Εργαστηριακή Ομάδα *52*

Γιουμερτάκης Απόστολος, 2017030142
Κατσούπης Ευάγγελος, 2017030077

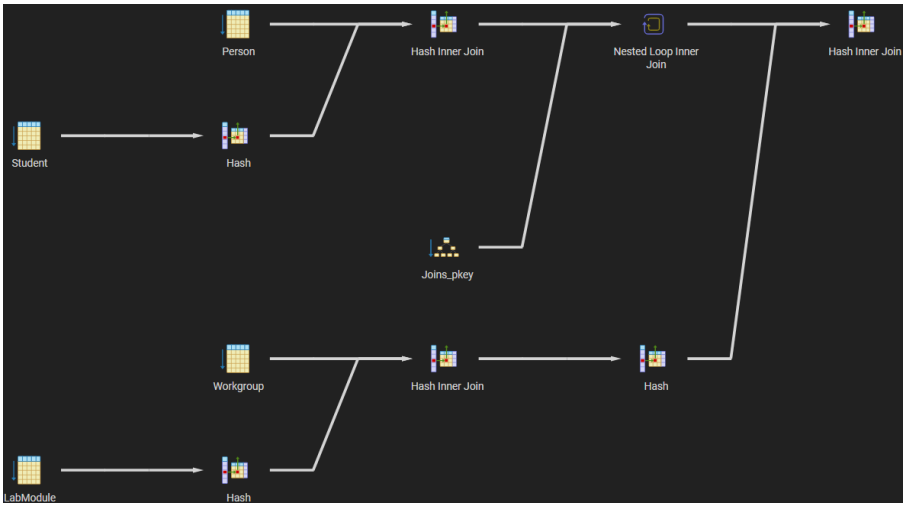
Βρες τα ονοματεπώνυμα φοιτητών με ημερομηνία εισαγωγής (entry_date) μετά την $1-09-2010$ και πριν την $1-09-2011$ οι οποίοι έχουν συμμετάσχει σε ομάδες για την εκπόνηση εργαστηριακής εργασίας τύπου project με βαθμό μεγαλύτερο του 8.

Hash Join Enabled

Unoptimized Query

Αρχικά θα αναλυθεί χρονικά το παραπάνω αίτημα χωρίς κάποια επεξεργασία - βελτιστοποίηση. Ο συνολικός μέσος χρόνος περάτωσης του αιτήματος είναι περίπου $\underline{235ms}$.

Γραφικά το πλάνο που επιλέγει ο compiler:



Σχήμα 1: Unoptimized compilation with hashjoin enabled

Από κάτω προς τα πάνω τα βήματα:

- Sequential Scan (SS) στον πίνακα LabModule ώστε να παρθούν τα tuples όπου έχουν καταχωρηθεί με τύπο project. Από τις 2000 διαφορετικές καταχωρίσεις, με την χρήση του παραπάνω φίλτρου, επιστρέφονται μόλις 345 γραμμές και σε μέσο μέγιστο χρόνο 0.4 ms
- Δημιουργία ενός hash table (HT) για τα παραπάνω δεδομένα ώστε να συνενωθούν (join) αργότερα. Ο χρόνος δημιουργίας του ειναι 0.31 ms.
- Σειριακή σάρωση (SS) στον πίνακα Workgroup ώστε να παρθούν οι εργαστηριακές ομάδες όπου έχουν βαθμό μεγαλύτερο του 8. Από τις $376K$ διαφορετικές καταχωρίσεις, με την χρήση φίλτρου, επιστρέφονται 209 γραμμές και σε μέσο μέγιστο χρόνο τα 134 ms
- Inner hash join στα 2 προηγούμενα HT, με κριτήριο το ίδιο module_no. Ο μέσος χρόνος αναζήτησης είναι τα 122 ms και με τελικά 32 tuples για να ελεγχθούν αργότερα.
- HT στα παραπάνω δεδομένα, με μεσο χρόνο τα $122ms$.

Σε αυτό το σημείο, έχοντας δημιουργηθεί το παραπάνω sub-query θα επιστραφούν τα tuples όπου θα εμπεριέχουν τις ομάδες με τους φοιτητές αναφοράς, ώστε να γίνουν join με τα παραπάνω workgroups.

- SS στον πίνακα Students ώστε να γίνουν fetch οι φοιτητές που έχουν εισαχθεί μεταξύ των ημερομηνιών που μας ενδιαφέρουν. Από τις $54K$ διαφορετικές καταχωρίσεις, με την χρήση φίλτρου, επιστρέφονται $4K$ γραμμές και σε μέσο μέγιστο χρόνο τα 10.5 ms

- Δημιουργία HT στα παραπάνω, για join αργότερα. Ο χρόνος δημιουργίας του ειναι στα 11.30 ms.
- **(A)** Index SS με χρήση του πρωτεύοντος κλειδιού (το επέλεξε ο compiler), στον πίνακα "Joins" ώστε στο τέλος να έχουμε μόνο τους φοιτητές που έχου συμμετάσχει σε εργαστήριο. Ο μέσος, μέγιστος χρόνος εκτέλεσης είναι τα 0.022 ms. Λόγω του ότι το κάθε tuple είναι και primary key, επιλέχθηκε και σαν index.
- SS στον πίνακα Person για να παρθούν τα στοιχεία των φοιτητών με τα ΑΜΚΑ που μας ενδιαφέρουν. Ο μέσο μέγιστος χρόνος είναι τα 7 ms.
- Inner hash Join των αμκα των φοιτητών με τα στοιχεία από το Person για να πάρουμε τα στοιχεία του κάθε ένα από αυτούς. Ο χρόνος για την δημιουργία των tuples είναι 35.5 ms.
- Nested Loop μεταξύ των αμκα των φοιτητών και των ομαδών του Joins για να παρθούν οι ομάδες που εμπεριέχουν τους παραπάνω φοιτητές. Ο μέσος μέγιστος χρόνος αναζήτησης είναι τα 190 ms. Για κάθε row από το **(A)**, υπολογίζονται ποιες ομάδες θέλουμε να πάρουμε.

Τέλος πραγματοποιείται inner hash join μεταξύ των παραπάνω sub queries με κριτήριο να έχουν ίδιο module_no. Ο μέγιστος μέσος χρόνος για να βρεθούν οι ομάδες αλλά και να ολοκληρωθεί το αίτημα έιναι στα $250 - 300$ ms (ανάλογα με τον επεξεργαστή)

Optimized Query

Για την κατανόηση του αιτήματος θα δούμε το ερώτημα από το "τέλος". Με αυτό τον τρόπο θα σπάσουμε το αίτημα σε υποπροβλήματα ώστε να κατανοηθεί ο τρόπος αναζήτησης και να γίνει μια προσπάθεια βελτιστοποίησης.

- Για την αναζήτηση πάνω στον πίνακα "Labmodule" θα κατασκευαστεί ευρετήριο (index) στον πίνακα LabModule, χρησιμοποιώντας ένα δυαδικό δένδρο (btree) ως προς το attribute 'type' (lab_exercise, project). Λόγω της μεγάλης επιλεξιμότητας του τύπου, φιλτράρονται οι πλειάδες του LabModule, σύμφωνα με τον τύπο της εργασίας. Η παραπάνω διαδικασία μπορεί να βελτιστοποιηθεί περαιτέρω, συσταδοποιώντας (clustering) τα στοιχεία του πίνακα σύμφωνα με το παραπάνω ευρετήριο.

  Λόγω του ότι το ερώτημα έχει ως φίλτρο ισότητα (point query) και όχι εύρος, η χρήση hash index "πάνω" στον τύπο, θα καταλήξει σε ελαφρώς καλύτερο χρόνο (0.125 ms) σε σχέση με την σειριακή αναζήτηση ($0,170$ ms), αλλά σε αρκετά χειρότερο σε σχέση με την χρήση btree.

  Ο μέσος μέγιστος χρόνος για την ανάκτηση των εργαστηριακών εργασιών τύπου project, μετά την παραπάνω βελτιστοποίηση, είναι $\sim$ **0.07 ms** από 0.17 ms που χρειάζόνταν αρχικά.

  Κατασκευάζοντας ευρετήριο για κάποιο άλλο γνώρισμα (attribute) του πίνακα LabModule, δεν θα δούμε καλύτερα αποτελέσματα μιας και δεν έχουν πληροφορία που να σχετίζεται με το φίλτρο σε αυτή την περίπτωση.

- Για τον πίνακα WorkGroup, όπου θέλουμε να τις ομάδες με βαθμό > 8, όμοια, θα γίνει χρήση ευρετηρίου με δέντρο, πάνω στο γνώρισμα grade . Η αναζήτηση ομάδων (workgroups) χωρίς κάποιο ευρετήριο, θα ολοκληρωθεί σε μέσο χρόνο ~ 53 ms.

  Με την δημιουργία ενός δυαδικού δένδρου (btree) ως προς τον βαθμό αλλά και με clustering αυτού, παίρνουμε τα δεδομένα σε περίπου **0.5** ms, χρόνος αρκετά μικρότερος από το αρχικά απαιτούμενο. Και σε αυτή την περίπτωση δεν θα γίνει χρήση hash indexing (~ 50 ms) αλλά και ούτε ευρετηρίων σε άλλα attributes.

- Σε αυτό το σημείο μπορούμε να βρούμε ποιά είναι τα εργαστήρια τα οποία ικανοποιούν τους δυο παραπάνω περιορισμούς. Χωρίς την χρήση indexing απαιτούνται περίπου $50ms$, όσο περίπου και ο χρόνος της πιο αργής αναζήτησης. Με την χρήση των δύο παραπάνω ευρεστικών μεθόδων, απαιτούνται μόλις $\sim$ **0,175** ms

Έχοντας, στα χέρια μας τις ομάδες που ικανοποιούν τα παραπάνω κριτήρια μπορούμε να υπολογίσουμε ποιοι είναι οι φοιτητές που ανοίκουν σε αυτές, και με αυτές να υπολογίσουμε ποια είναι τα στοιχεία, αυτών που μας ενδιαφέρουν.

- Για να βρούμε ποιες είναι οι ομάδες, στον πίνακα Joins θα κατασκευαστεί ένα btree index πάνω στο attribute module_no, ακολουθώντας συσταδοποιήση αυτών.

  Χρησιμοποιώντας, το παραπάνω, ο μέσος χρόνος που απαιτείται για το fetch των στοιχείων από τον πίνακα, είναι στα **6** ms.
  Θα μπορούσε να γίνει και χρήση hashing πάνω στο ίδιο attribute, απαιτώντας χρόνους ίδιας τάξης αλλά λίγο χειρότερους, μιας και τα τελικά στοιχεία τα δεν είναι αρκετά (για να καθυστερήσει το sequential scan).
  Στην περίπτωση που δεν χρησιμοποιούσαμε κάποιο index, θα χρειαζόταν περίπου 450 ms.

- Σε αυτό το σημείο, θα χρειαστεί να βρούμε τα αμκα των φοιτητών, για να ελέγξουμε αν ανήκουν στις παραπάνω ομάδες. Όπως και πριν, τα αμκα θα βρεθούν με την χρήση index πάνω στο entry_date και συσταδοποίηση αυτών. Ο χρόνος που απαιτείται είναι τα $0,55$ ms, ενώ με hashing στα 5 ms όσο και ο χρόνος χωρίς την χρήση κάποιου index.

- Στην συνέχεια βρίσκονται τα αμκα τα οποία ειναι και στις παραπάνω ομάδες, από τον πίνακα Join.Ο έλεγχος γίνεται σχεδόν αμέσως μια και χρειάζονται μόλις 0.008 ms.

- Τέλος, με την χρήση indexing με btree παίρνουμε τα στοιχεία του κάθε φοιτητή από τον πίνακα Person και εξάγουμε τα αποτελέσματα. Ο χρόνος τέλεσης είναι όμοια στα 0.008 ms.

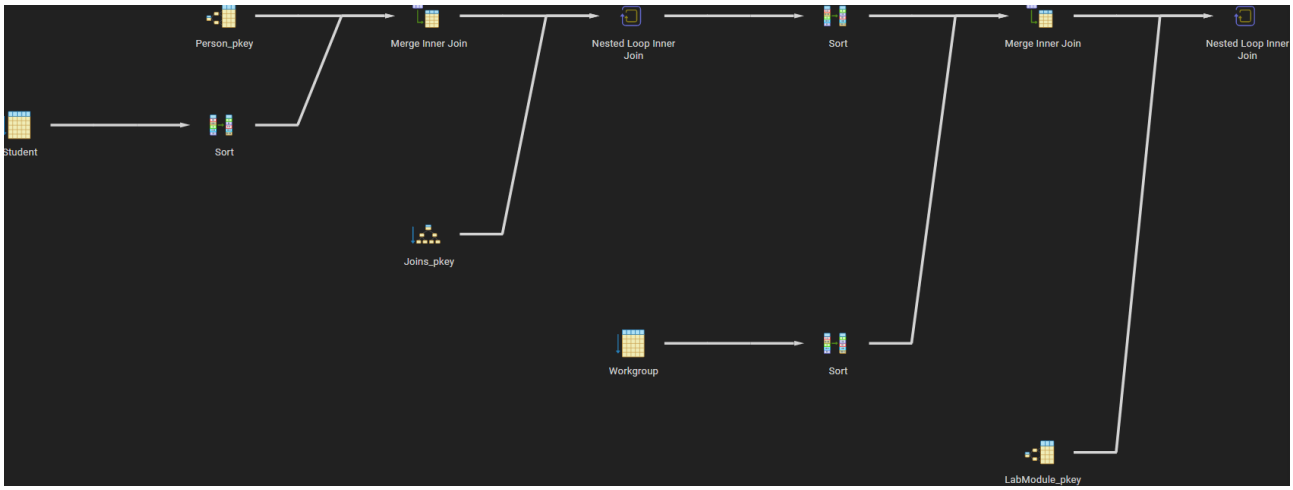
Με την χρήση των παραπάνω, χρειάζονται περίπου 5 ms. Συγκρίνοντας με την αρχική απαίτηση χρόνου, 250 ms, πετύχαμε βελτιστοποίηση $\sim 98\%$. Γραφικά το πλάνο που επιλέγει ο compiler:

Σχήμα 2: Optimized compilation with hashjoin enabled

Hash Join Disabled

Unoptimized Query

Όμοια θα δούμε το αίτημα χωρίς κάποια βελτιστοποίηση. Το πλάνο, που ακολουθεί ο compiler, γραφικά: Όμοια:



Σχήμα 3: Unoptimized compilation with hashjoin disabled

- Index scan (ID) με χρήση του πρωτεύοντος κλειδιού, στον πίνακα Lab Module με χρόνο τα 0.053 ms και για τιμές όπου το module_no είναι το ίδιο με αυτό από τα tuples του πίνακα Join. Ουσιαστικά, για κάθε στοιχείο του δεύτερου πίνακα (θα αναλυθεί στην συνέχεια), γίνεται η παραπάνω διαδικασία.
- Sequential scan στον πίνακα WorkGroup με χρόνο ολοκλήρωσης $\sim 50ms$.
- Ταξινόμηση (Sorting) των στοιχείων που πάρθηκαν από τον πίνακα WorkGroup με βάσει το κλειδί $module\_no, wgID$ με συνολικό χρόνο εκτέλεσης τα $\sim 50ms$.
- ID με χρήση του πρωτεύοντος κλειδιού, στον πίνακα Join με χρόνο τα 0.011 ms και για τιμές όπου το amka είναι το ίδιο με αυτό των φοιτητών αναφοράς (θα αναλυθεί στην συνέχεια). Όμοια, για κάθε amka φοιτητή, θα βρίσκεται σε ποιά ομάδα είναι (wgID), και στην συνέχεια θα ελέγχεται για το αν ο τύπος εργασίας, που συμμετέχει, είναι project.
- SS στον πίνακα Student με χρόνο ολοκλήρωσης τα $\sim 4.5ms$, για τα amka της χρονιάς εισαγωγής που μας ενδιαφέρει. Για κάθε amka θα υπολογίζεται σε ποια ομάδα ανήκει και θα ελέγχεται ο τύπος εργασίας και βαθμολογία.
- Sort των στοιχείων που πάρθηκαν από τον πίνακα Student με βάσει το κλειδί $amka$ με συνολικό χρόνο εκτέλεσης τα $\sim 20ms$.
- ID με χρήση του πρωτεύοντος κλειδιού, στον πίνακα Person με χρόνο τα 8.2 ms και για τιμές όπου το amka είναι το ίδιο με αυτό των φοιτητών αναφοράς.
- Merge Join των στοιχείων του πίνακα Person με τα amka των φοιτητών για να πάρουμε τα ονοματεπώνυμα αυτών. Ο χρόνος εκτέλεσης είναι τα 48 ms περίπου.
- Ο συνολικός χρόνος για να βρεθούν οι ομάδες, από τον πίνακα Join όπου ανήκει ο κάθε φοιτητής, είνα τα 117 ms περίπου. Μπορούμε να αντιληφθούμε την γραμμική συσχέτιση που έχει ο χρόνος αυτός, συναρτήσει του πλήθους των φοιτητών, των χρόνων για το sorting των πινάκων αλλά και του χρόνου που χρειάζεται να βρεθεί η κάθε ομάδα.

- Sort των στοιχείων με βάσει το κλειδί $module\_no, wgID$ με συνολικό χρόνο εκτέλεσης τα $\sim 200ms$.
- Αντίστοιχα ο χρόνος για να βρεθούν, ποιές από τις παραπάνω ομάδες έχουν βαθμό μεγαλύτερο του 8, είνα τα 250 ms περίπου. Όμοια, ο έλεγχος γίνεται στα sorted δεδομένα του πίνακα WorkGroup.
- Τέλος υπολογίζεται ποιες ομάδες είναι τύπου project, σε χρόνο ~ 265 ms.

Δεδομένου των παραπάνω, μπορούμε να πάρουμε τα ονοματεπώνυμα των φοιτητών, που τηρούν τις προϋποθέσεις του αιτήματος σε, περίπου, $\sim$**265 ms**, όσο και ο τελικός χρόνος της παραπάνω ανάλυσης. Είναι σημαντικό να κατανοηθεί πως οι παραπάνω χρόνοι δεν προστίθενται μεταξύ τους, μιας και έχουν σημειωθεί μόνο αυτοί για την ολοκλήρωση του κάθε βήματος. Για παράδειγμα, ο υπολογισμός για εργασίες με φοιτητές αναφοράς, που είναι τύπου project ξεκινά στα $\sim 240ms$, δηλαδή στον χρόνο που χρειάστηκε να υπολογιστεί το πρώτο στοιχείο του προηγούμενου βήματος κτλ.

Optimized Query

Αν γίνει χρήση των βελτιστοποιήσεων που χρησιμοποιήθηκαν στην περίπτωση με ενεργοποιημένο hash join, ο χρόνος ολοκλήρωσης είναι πάλι τα **5 ms** και το ποσοστό βελτιστοποίησης, όμοια στα **98%**.

Γραφικά το πλάνο:

Σχήμα 4: Optimized compilation with hash join disabled

Query optimization at more data

Αρχικά προστέθηκαν νέες εργασίες στον πίνακα Labmodule και παράλληλα εργαστηριακές ομάδες στον πίνακα Workgroup. Νέες εισαγωγές έγιναν και στον πίνακα Joins αλλά λιγότερες σε αριθμό μιας και δεν τηρούνται όλλες οι προϋποθέσεις για εγγραφή φοιτητών στα αντίστοιχα μαθήματα για το κάθε πεδίο. Σε αυτό σημείο υπάρχουν 500335 tuples στον πίνακα Labmodule, 2863365 στον πίνακα Workgroup και 1610139 στον πίνακα Joins.

Τα πλάνα που ακολουθούνται και για τις δυο περιπτώσεις hash join, είναι τα ίδια και με χρόνους ολοκλήρωσης:

- hash join enabled: 700 ms.
- hash join disabled: 760 ms

Αν ακολουθήσουμε τις παραπάνω βελτιστοποιήσεις, θα πετύχουμε του ακόλουθους χρόνους:

- hash join enabled: 220 ms.
- hash join disabled: 300 ms

Λόγω του μεγάλου πλήθους των στοιχείων στους πίνακες Labmodule και Joins μπορούμε να εφαρμόσουμε hash indexing, μιας και ελέγχουμε για ισότητα στα δεδομένα.

Με εφαρμογή hash indexing στον πίνακα Joins, ως προς το module_no (μεγάλη απαίτηση χρόνου στο πλάνο), το αίτημα χρειάζεται τον ίδιο χρόνο για να ολοκληρωθεί (220 ms και 300 ms) και για τις δύο περιπτώσεις.

Με την εφαρμογή hash indexing στον πίνακα Labmodule, ως προς το type,κερδίζουμε από 10 ms σε κάθε περίπτωση (210 ms και 290 ms).

- hash join enabled: **210** ms.
- hash join disabled: **290** ms

Οποιαδήποτε άλλη απόπειρα βελτιστοποίησης δεν οδήγησε σε κάλυτερους χρόνους.