ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικό και Καποδιστριακό
Πανεπιστήμιο Αθηνών

Ανάπτυξη Λογισμικού για Πληροφοριακά Συστήματα

- Join Πολλαπλών Σχέσεων -

Καλοπίσης Ιωάννης - Μυστιλόγλου Θεόδωρος
1115201500059 - 1115201500107
3 Δεκεμβρίου 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 πρώτου παραδοτέου.
- **make part1_main**. Κατασκευάζει το εκτελέσιμο που εκτελεί την Radix Hash Join μεταξύ 2 στηλών πινάκων.
- **make part2_main**. Κατασκευάζει το εκτελέσιμο που διαβάζει τις σχέσεις και εκτελεί τα queries.
- **make unit**. Κατασκευάζει το εκτελέσιμο αρχείο για τα unit testing και το εκτελεί.

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 .
- **./part1_main**. Με αυτή την εντολή εκτελείται το πρώτο παραδοτέο. 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.
- **./part2_main**. Με αυτή την εντολή εκτελείται το δεύτερο παραδοτέο, το οποίο παίρνει στο stdin το αρχείο με τα ονόματα των σχέσεων και στο file descriptor 3 το αρχείο με τα queries. Τέλος βγάζει στο stdout το άθροισμα των γραμμών που ικανοποιούν τα predicates.
- **./unit**. Με αυτή την εντολή εκτελούνται τα unit tests.

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

2.1 Part1_main

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.

2.2 Part2_main

Τα δεδομένα που δεχόμαστε ως είσοδο είναι αποθηκεύμένα σε binary μορφή στα αρχεία που διαβάζουμε.
Το format των δεδομένων είναι: uint64_t numTuples|uint64_t numColumns|uint64_t T0C0|uint64_t T1C0|..|uint64_t TnC0|uint64_t T0C1|..|uint64_t TnC1|..|uint64_t TnCm, χωρίς το σύμβολο της σωλήνωσης ”|”.
Το format των query είναι π.χ.: ”0 2 4|0.1=1.2&1.0=2.1&0.1>3000|0.0 1.1, όπου το πρώτο μέρος είναι οι σχέσεις που χρειάζονται, το δεύτερο μέρος είναι τα predicates (join και φίλτρα)(εδώ το 0 αναφέρεται στη πρώτη σχέση, το 1 στη δεύτερη κ.λπ.) και το τρίτο μέρος αναφέρατει στα αποτελέσματα/προβολές.

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 του part1, των μεγεθών τους, την δέσμευση και αρχικοποοίηση πινάκων, την καταστροφή τους και βοηθητικές συναρτήσεις για το part2. Το αρχείο SQL_queryParser.c περιέχει τις συναρτήσεις για την επεξεργασία και εκτέλεση των 3 πεδίων του query.
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. Άμα υπάρχουν ενδιάμεσα αποτελέσματα που περιέχουν το relation που θα κάνουμε join τότε αποθηκεύουμε και τα rowIds των υπολοίπων ενδιάμεσων αποτελεσμάτων που σχετίζονται με αυτό.

- **Results**. Η λίστα αποτελεσμάτων. Επιστρέφει τα ζευγάρια των rowIds που έγιναν match κατά το join. Αν υπήρχαν ενδιάμεσα αποτελέσματα επιστρέφονται και τα rowIds των υπολοίπων relation από τις γραμμές που έγιναν match, για κάθε έναν από τους πίνακες που έγιναν join. 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.

- **Predicates**. Για κάθε ένα predicate του query φτιάχνουμε ένα κόμβο στη λίστα και το αναλύουμε εκεί. Δίνουμε το κατάλληλο βάρος για να έχει την κατάλληλη προτεραιότητα στην εκτέλεση. Η λίστα έχει και έναν κόμβο κεφαλή που περιέχει δείκτη στο πρώτο και στο τελευταίο στοιχείο και το πόσα predicates υπάρχουν μέσα στη λίστα.

- **Intermediate Results**. Πρόκειται για τη δομή των ενδιάμεσων αποτελεσμάτων. Για κάθε φίλτρο ή join που εκτελείται παράγεται και αποθηκεύεται ένας κόμβος λίστας με τα rowIds

που ικανοποιούν το predicate. Σε περίπτωση που υπάρχει ήδη ενδιάμεσο αποτέλεσμα για ένα relation χρησιμοποιούμε αυτά για είσοδο στο predicate και ενημερώνεται ο κόμβος με τα καινούργια αποτελέσματα. Σε περίπτωση join κρατάμε σε κάθε κόμβο έναν πίνακα δεικτών με όλους τους κόμβους ενδιάμεσων αποτελεσμάτων με τους οποίους έχει γίνει join ο κόμβος.

4 Συμβάσεις

- (part1)Στα αρχεία προς ανάγνωση θεωρούμε πως τα δεδομένα είναι unsigned int των 64 bits αποθηκευμένα κατά στήλες και είναι tab seperated.
- (part1)Τα ονόματα των αρχείων προς ανάγνωση και ο αριθμός στήλης που επιλέγεται για Join για το κάθε αρχείο είναι hardcoded και ορισμένα στην main.
- (part1)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.
- (part2)Θεωρούμε ότι τα πεδία του query είναι έγκυρα και έτσι δε κάνουμε κάποιον περαιτέρω έλεγχο ή εμφάνιση κάποιου μηνύματος σε περίπτωση λάθους.
- (part2)Θεωρούμε ότι πάντα θα εκτελούνται πρώτα τα φίλτρα, αν υπάρχουν στα ερωτήματα.
- 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. Μαζί με την εντολή/target που υπάρχει στο Makefile **make unit** εκτός από το compilation και το linking γίνεται και η εκτέλεση των tests.