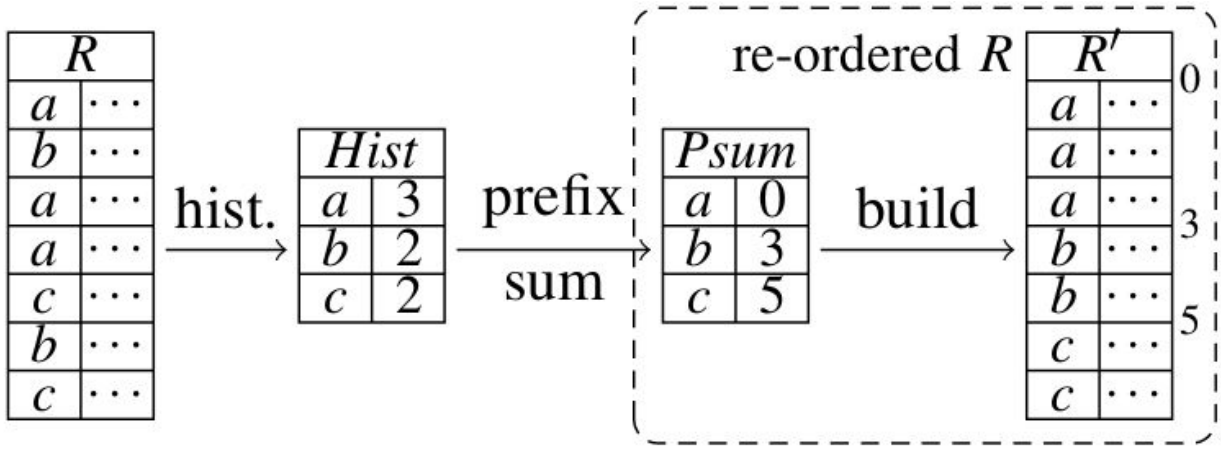
Sort Merge Join

Στο πρώτο μέρος της άσκησης καλείστε να υλοποιήσετε των τελεστή ζεύξης ισότητας υποθέτοντας ότι τα δεδομένα σας είναι αποθηκευμένα ανά στήλες. Πιο συγκεκριμένα καλείστε να υλοποιήσετε τον Sort Merge Join αλγόριθμο. Η ιδέα του SMJ είναι να ταξινομηθούν οι σχέσεις ως προς το κλειδί της ζεύξης τους και στη συνέχεια η ζεύξη να γίνει με απλή συγχώνευση (merge) καθώς θα έχουμε ένα πέρασμα ταξινομημένων πινάκων.

Στο στάδιο της ταξινόμησης, το οποίο θα είναι και το πιο απαιτητικό, οι σχέσεις θα ταξινομούνται με την εξής μέθοδο (radix-sort). Τα δεδομένα από τις σχέσεις θα αναλυθούν σε κάδους (buckets) μεγέθους όσο και η L1-cache του επεξεργαστή (θεωρήστε το μέγιστο μέγεθος 64KB). Οι κάδοι προκύπτουν ως εξής: Αρχικά θα τμηματοποιηθούν τα δεδομένα σε 2^8 τμήματα ανάλογα με την τιμή των πρώτων 8 bits τους.

Ο νέος πίνακας θα προκύψει από την εξής διαδικασία: Δεσμεύουμε ένα καινούργιο πίνακα με μέγεθος όσο ο αρχικός. Στον πίνακα αυτό θα αποθηκεύσουμε σε σειρα τα δεδομένα

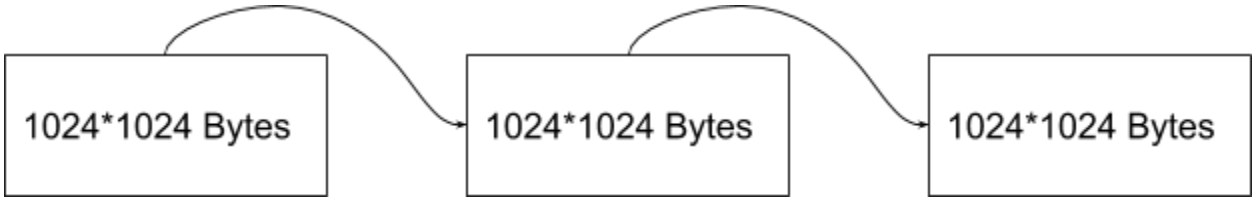
του κάθε κάδου. Για να το κάνουμε αυτό χρειάζεται να ξέρουμε σε ποια θέση του νέου πίνακα ξεκινάνε τα δεδομένα του κάθε κάδου. Για τον λόγο αυτό δημιουργούμε ένα πίνακα (ιστόγραμμα) 2^8 θέσεων όπου στην κάθε θέση του κρατάμε το πλήθος των στοιχείων που υπάρχουν σε αυτόν τον κάδο. Στη συνέχεια θα δημιουργήσουμε το αθροιστικό του ιστόγραμμα που θα δείχνει στη θέση από όπου θα ξεκινάει ο κάθε καδος. Έχοντας αυτό το ιστόγραμμα μπορούμε με ένα πέρασμα του αρχικού πίνακα να γράψουμε τα δεδομένα στη σωστή θέση του νέου πίνακα. Η φάση της τμηματοποίησης φαίνεται στο πιο κάτω περιληπτικό σχήμα:



Η διαδικασία αυτή θα επαναληφθεί για κάθε κάδο αναδρομικά μέχρι το μέγεθος του κάδου να είναι μικρότερο από το επιθυμητό (64KB). Όταν συμβεί αυτό, μπορεί κάθε κάδος να ταξινομηθεί χρησιμοποιώντας μια αποδοτική μέθοδο ταξινόμησης (π.χ. quicksort).

Στο τέλος θα έχουμε δύο πλήρως ταξινομημένες σχέσεις ως προς το κλειδί της ζεύξης (join key). Για να πάρουμε τα τελικά αποτελέσματα, θα πρέπει να σαρώσουμε παράλληλα τα δεδομένα των δύο σχέσεων και να εξάγουμε τα τελικά αποτελέσματα.

Επειδή δεν ξερουμε εκ των προτέρων τον όγκο των αποτελεσμάτων, τα αποτελέσματα της ζεύξης θα γραφτούν σε μια νέα δομή η οποία έχει τη μορφή λίστας. Κάθε κάδος της λίστας θα κρατάει έναν buffer με μέγεθος 1MB και έναν δείκτη στο επόμενο bucket. Η λογική είναι πως γράφουμε δεδομένα σε έναν buffer. Μόλις ο buffer γεμίσει τότε θα εισάγουμε έναν καινούριο κάδο στην λίστα κ.ο.κ.. Η δομή αυτή θα έχει την μορφή που φαίνεται στο πιο κάτω σχήμα:



Πρότυπα συναρτησεων και δομών

Πιο κάτω δίνονται ενδεικτικοί ορισμοί των συναρτήσεων και των βασικών δομών. Μπορείτε να χρησιμοποιήσετε δικούς σας ορισμούς αν θέλετε.

```
/** Type definition for a tuple */
struct tuple {
        int64_t key;
        Int64_t payload;
};

/**
 * Type definition for a relation.  It consists of an array of tuples and a size of the relation.
 */
struct relation {
  tuple  *tuples;
  uint64_t num_tuples;
};

/**
 * Type definition for a relation.  It consists of an array of tuples and a size of the relation.
 */
struct result {
 ...
};

/** Sort Merge Join**/
result* SortMergeJoin(relation *relR, relation *relS);
```

Παράδοση εργασίας

Η εργασία είναι ομαδική, **2 ή 3 ατόμων**.
Προθεσμία παράδοσης: 05/11/2019.
Γλώσσα υλοποίησης: C / C++ χωρίς χρήση stl.
Περιβάλλον υλοποίησης: Linux (gcc 5.4+).

Παραδοτέα: Η παράδοση της εργασίας θα γίνει με βάση το τελευταιο commit πριν την προθεσμία υποβολής στο git repository σας. **Η χρήση git είναι υποχρεωτική**.
Επιπλέον, εκτός από τον πηγαίο κώδικα, θα παραδώσετε μια σύντομη αναφορά, με τις σχεδιαστικές σας επιλογές καθώς και να εφαρμόσετε ελέγχους ως προς την ορθότητα του λογισμικού με τη χρήση ανάλογων βιβλιοθηκών (Software testing).