ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Ανάπτυξη Λογισμικού για Πληροφοριακά Συστήματα

- Αλγόριθμος Radix Hash Join -

Καλοπίσης Ιωάννης - Μυστιλόγλου Θεόδωρος
1115201500059 - 1115201500107
5 Νοεμβρίου 2018

Περιεχόμενα

1 Μεταγλώττιση - Εκτέλεση

Το Makefile του προγράμματος βρίσκεται στο φάκελο του Project. Υπάρχουν οι εξής οδηγίες make στο Makefile:

- **make random**. Κατασκευάζει το εκτελέσιμο που παράγει τα input αρχεία του κεντρικού προγράμματος.
- **make main**. Κατασκευάζει το κύριο εκτελέσιμο που εκτελεί την Radix Hash Join μεταξύ 2 στηλών πινάκων.
- **make unit**. Κατασκευάζει τα 4 εκτελέσιμα αρχεία για τα unit testing του κύριου προγράμματος.

Τα εκτελέσιμα δημιουργούνται στον ίδιο φάκελο με το Makefile. Η εκτέλεση των προγραμμάτων γίνεται με τον εξής τρόπο:

- **./random rows1 columns1 rows2 columns2**. Το πρόγραμμα που φτιάχνει τα test αρχεία παίρνει σαν είσοδο τις γραμμές και τις στήλες του πρώτου αρχείου και έπειτα τις γραμμές και τις στήλες του δεύτερου αρχείου και παράγει τα αρχεία R_file.txt και S_file.txt .
- **./main**. Με αυτή την εντολή εκτελείται το κύριο πρόγραμμα. Δε χρειάζεται παραμέτρους γιατί έχουμε βάλει hardcoded τα αρχεία εισόδου και το ποια στήλη θα κάνει join και αφού στα πλαίσια αυτού του κομματιού της εργασίας δε χρειάζεται να δίνονται σαν είσοδοι.
- Οι παρακάτω εντολές εκτελούν τα αντίστοιχα test προγράμματα των ξεχωριστών αρχείων:
  1. **./unit_general_functions**
  2. **./unit_relations**
  3. **./unit_results**
  4. **./unit_rhj**

Επίσης κατασκευάσαμε και ένα βοηθητικό bash script ονόματει simularityCheck.sh το οποίο ελέγχει έαν 2 τετριμμένα στοιχεία αρχείων είναι ίσα. Βρίσκεται στον ίδιο φάκελο με τα υπόλοιπα εκτελέσιμα και καλείται ως εξής: **./simularityCheck.sh file1.txt rowId1 colJoin1 file2.txt rowId2 colJoin2**.

2 Μορφή και Διάβασμα Αρχείων

Έχουμε φτιάξει το test αρχείο **random_number_generator.c** που φτιάχνει τα δύο αρχεία εισόδου που χρειαζόμαστε. Το εκτελέσιμο ./random παίρνει σαν είσοδο τον αριθμό γραμμών και στηλών του κάθε αρχείου που θα φτιάξει και παράγει τα αντίστοιχα αρχεία με τυχαίους αριθμούς από το διάστημα [0, 4.294.967.295]. Τα αρχεία αυτά θεωρούμε ότι είναι αποθηκευμένα κατά στήλες, δηλαδή η κάθε γραμμή του αρχείου είναι η στήλη/σχέση της ”βάσης δεδομένων”. Έτσι το διάβασμα του αρχειου είναι πολύ πιο γρήγορο.

3 Δομές Δεδομένων και Οργάνωση Αρχείων

Όσον αφορά την **Οργάνωση των Αρχείων** έχουμε τον φάκελο include στον οποίο περιέχονται όλα τα .h αρχεία μας, τον φάκελο test_txts που έχουμε μερικά αρχεία .txt για τα unit test και τον φάκελο src που περιέχει έναν φάκελο με όλα τα αντικειμενικά αρχεία, έναν φάκελο με τα .c αρχεία των unit tests και έναν με το αρχείο που χρησιμοποιούμε για την δημιουργεία των αρχείων εισόδου. Τον αλγόριθμο Radix Hash Join τον έχουμε χωρίσει σε 3 στάδια (RHJ_stage1, RHJ_stage2, RHJ_stage3), όπως περιγράφονται και στην εκφώνηση, και καλούμε τα αντίστοιχα αρχεία μέσα από το αρχείο RHJ.c το οποίο μας δίνει τα τελικά αποτελέσματα. Ακόμα υπάρχει το αρχείο error_check.h που έχει μέσα μερικές defined συναρτήσεις για τον έλεγχο λαθών (π.χ. malloc, λάθος τιμές κ.λπ.) και το αρχείο hash_functions.h με defined τις hash συναρτήσεις και χρήσιμα μεγέθη για τα buckets. Το αρχείο general_functions.c περιέχει συναρτήσεις για το διάβασμα των αρχείων εισόδου, των μεγεθών τους, την αρχικοποοίηση και δέσμευση πινάκων και την καταστροφή τους.
Έχουμε χρησιμοποιήσει τις παρακάτω **Δομές** για την ανάπτυξη και οργάνωση του κώδικα:

- **Relation**. Αποθηκεύουμε τη στήλη που θα γίνει join και το rowId της ”γραμμής” του κάθε στοιχείου.
- **Results**. Η λίστα αποτελεσμάτων. Για τη λίστα αυτή χρησιμοποιούμε και έναν κόμβο κεφαλή για αποθήκευση του αριθμού των ζευγαριών που χωράνε σε κάθε buffer μεγέθους 1 MB και για γρήγορη πρόσβαση στον τελευταίο κόμβο.

4 Συμβάσεις

- Στα αρχεία προς ανάγνωση θεωρούμε πως τα δεδομένα είναι unsigned int των 32 bits αποθηκευμένα κατά στήλες και είναι tab seperated.
- Τα ονόματα των αρχείων προς ανάγνωση και ο αριθμός στήλης που επιλέγεται για Join για το κάθε αρχείο είναι hardcoded και ορισμένα στην main.
- Όταν ο αριθμός στήλης που επιλέγεται για Join είναι μεγαλύτερος από το πλήθος των στηλών τότε επιλέγεται ως default η τελευταία στήλη.
- Για την κατασκευή του ευρετηρίου επιλέγουμε το μικρότερο από τα 2 Relations. Σε περίπτωση που τα 2 Relations έχουν ίδιο μέγεθος θεωρούμε ”μικρότερο” αυτό που δώσαμε ως πρώτο όρισμα στη συνάρτηση RadixHashJoin.
- Στα Result Tuples το rowId1 αναφέρεται στον μικρότερο από τα 2 Relations και το rowId2 στο άλλο Relation. Η περίπτωση ισομεγεθών Relation αναλύεται στο παραπάνω Bullet.
- Τις hash functions τις ελέγξαμε πειραματικά στο σύστημά μας με επεξεργαστή Intel(R) Core(TM) i5-3320M CPU 2.60GHz και L3 cache 3072K

5 Unit Testing

Όπως ζητήθηκε και στο μάθημα εφαρμόσαμε την τεχνική του **unit testing** για να ελέγξουμε την ορθότητα του κώδικά μας. Έτσι χρησιμοποιήσαμε το framework **CUnit** για να μπορέσουμε να κάνουμε καλύτερο έλεγχο του κώδικα. Για κάθε αρχείο .h που φτιάξαμε, το οποίο περιέχει

συναρτήσεις που συνολικά επιτελούν ένα κομμάτι της εργασίας, δημιουργήσαμε ένα αρχείο με πολλά test για κάθε συνάρτηση του .h.