ΑΝΑΦΟΡΑ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ 2018

ΟΜΑΔΑ: Ντουσάκης Γρηγόρης , Τσούρρα Άρτ

ΑΜ: 2014030081 , 2014030193

2. **Μελέτη απόδοσης ερωτήσεων – φυσικός σχεδιασμός:**

Α) Αρχικά για να βρούμε τους φοιτητές (αμκα,όνομα.επίθετο) των οποίων το επίθετο τους ανήκει στο διάστημα ‘ΜΑ’ εως ‘ΜΟ’ χρησιμοποιούμε το παρακάτω statement:

```
select amka,name,surname
from "Student"
where 'MA'<= surname and surname < 'ΜΠ';
```

(γράφουμε <’ΜΠ’ διότι θέλουμε να έχουμε ολα τα επώνυμα που ξεκινάνε με ‘ΜΟ’ αλλίως θα γράφαμε <=’ΜΟ’ χωρίς να τα δέχεται)
Έπειτα χρησιμοποιώντας την explain analyze παίρνουμε το παρακάτω αποτέλεσμα (παραθέτω και φωτογραφία):

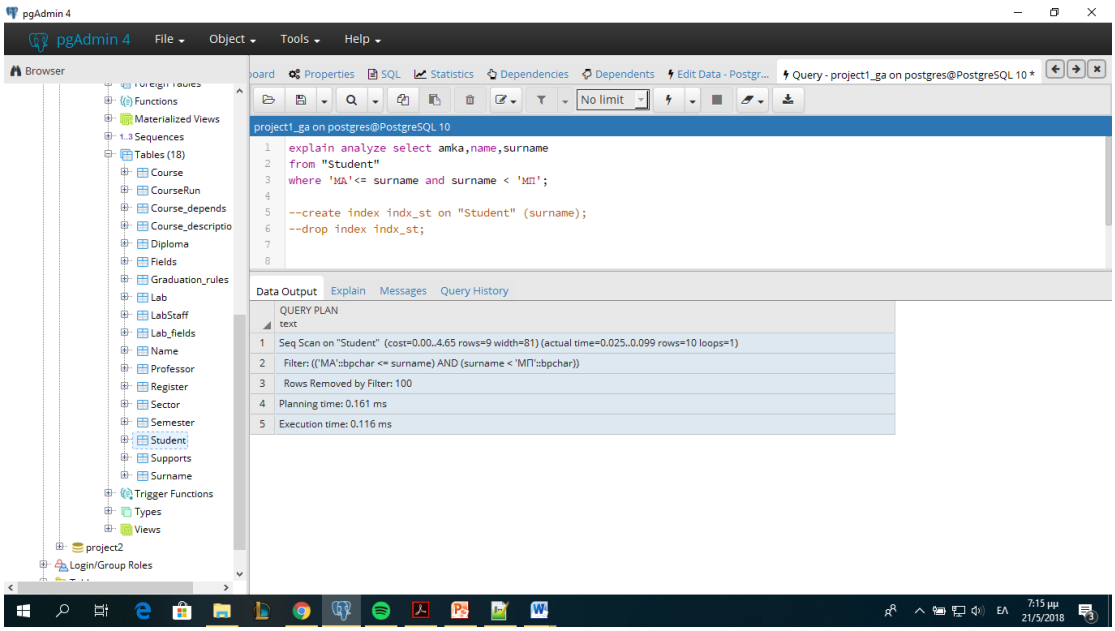
"Seq Scan on "Student" (cost=0.00..4.65 rows=9 width=81) (actual time=0.025..0.099 rows=10 loops=1)"

" Filter: (('MA'::bpchar <= surname) AND (surname < 'ΜΠ'::bpchar))"

" Rows Removed by Filter: 100"

"Planning time: 0.161 ms"

"Execution time: 0.116 ms"



Βλέπουμε ότι στην 1η γραμμή μας εξηγεί ότι το πρόγραμμα κάνει sequential scan, δηλαδή παίρνει με την σειρά τα tuples και τα ελέγχει 1-1, κάτι που είναι απλό σαν ιδέα,κάνει την δουλεία του σε λογικό χρόνο(0.116 ms) αυτό όμως συμβαίνει επειδή έχουμε λίγα δεδομένα(λίγες εκατοντάδες). Σε περίπτωση όμως που η βάση μας είχε πολλά δεδομένα (δεκάδες χιλιάδες ή εκατοντάδες χιλιάδες) τότε θα ήταν αρκετά χρονοβόρο.

Στην περίπτωση λοιπόν που έχουμε πολλά δεδομένα μια λύση για να βελτιστοποιήσουμε την εύρεση των φοιτητών ως προς τον χρόνο είναι να χρησιμοποιήσουμε ευρετήρια. Πολλά συστήματα βάσεων δεδομένων φτιάχνουν αυτόματα ευρετήρια κυρίως σε πρωτεύοντα κλειδιά του πίνακα. Το surname στην δική μας περίπτωση δεν ειναι πρωτεύον κλειδί οπότε θα φτιάξουμε εμείς ενα.

Το επόμενο ερώτημα είναι τι είδους ευρετήριο.Υπάρχουν αρκετά είδη ευρετηρίων αλλά εμείς ασχοληθήκαμε με 2. 1) B+-Tree index το οποίο βασίζεται σε balance trees και μπορεί να χρησιμοποιηθεί για point queries (ένα αποτέλεσμα '=') αλλα χρησιμοποιείται κυρίως για range queries (πολλά αποτελέσματα '>=/<='). Τα B+-Tree indexes έχουν λογαριθμικό κόστος σε I/O. 2) Hash index χρησιμοποιούνται μόνο για point queries και έχουν κόστος σε I/O τυπικά σταθερό.

Επομένως για το συγκεκριμένο ερώτημα θα φτίαξουμε ένα B+-Tree index(indx_st) το οποίο είναι προκαθορισμένο είδος:
create index indx_st on "Student" (surname);

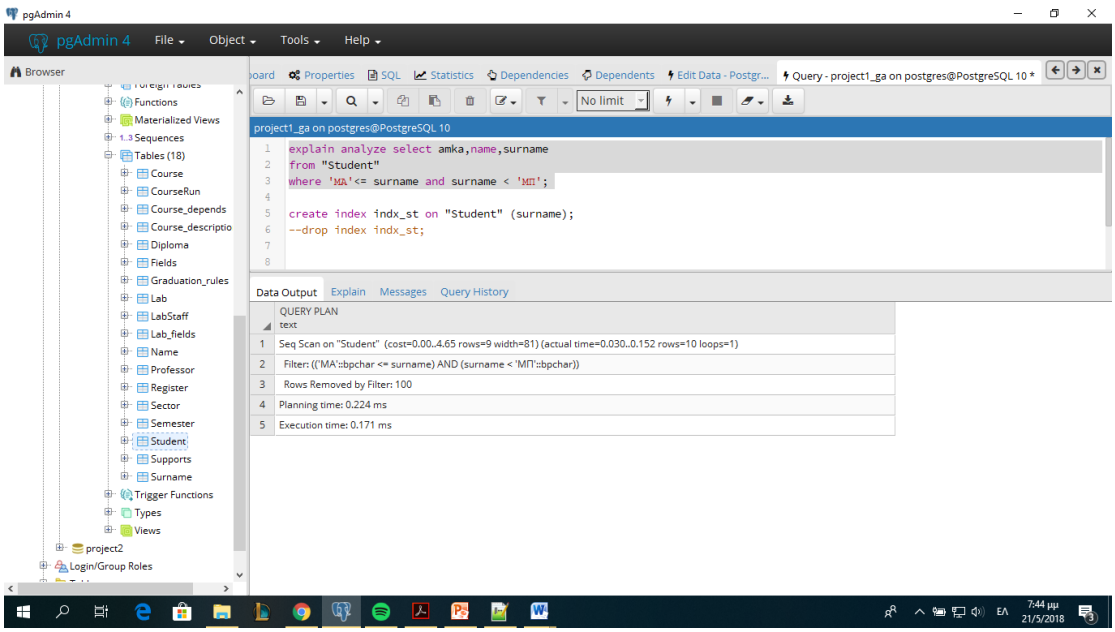
Ξανατρέχουμε λοιπόν την explain analyze αφότου φτιάξαμε το index και παίρνουμε τα εξής αποτελέσματα:
"Seq Scan on "Student" (cost=0.00..4.65 rows=9 width=81) (actual time=0.030..0.152 rows=10 loops=1)"

" Filter: (('MA'::bpchar <= surname) AND (surname < 'ΜΠ'::bpchar))"

" Rows Removed by Filter: 100"

"Planning time: 0.224 ms"

"Execution time: 0.171 ms"



Βλέπουμε απο την 1η γραμμή ότι ο optimizer επιλέγει sequential scan ξανά (έχοντας φτιάξει ευρετήριο) και εκτελείτε σε ίδιο περίπου χρόνο.Αυτό είναι κάτι αναμενόμενο αφού όπως προαναφέραμε στη συγκεκριμένη περίπτωση έχουμε λίγα σχετικά δεδομένα και το ευρετήριο μας δίνει αποτέλεσμα σε βάσεις με πολλά δεδομένα.Το ίδιο θα γίνει χρησιμοποιώντας clustering (επειδη τα δεδομένα μας είναι λίγα).

Αφού ο optimizer συνεχίζει και επιλέγει sequential scan αυτό που θα κάνουμε είναι να αυξήσουμε τα δεδομένα μας.Χρησιμοποιώντας την συνάρτηση insert_student1_1(num,date) που φτίαξαμε στο ερώτημα 1.1 του 1ου μέρους θα εισάγουμε 10.000 νέους φοιτητές στον πίνακα Student: select insert_student1_1(10000,'2020-10-10') (τυχαίο date)

Επαναλαμβάνοντας την προηγούμενη διαδικάσια και διαγράφοντας το ευρετήριο: drop index indx_st;

Εκτελούμε το query με την explain analyze και παίρνουμε:

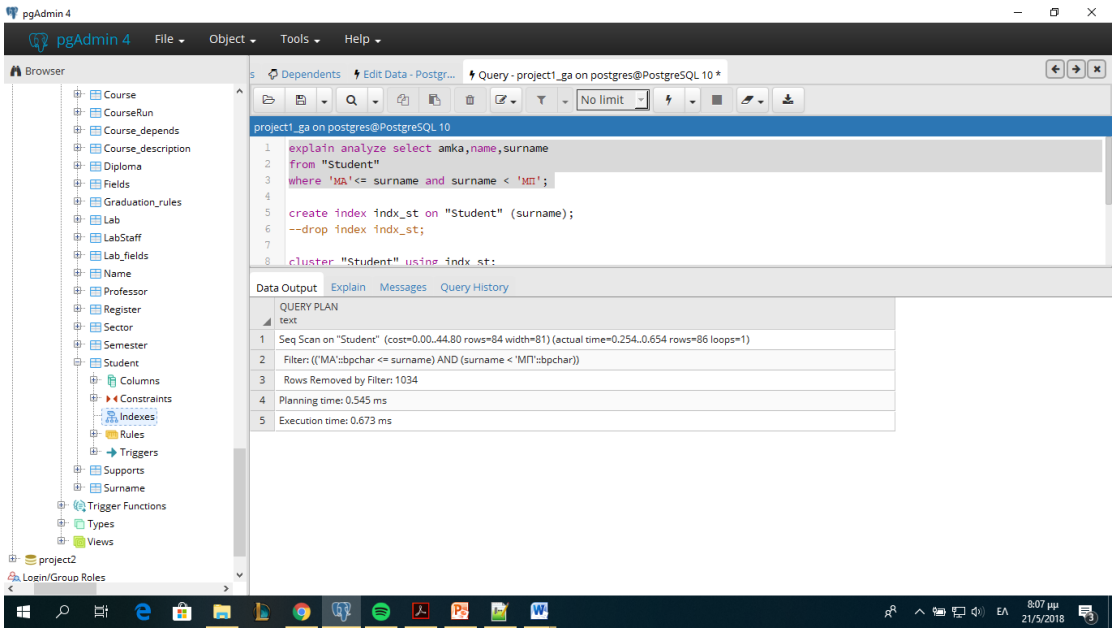
"Seq Scan on "Student" (cost=0.00..44.80 rows=84 width=81) (actual time=0.254..0.654 rows=86 loops=1)"

" Filter: (('MA'::bpchar <= surname) AND (surname < 'ΜΠ'::bpchar))"

" Rows Removed by Filter: 1034"

"Planning time: 0.545 ms"

"Execution time: 0.673 ms"



Παρατηρούμε ότι ο optimizer επιλέγει sequential scan (αφοτου διαγράψαμε το ευρετήριο) με κόστος που φτάνει: 44.8 ,actual time: 0.254..0.654 και execution time: 0.673ms

Τώρα θα τρέξουμε το query έχοντας φτίαξει ευρετήριο: create index indx_st on "Student" (surname); Και έχουμε:

"Bitmap Heap Scan on "Student" (cost=5.14..34.40 rows=84 width=81) (actual time=0.128..0.143 rows=86 loops=1)"

" Recheck Cond: (('MA'::bpchar <= surname) AND (surname < 'ΜΠ'::bpchar))"

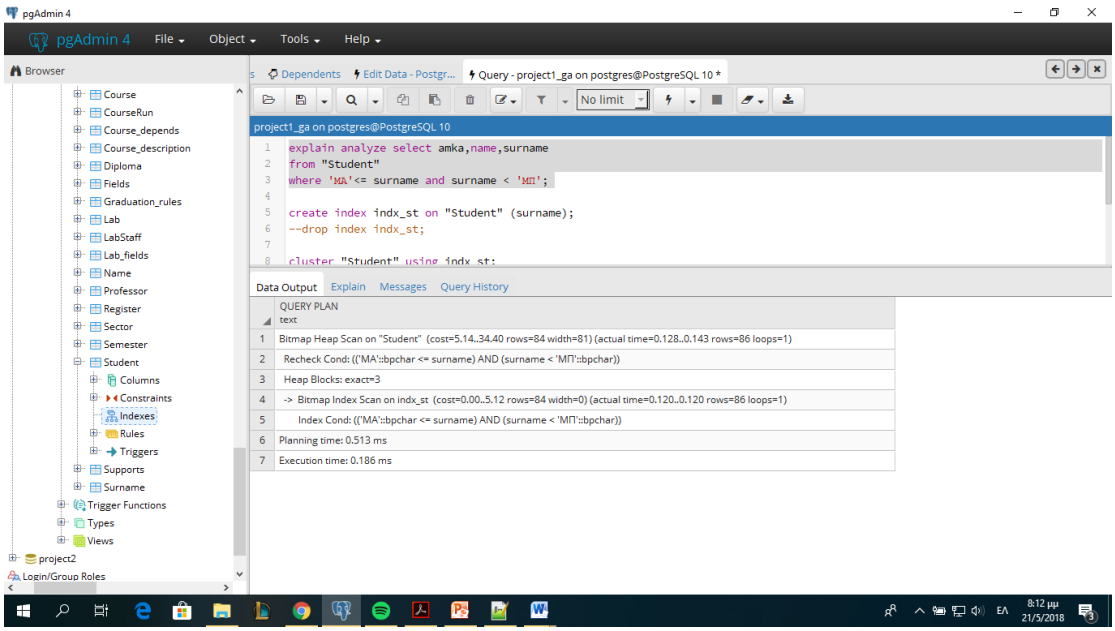
" Heap Blocks: exact=3"

" -> Bitmap Index Scan on indx_st (cost=0.00..5.12 rows=84 width=0) (actual time=0.120..0.120 rows=86 loops=1)"

"   Index Cond: (('MA'::bpchar <= surname) AND (surname < 'ΜΠ'::bpchar))"

"Planning time: 0.513 ms"

"Execution time: 0.186 ms"



Παρατηρούμε ότι ο optimizer επιλέγει heap scan και το κόστος να έχει μειωθεί σε: 33.40, το actual time: 0.128..0.143 και το execution time: 0.186ms. Ένα αναμενόμενο αποτέλεσμα αφού όπως προαναφέραμε το index δίνει αποτέλεσμα όταν υπάρχουν πολλά δεδομένα.Όσα περισσότερα δεδομένα τοσο πιο εμφανή η διαφορά.

Αξιοποιώντας τώρα την δυνατότητα ομαδοποίησης (clustering) : cluster "Student" using indx_st; Έχουμε:

"Bitmap Heap Scan on "Student" (cost=5.14..34.40 rows=84 width=81) (actual time=0.091..0.106 rows=86 loops=1)"

" Recheck Cond: (('MA'::bpchar <= surname) AND (surname < 'ΜΠ'::bpchar))"

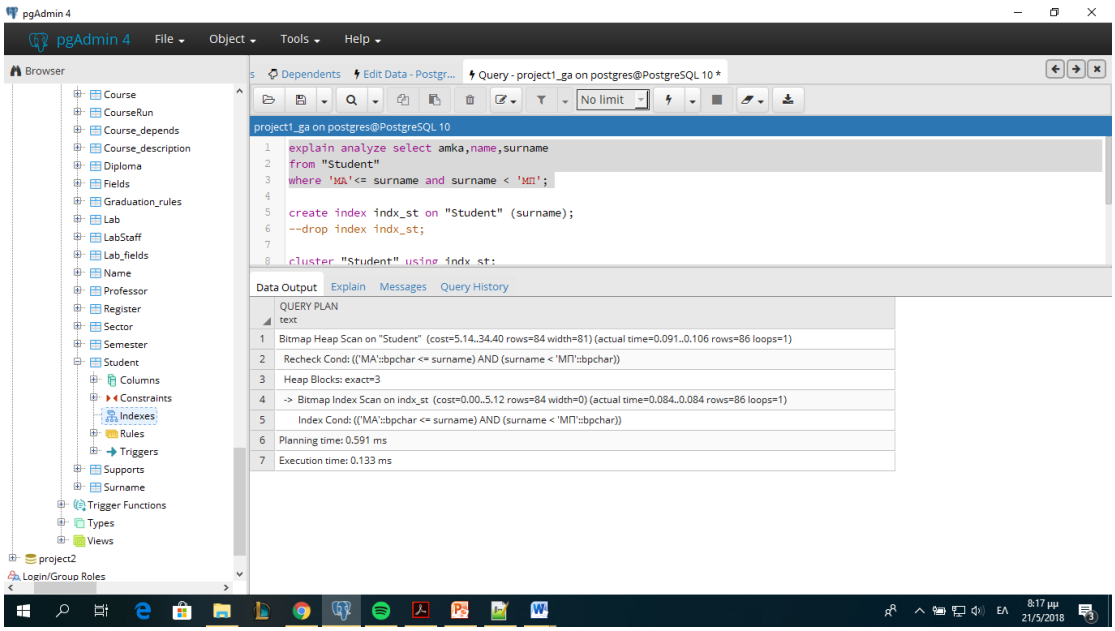
" Heap Blocks: exact=3"

" -> Bitmap Index Scan on indx_st (cost=0.00..5.12 rows=84 width=0) (actual time=0.084..0.084 rows=86 loops=1)"

"   Index Cond: (('MA'::bpchar <= surname) AND (surname < 'ΜΠ'::bpchar))"

"Planning time: 0.591 ms"

"Execution time: 0.133 ms"



Το κόστος έχει αυξηθεί λιγο σε 34.40 και το actual time έχει μειωθεί σε 0,083..0,099.

Επομένως όταν έχουμε λίγα σχετικά δεδομένα ο optimizer επιλέγει sequential scan γιατι δεν υπάρχει μεγάλη διαφορά σε σχέση με την χρήση ευρετηρίων.Όταν όμως έχουμε μια βάση με δεκάδες χιλιάδες ή εκατοντάδες χιλιάδες δεδομένα τότε, η χρήση άλλων μέσων βελτίωσης του χρόνου εκτέλεσης μιας εντολής είναι απαραίτητη (όπως indexes, clustering, sorting etc.)

Τέλος πιστεύω πώς πρέπει να σημειωθούν δυο πράγματα.

1)Αν και εδώ χρησιμοποιήσαμε indexes και εξετάζουμε τα πλεονεκτήματα τους τα ευρετήρια έχουν και μειωνεκτήματα.Θα αναφέρω 3 απο αυτά με σειρά απο λιγότερης σε περισσότερης σημασίας:
α)Καταλαμβάνουν χώρο (μηδαμινό το κακό αφού η μνήμη στην σημερινή εποχή είναι φθηνή)
β)Κατασκευή ενός ευρετηρίου (σε μερικές περιπτώσεις σου τρώει χρόνο)
γ)Συντήρηση ενός ευρετηρίου (όταν τροποποιούνται πίνακες και δεδομένα τότε θα πρέπει να γίνεται και η κατάλληλη αλλαγή στα ευρετήρια ώστε να συμβαδίζουν)

2)Η επιλογή της κατασκευής ενός ευρετηρίου θα πρέπει να γίνεται με βάση 3 πράγματα:
α)Το μέγεθος ενός πίνακα (και των δεδομένων της βάσης μας)
β)Το είδος των δεδομένων και η διανομή τους
γ)Το off-trade δηλαδή το πόσο συχνά θα χρησιμοποιηθεί με το αν αξίζει τον κόπο κατασκευής(αυτό υπάρχει και στις Όψεις).