Διαχείριση Σύνθετων Δεδομένων

Αναφορά 3ης Εργαστηριακής Άσκησης

ΙΩΑΝΝΗΣ ΜΠΟΥΖΑΣ
ΑΜ:5025

Μέρος 1ο: Containment queries

Μας ζητήθηκε να γράψουμε ένα πρόγραμμα, το οποίο διαβάζει τα δεδομένα από το αρχείο και κατασκευάζει δομές, οι οποίες μπορούν να χρησιμοποιηθούν για την αποτίμηση Containment queries. Σε αυτά τα ερωτήματα, οι συναλλαγές θεωρούνται ότι είναι σύνολα (set semantics), η ερώτηση είναι κι αυτή ένα σύνολο και το ζητούμενο είναι να βρεθούν οι συναλλαγές που περιέχουν όλα τα αντικείμενα της ερώτησης

Αρχικά, δημιουργούμε την μέθοδο ***load_transactions(file_path)*** η οποία φορτώνει και επεξεργάζεται κάθε συναλλαγή και στοιχείο συναλλαγής. Συγκεκριμένα κάθε γραμμή αναπαριστά μια συναλλαγή σε μια μορφή σετ για να αποφύγουμε τα διπλότυπα που σε μία συναλλαγή είναι δυνατόν το ίδιο αντικείμενο να υπάρχει πολλές φορές. Έπειτα η μέθοδος κατασκευάζει τα παρακάτω αρχεία τα οποία θα χρησιμοποιηθούν στην συνέχεια από την εκάστοτε μέθοδο.

- **Transactions list**: Αποθηκεύει κάθε συναλλαγή σαν ένα σετ στην λίστα
- **Signature File**: Μια bitmap αναπαράσταση κάθε συναλλαγής. Αρχικοποιεί bitmap_int = 0 για κάθε συναλλαγή. Για κάθε αντικείμενο στην συναλλαγή θέτει το αντίστοιχο bit σε 1 χρησιμοποιώντας bitwise OR. Αυτό δημιουργεί ένα bitmap όπου το αντικείμενο i υποδεικνύεται από το i-οστό bit που τίθεται σε 1. Τέλος αποθηκεύει το τελικό bitmap για κάθε συναλλαγή
- **Bitslice**: Για κάθε στοιχείο, διατηρεί ένα bitmap όπου το i-οστό bit δείχνει αν το στοιχείο εμφανίζεται στη συναλλαγή i. Αν το στοιχείο δεν υπάρχει στο bitslice αρχικοποιεί ένα bitmap για νέα στοιχεία και θέτει το bit που αντιστοιχεί στο τρέχον id της συναλλαγής
- **Inverted Index**: Αντιστοιχεί κάθε στοιχείο συναλλαγής σε μια λίστα από transaction IDs όπου εμφανίζεται αυτό το στοιχείο

Naïve method

Η baseline μέθοδος μας. Ελέγχει άμεσα αν το σύνολο ερωτημάτων είναι υποσύνολο κάθε συναλλαγής. Η πολυπλοκότητα της είναι O(n×m) όπου n είναι ο αριθμός των συναλλαγών και m είναι το μέσο μέγεθος των συναλλαγών. Απλή αλλά αναποτελεσματική για μεγάλα σύνολα δεδομένων.

Signature File Method

Αυτή η μέθοδος δημιουργεί ένα bitmap για το ερώτημα συνδυάζοντας τα bit κάθε στοιχείου μέσω λογικής πράξης OR.

Κάθε bit του query_bitmap αντιστοιχεί στην παρουσία ενός στοιχείου στο ερώτημα.

Για κάθε συναλλαγή, γίνεται bitwise AND μεταξύ του bitmap της συναλλαγής και του bitmap του ερωτήματος.

Αν το αποτέλεσμα της πράξης ισούται με το query_bitmap, σημαίνει ότι όλα τα στοιχεία του ερωτήματος υπάρχουν στη συγκεκριμένη συναλλαγή.

Οι συναλλαγές που ικανοποιούν τις συνθήκες μας προστίθενται στη λίστα αποτελεσμάτων.

Bitsliced Signature Method

Η μέθοδος αποτίμησης υπολογίζει το λογικό AND των bitmaps που αντιστοιχούν στα αντικείμενα που περιλαμβάνονται στην ερώτηση και μετά βρίσκει τις συναλλαγές που αντιστοιχούν στις θέσεις όπου το bitmap που προκύπτει έχει 1 και τις προσθέτει στο αποτέλεσμα. Αρχικά, ελέγχει αν το ερώτημα είναι κενό και επιστρέφει κενή λίστα αν είναι. Ορίζει το result_bitmap με το bitmap του πρώτου στοιχείου του ερωτήματος.

Για κάθε επόμενο στοιχείο του ερωτήματος:

- Εάν υπάρχει στο bitslice, γίνεται λογική ΚΑΙ (AND) με το υπάρχον result_bitmap.
- Εάν δεν υπάρχει, επιστρέφεται αμέσως κενή λίστα, διότι το στοιχείο δεν υπάρχει σε καμία συναλλαγή.

Στη συνέχεια, το αποτέλεσμα του bitmap μετατρέπεται σε λίστα αναγνωριστικών συναλλαγών (results).

Μέσα σε ένα βρόχο while, ελέγχεται κάθε bit του bitmap:

- Εάν το bit είναι ενεργό (1), το τρέχον tid (transaction ID) προστίθεται στη λίστα αποτελεσμάτων.
- Το bitmap μετατοπίζεται δεξιά κατά 1 και το tid αυξάνεται.

Αξίζει να σημειωθεί ότι με αυτή την προσέγγιση για βρούμε τις συναλλαγές που αντιστοιχούν στις θέσεις όπου το bitmap που προκύπτει έχει 1 και να τις προσθέσουμε στο αποτέλεσμα ο χρόνος που παίρνει για να γίνει αυτό είναι μεγαλύτερος από την naïve πράγμα που δεν είναι και πολύ λογικό.

Για αυτό μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο του Brian Kernighan. Αυτός ο αλγόριθμος είναι ιδιαίτερα αποδοτικός γιατί εντοπίζει απευθείας το λιγότερο σημαντικό ενεργό bit (least significant bit - LSB) και το καθαρίζει, μειώνοντας έτσι τον αριθμό των

επαναλήψεων στον βρόχο. Έχει πολυπλοκότητα ανάλογη με τον αριθμό των ενεργών bits και όχι με το συνολικό αριθμό θέσεων bit. Είναι πολύ χρήσιμος σε περιπτώσεις όπου το bitmap έχει λίγα ενεργά bits (sparse bitmap).

Inverted Index Method

Αυτή η μέθοδος αποτίμησης υπολογίζει την τομή των λιστών που αντιστοιχούν στα αντικείμενα που περιλαμβάνονται στην ερώτηση (με χρήση merge αλγορίθμου για sorted lists) και προσθέτει τις συναλλαγές που συμπεριλαμβάνονται στην τομή στο αποτέλεσμα. Πιο συγκεκριμένα:

- Αρχικά ελέγχεται αν το ερώτημα είναι κενό ή περιέχει στοιχεία που δεν υπάρχουν στο ανεστραμμένο ευρετήριο (inverted index). Σε αυτές τις περιπτώσεις, επιστρέφεται κενή λίστα.
- Για κάθε στοιχείο στο inverted, γίνεται ταξινόμηση της αντίστοιχης λίστας συναλλαγών για να διασφαλιστεί σωστή λειτουργία της επόμενης διαδικασίας σύγκρισης.
- Τα στοιχεία του ερωτήματος ταξινομούνται σύμφωνα με το μήκος της λίστας τους (από το μικρότερο προς το μεγαλύτερο), με στόχο τη μείωση του κόστους της επαναλαμβανόμενης σύγκρισης.
- Η σύγκριση των λιστών πραγματοποιείται με χρήση της συνάρτησης merge_intersection, η οποία συγκρίνει δύο ταξινομημένες λίστες και επιστρέφει την τομή τους.

merge_intersection(list1, list2)

Αυτή η συνάρτηση εκμεταλλεύεται το γεγονός ότι οι λίστες είναι ταξινομημένες και συγκρίνει τα στοιχεία ένα προς ένα, όπως γίνεται σε συγχώνευση (merge) σε merge sort.

Ο χρόνος εκτέλεσης είναι γραμμικός ως προς το άθροισμα των μηκών των δύο λιστών: O(len(list1) + len(list2).

Είναι ιδανική για μεγάλα σύνολα δεδομένων με σχετικά μικρή τομή.

Main

Στην main πολύ απλα παίρνουμε από τον χρήστη τα δεδομένα που θα εισάγει στο terminal και ανάλογα με το qnum και το method θα εκτελέσουμε τις αντίστοιχες μεθόδους μας για τα αντίστοιχα queries μας

Μέρος 2ο: Relevance queries

Αυτή την φορά γράψαμε ένα πρόγραμμα, το οποίο διαβάζει τα δεδομένα από το αρχείο και κατασκευάζει μία δομή inverted file η οποία εξυπηρετεί ερωτήσεις σχετικότητας (relevance queries). Λάβαμε υπόψη (α) το γεγονός ότι ένα αντικείμενο μπορεί να εμφανίζεται πολλές φορές σε μία 3 συναλλαγή και (β) τη σπανιότητα εμφάνισης των αντικειμένων. Έστω ότι T είναι το σύνολο των συναλλαγών και $|T|$ είναι ο πληθάριθμός τους. Για μία συναλλαγή $\tau \in T$ και μία ερώτηση q που αποτελείται από ένα σύνολο αντικειμένων, η σχετικότητα της τ με την q ορίζεται ως:

$rel(\tau, q) = \Sigma i \in q \left(occ(i, \tau) \cdot \frac{|T|}{trf(i,T)} \right)$, όπου $occ(i, \tau)$ είναι το πόσες φορές εμφανίζεται το αντικείμενο i στη συναλλαγή τ, και $trf(i, T) = |\{\tau : \tau \in T \wedge i \in \tau\}|$ είναι το πόσες συναλλαγές στο σύνολο T περιλαμβάνουν το αντικείμενο i. Στόχος είναι να υπολογιστούν οι συναλλαγές τ με $rel(\tau, q) > 0$ σε φθίνουσα σειρά με βάση το $rel(\tau, q)$

Πιο συγκεκριμένα:

Η συνάρτηση αυτή **load_transactions(file_path)** πλέον έχει σαν στόχο και τη δημιουργία του αντεστραμμένου αρχείου (inverted file) και τον υπολογισμό των τιμών IDF (Inverse Document Frequency) για κάθε αντικείμενο:

- **Ανάγνωση Συναλλαγών:**
    - Κάθε γραμμή του αρχείου αναπαριστά μία συναλλαγή, η οποία περιέχει μία λίστα αντικειμένων.
    - Οι γραμμές μετατρέπονται από κείμενο σε λίστα με χρήση της eval() και αποθηκεύονται στη λίστα transactions.
- **Κατασκευή Αντιστραμμένου Αρχείου:**
    - Για κάθε αντικείμενο σε κάθε συναλλαγή αποθηκεύεται το tid (transaction id) και ο αριθμός εμφανίσεων του αντικειμένου στην εν λόγω συναλλαγή.
    - Παράλληλα υπολογίζεται και ο αριθμός συναλλαγών στις οποίες εμφανίζεται το κάθε αντικείμενο (item_transaction_count).

- **Υπολογισμός IDF:**
    - Για κάθε αντικείμενο, υπολογίζεται η τιμή IDF
- **Αποθήκευση σε αρχείο:**
    - Οι πληροφορίες αποθηκεύονται στο αρχείο invfileocc.txt

Inverted Method

Η συνάρτηση αυτή υλοποιεί αναζήτηση σχετικότητας με χρήση του αντεστραμμένου αρχείου και των τιμών IDF.

- **Φιλτράρισμα Ερώτησης:**
    - Αφαιρούνται όσα αντικείμενα δεν υπάρχουν στο αντεστραμμένο αρχείο.
- **Ταξινόμηση Αντικειμένων Ερώτησης:**
    - Ταξινόμηση βάσει του μήκους των λιστών τους στο inverted file (για βελτίωση απόδοσης στη συγχώνευση).
- **Υπολογισμός Σκορ Σχετικότητας:**
    - Αρχικά υπολογίζονται τα σκορ για το πρώτο αντικείμενο.
    - Στη συνέχεια συγχωνεύονται επαναληπτικά οι λίστες κάθε αντικειμένου με τις ήδη υπολογισμένες τιμές.
- **Συγχώνευση (Merge):**
    - Για κάθε κοινό transaction id, τα σκορ αθροίζονται.
    - Αν δεν υπάρχει το tid στη μία από τις δύο λίστες, αντιγράφεται αυτούσιο.
- **Επιστροφή Αποτελεσμάτων:**
    - Τα αποτελέσματα ταξινομούνται φθίνουσα με βάση τη σχετικότητα και επιστρέφονται τα k κορυφαία.

Naïve Method

Η συνάρτηση αυτή υλοποιεί την naïve προσέγγιση της αναζήτησης σχετικότητας, χωρίς χρήση του αντεστραμμένου αρχείου.

- **Υπολογισμός Σκορ για Κάθε Συναλλαγή:**
    - Για κάθε συναλλαγή, υπολογίζεται το σκορ της βάσει του πλήθους εμφανίσεων των αντικειμένων της ερώτησης, σταθμισμένο με το IDF.
- **Φιλτράρισμα Μη Σχετικών Συναλλαγών:**

    - Αν η βαθμολογία μιας συναλλαγής είναι 0, αγνοείται.
- **Ταξινόμηση & Επιστροφή:**
    - Οι συναλλαγές ταξινομούνται κατά φθίνουσα σχετικότητα και επιστρέφονται τα k κορυφαία αποτελέσματα.

Main

Στην main ο χρήστης εισάγει τα στοιχεία που πρέπει και ανάλογα εμείς εκτελούμε τις μεθόδους αποτίμησης που πρέπει , για τα queries που θελουμε και για τα κ σχετικότερα αποτελέσματα που θέλουμε.