## 

## Βάσεις Δεδομένων

Φάση Α και φυσικός σχεδιασμός

**Ομάδα LAB30244760**

|  |
| --- |
| ***Τρίμας Χρήστος 2016030054*** |
| ***Παντελής Κωσταντίνος 2015030070*** |

## Σκοπός εργαστηριακής άσκησης:

Σκοπός της εργασίας είναι η εξοικείωση των φοιτητών με τις σχεσιακές βάσεις δεδομένων και με την γλώσσα PostgresSql, όπως και το περιβάλλον pgadmin. Η παρούσα αναφορά, περιλαμβάνει την αντιστοιχία των συναρτήσεων που υλοποιήθηκαν με τα ερωτήματα της Α φάσης, καθώς επίσης και την μελέτη απόδοσης των indexes που δημιουργήθηκαν κατά την φάση του φυσικού σχεδιασμού της βάσης.

## Φάση Α:

**Διαχείριση δεδομένων μέρος 3:**

3.1 🡪 χωρίστηκε σε 3 συναρτήσεις. insert\_professors\_3\_1() , insert\_students\_3\_1(), insert\_labstaff\_3\_1().

3.2 🡪 calc\_final\_grade\_3\_2().

**Ανάκτηση δεδομένων και υπολογισμοί μέρος 4:**

4.1 🡪 χωρίστηκε σε 2 συναρτήσεις. find\_4\_1\_lstaff(), find\_4\_1\_prof().

4.2 🡪 prof\_4\_2\_office\_hours().

4.3 🡪 find\_4\_3\_max\_grades().

4.4 🡪 find\_4\_4\_curr\_c\_r().

4.5 🡪 find\_4\_5\_after().

4.6 🡪 find\_4\_6\_lab().

4.7 🡪 find\_4\_7\_workload().

4.8 🡪 find\_4\_8\_room\_dif\_sub().

4.9 🡪 find\_4\_9\_operation().

4.10 🡪 find\_4\_10\_amka()

**Λειτουργικότητα με κατασκευή εναυσμάτων μέρος 5:**

5.1 🡪 **Συναρτήσεις:** check\_participant\_participation\_5\_1(), get\_person\_participation\_5\_1(), get\_lab\_activities\_5\_1(), get\_course\_lab\_hours\_5\_1(), check\_responsible\_participation\_5\_1() και **triggers:** inspect\_participant\_participation\_5\_1, inspect\_responsible\_participation\_5\_1.

5.2 🡪 **Συναρτήσεις:** check\_activity\_time\_constrains\_5\_2(), check\_activity\_schedule\_constrains\_5\_2(), **triggers**: notify\_on\_illegal\_activity\_input\_5\_2, notify\_on\_illegal\_activity\_sched\_5\_2.

5.3 🡪 **Συναρτήσεις:** copy\_existent\_courserun\_5\_3(), insert\_courserun\_5\_3(), **triggers:** create\_future\_courses\_5\_3.

**Λειτουργικότητα με χρήση όψεων μέρος 6:**

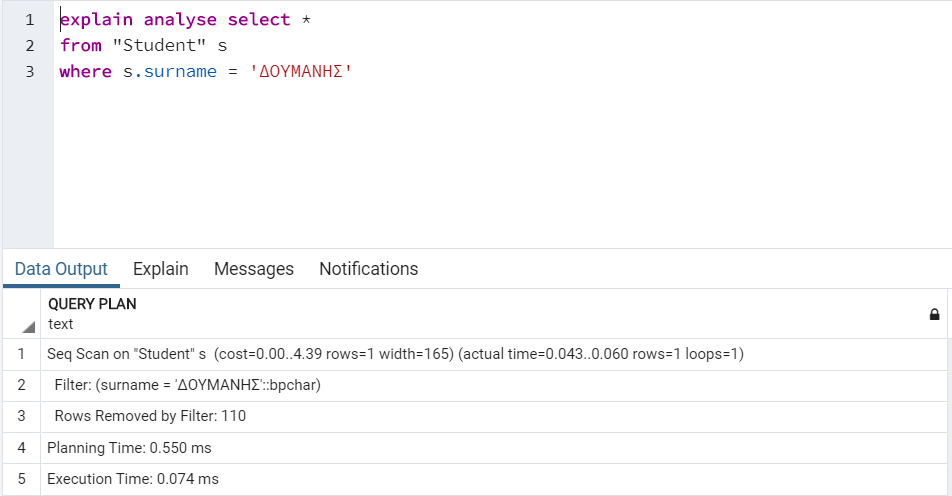
6.1 🡪 passed\_6\_1\_students

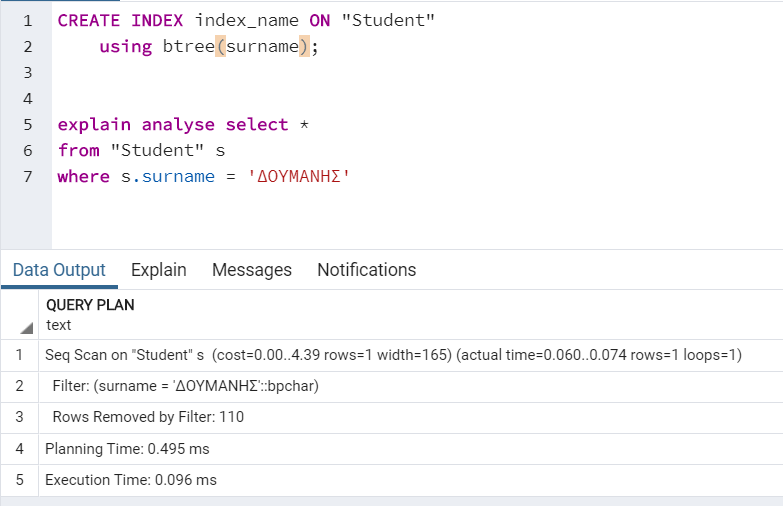
6.2 🡪 weekly\_6\_2\_program

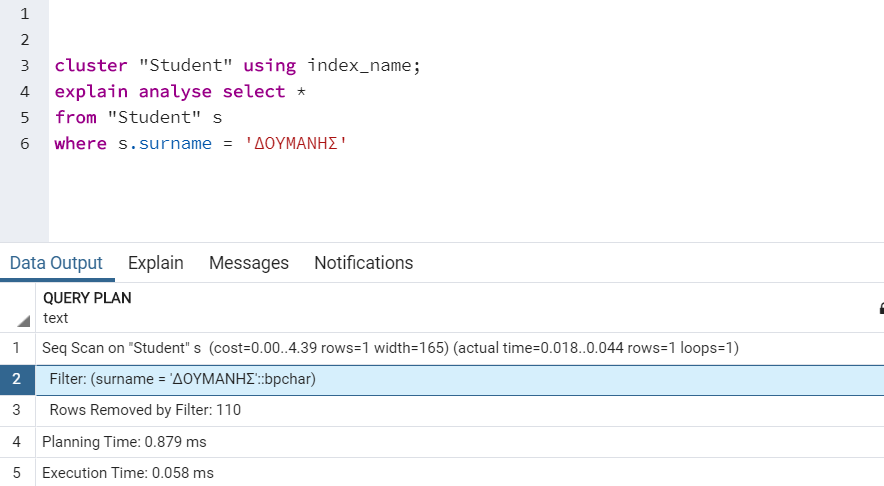
## Φυσικός σχεδιασμός:

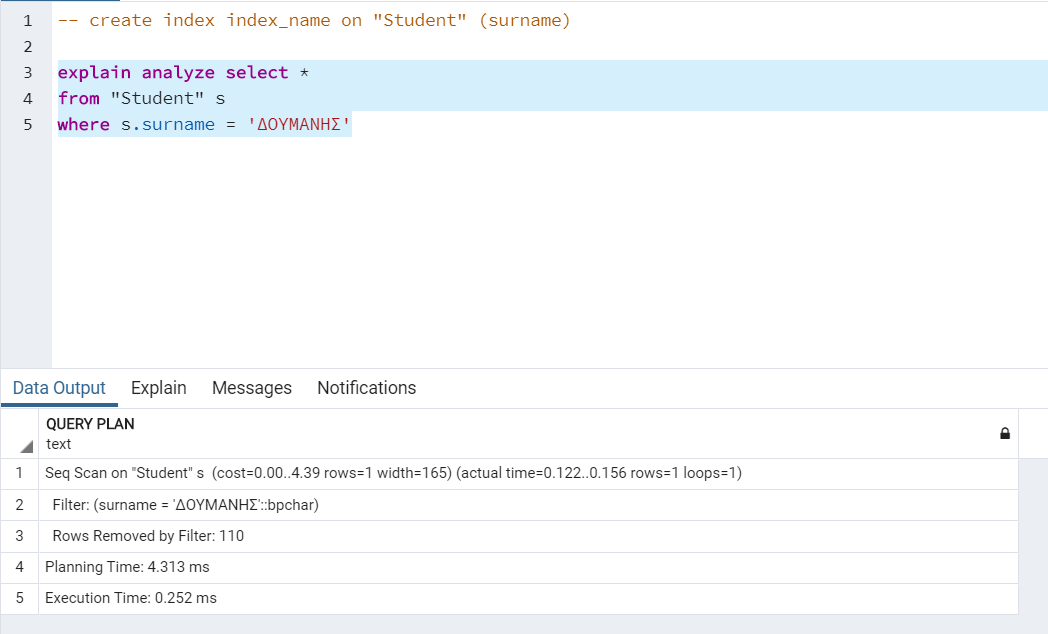
3.Α.) Στο συγκεκριμένο ερώτημα, ζητήθηκε να μελετηθεί το εξής ερώτημα: Να βρεθούν οι φοιτητές που έχουν ένα ορισμένο πατρώνυμο. Τυχαία επιλέχτηκε το επίθετο ΔΟΥΜΑΝΗΣ. Αρχικά για μικρό όγκο δεδομένων περίπου 90-110 φοιτητές, δοκιμάστηκε να εκτελεστεί η explain analyse εντολή της sql, με σκοπό να βρεθεί ο χρόνος εκτέλεσης και προγραμματισμού. Χωρίς την παρουσία κάποιου index τα αποτελέσματα έχουν ως εξής:

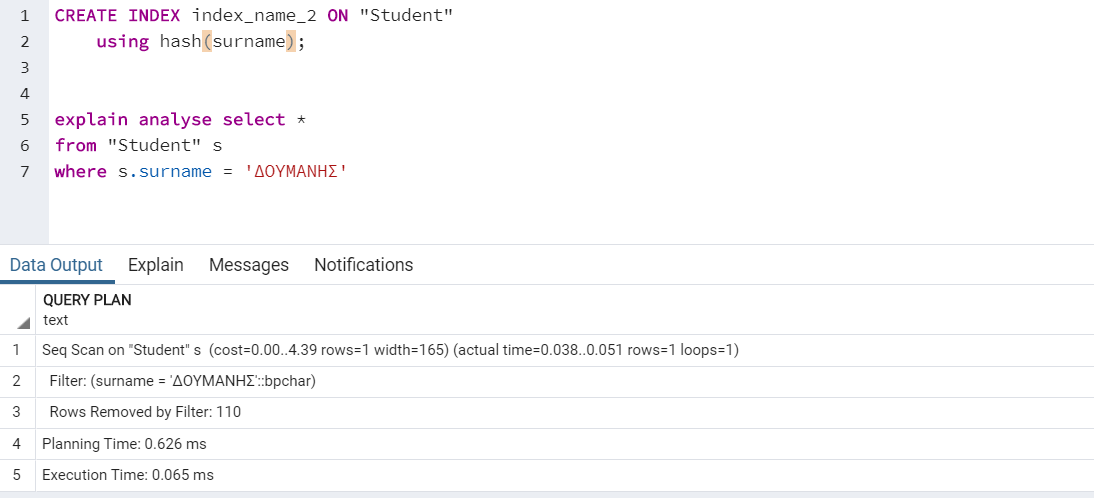
Με την δημιουργία indexes τύπου hash και btree, με ή χωρίς cluster για την 2η περίπτωση, σε μικρό αριθμό δεδομένων, ο “compiler” της sql, αναγνωρίζει ότι η βέλτιστη αναζήτηση γίνεται σειριακά. Καθώς λοιπόν μεγαλύτερο ενδιαφέρον έχουν οι αναζητήσεις για μεγάλο όγκο δεδομένων, παραθέτουμε ενδεικτικά κάποια αποτελέσματα και ο σχολιασμός σε βάθος τον αλγορίθμων θα γίνει παρακάτω.





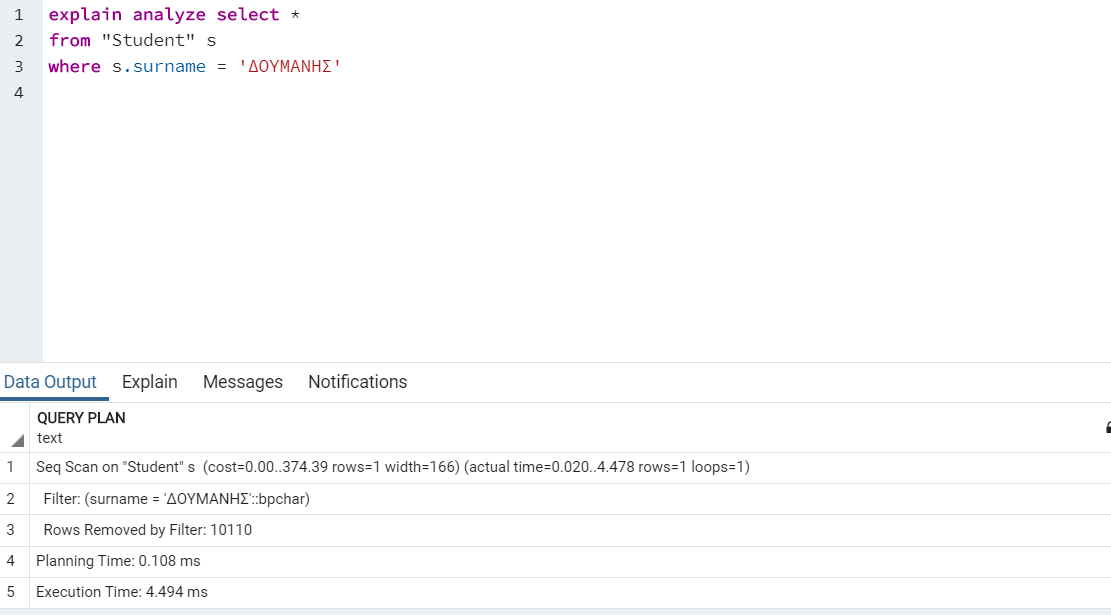


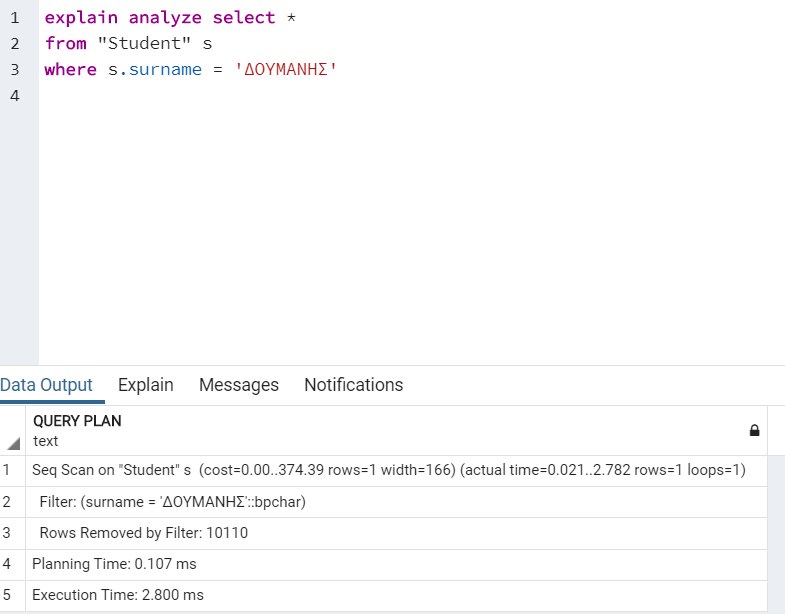


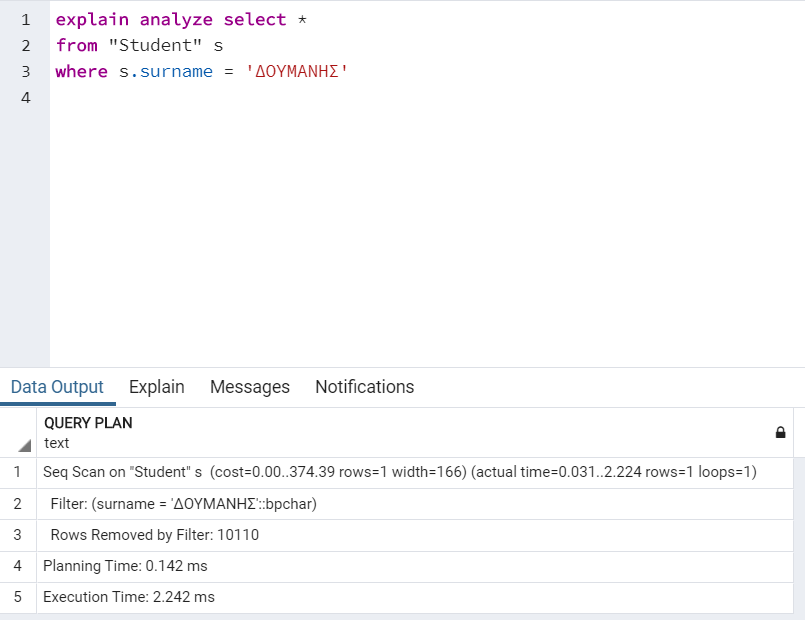


Στη συνέχεια, με χρήση της συνάρτησης insert\_mass\_data\_phase\_2(), προσθέτουμε στον πίνακα Student επιπλέον δεδομένα και επαναλαμβάνουμε το πείραμα μας.

Αρχικά λοιπόν, χωρίς την χρήση κάποιου ευρετηρίου, εκτελούμε σειριακή αναζήτηση.

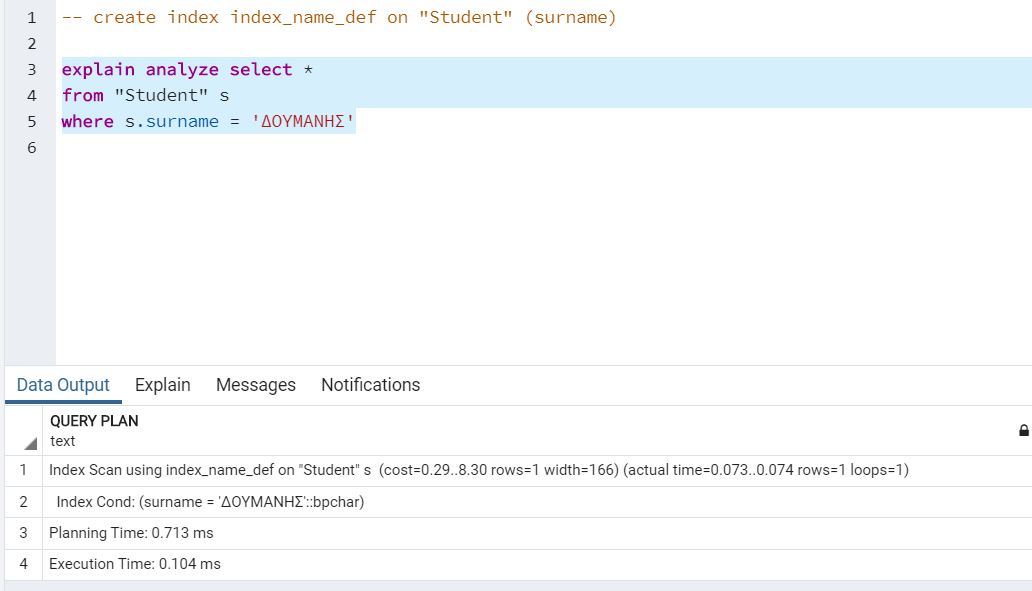




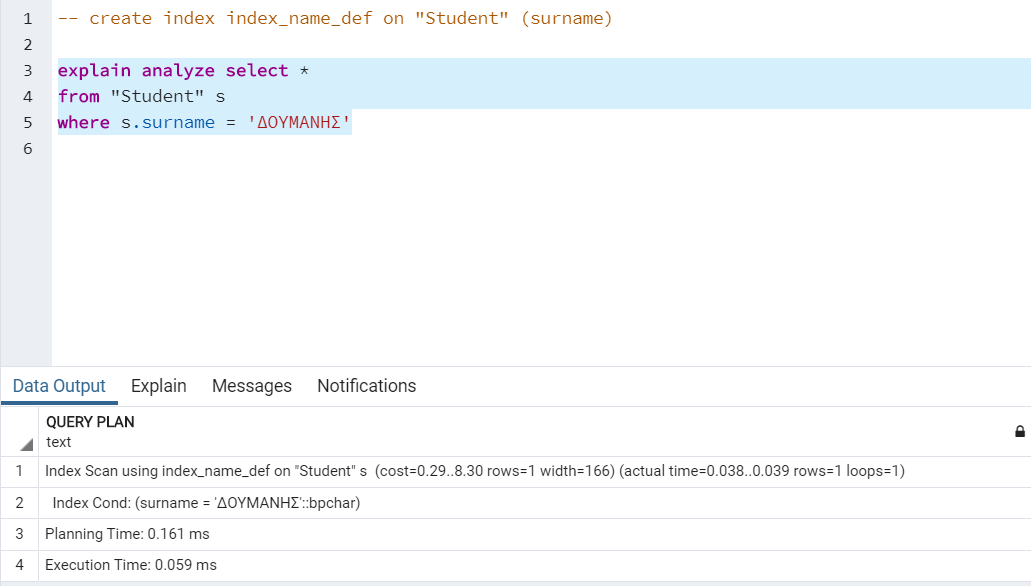


Παρατηρείται ότι το planning time κυμαίνεται στα 0.100 ms, ενώ το execution time χρειάζεται από 2.5 εως και 4.5 ms.

Έπειτα, δοκιμάζουμε να κατασκευάσουμε ένα default index χωρίς την χρήση κάποιου cluster.

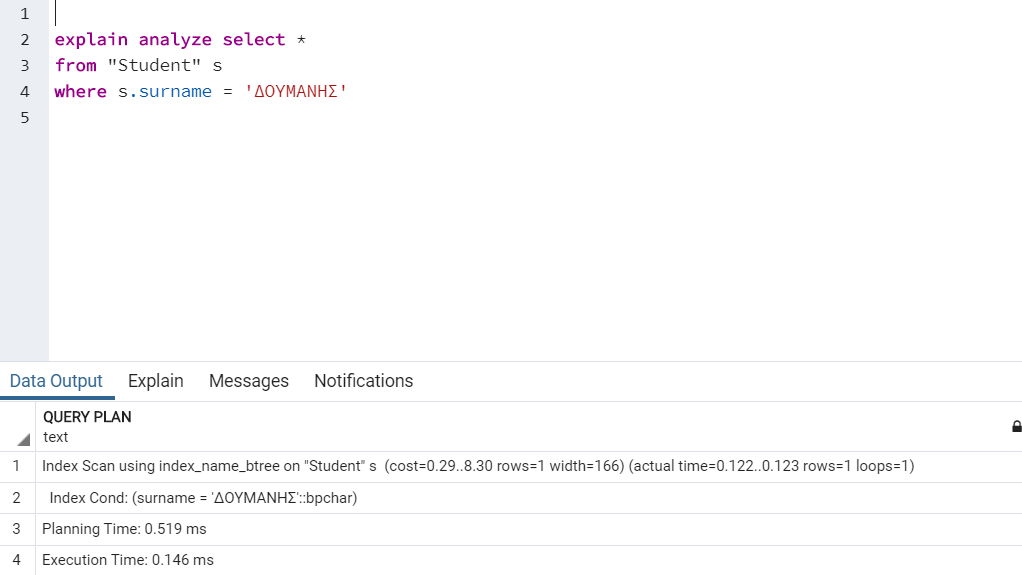




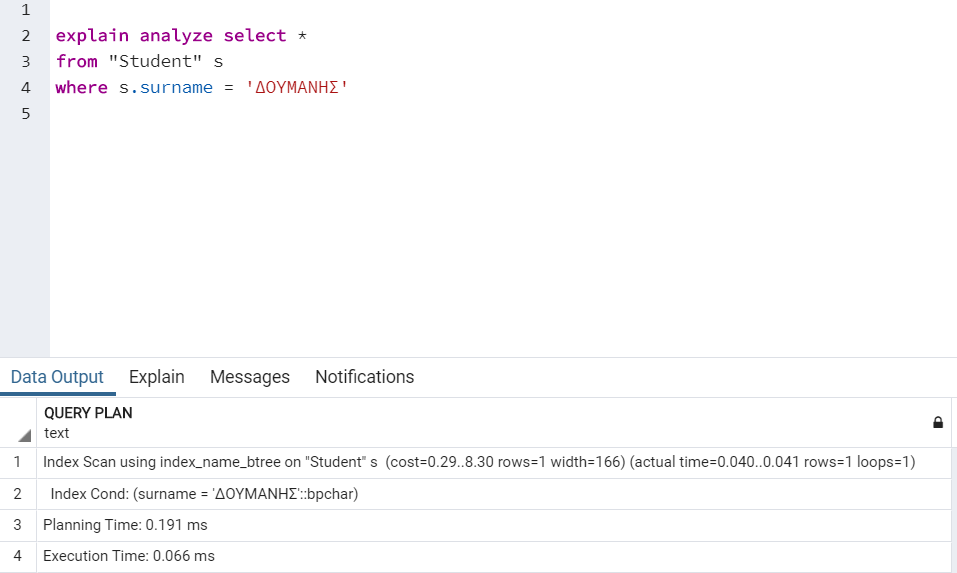


Σε αυτή την περίπτωση, αν και το planning time είναι μεγαλύτερο, φαίνεται ότι το execution time είναι σημαντικά μικρότερο.

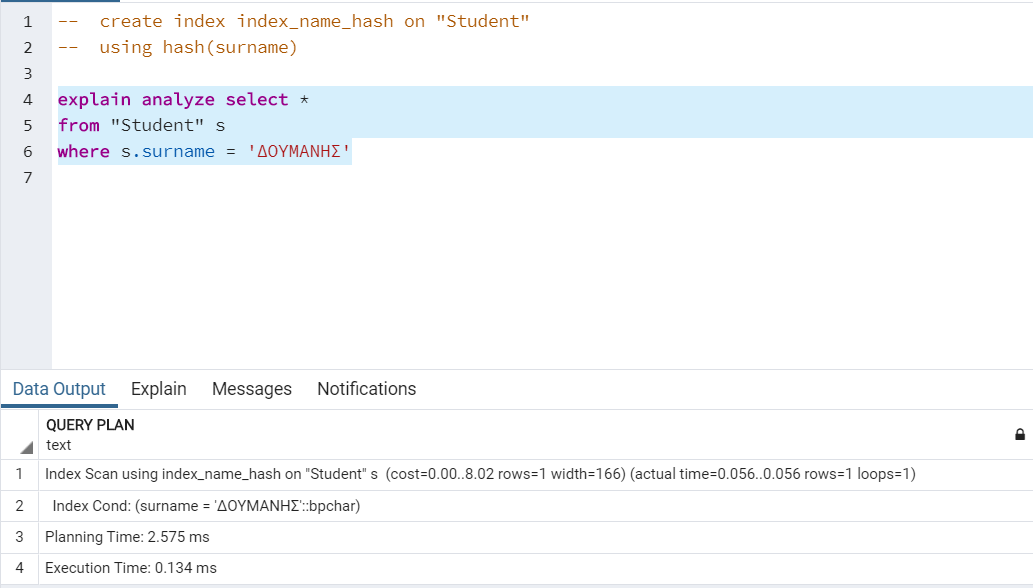
Στη συνέχεια θα εφαρμόσουμε indexes με τους αλγορίθμους hash και btree.

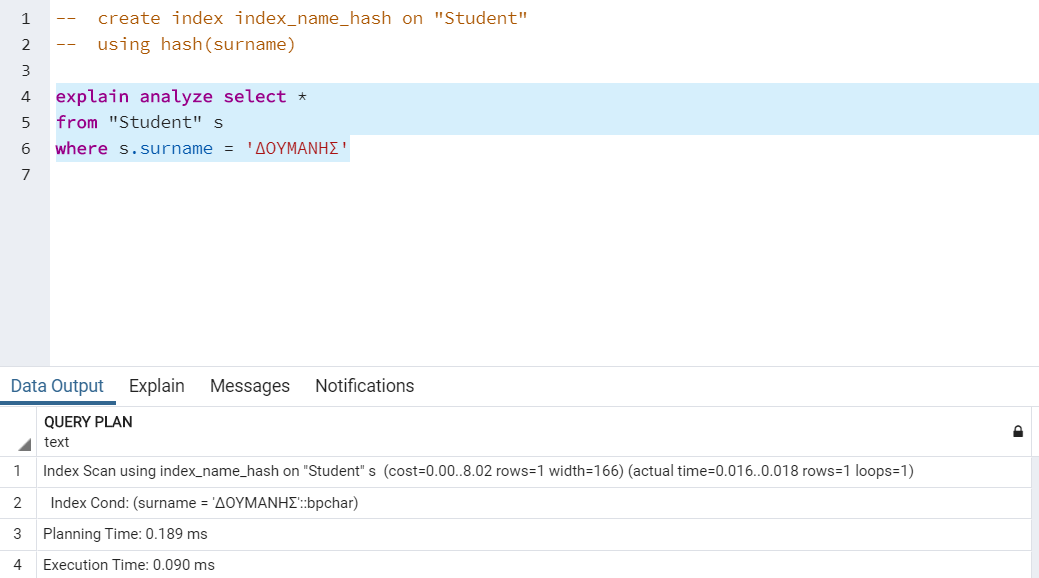


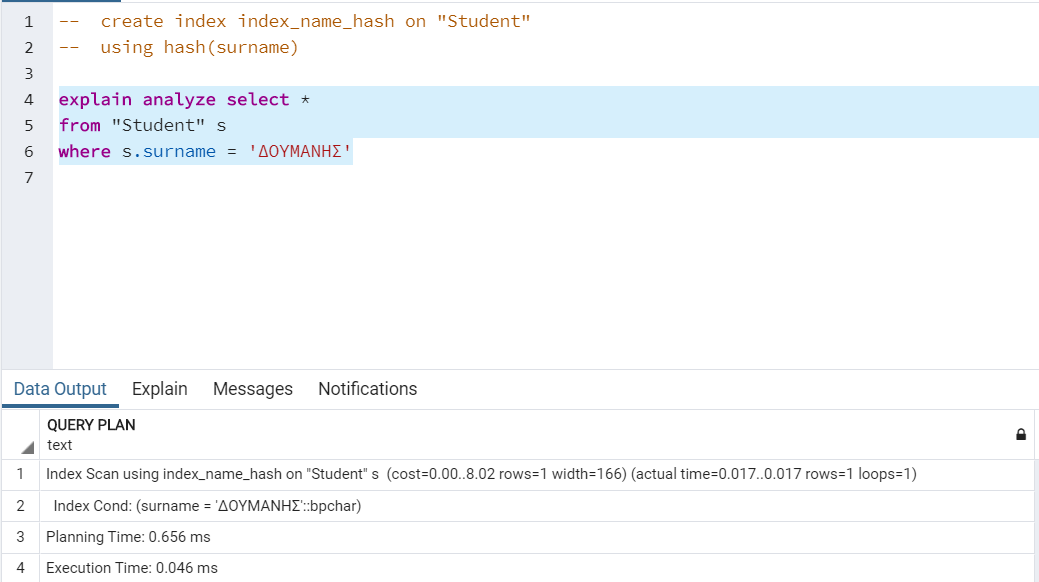




*Btree results.*

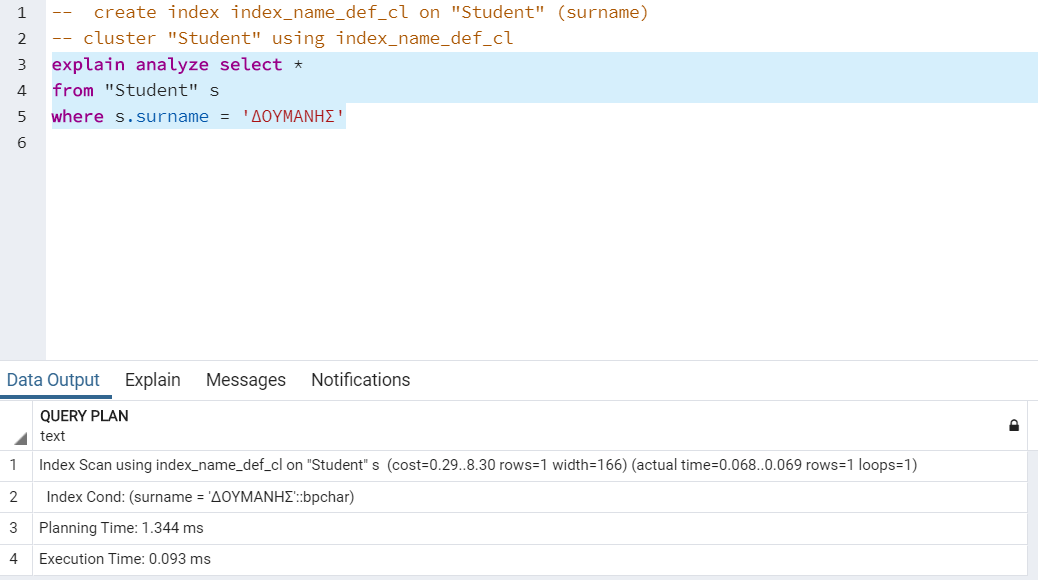


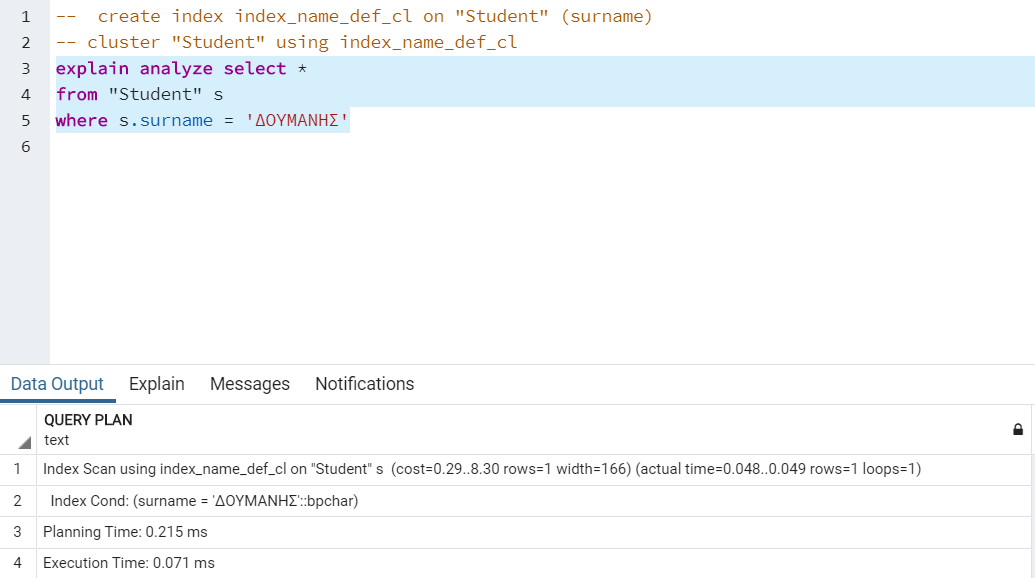


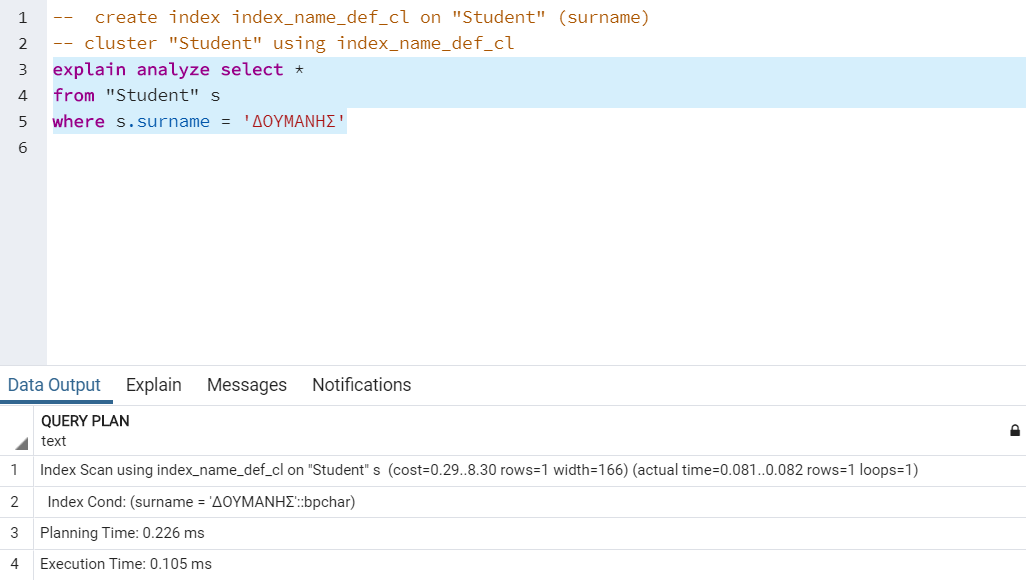


*Hash results.*

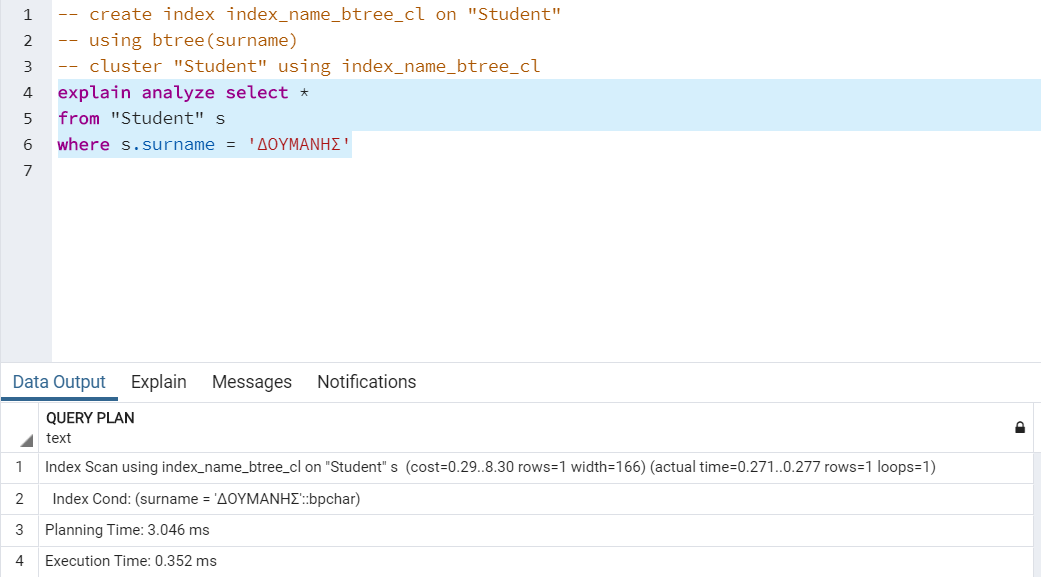
Πριν το σχολιασμό των αποτελεσμάτων, θα εκτελεστεί για το default ευρετήριο και για το btree η διαδικασία αναζήτησης με ομαδοποίηση αντίστοιχα.

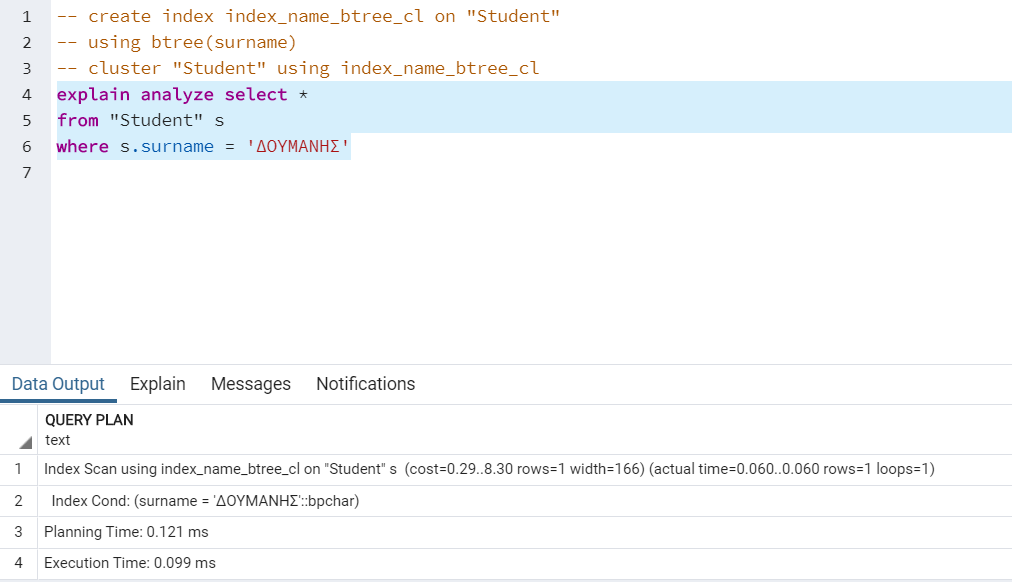


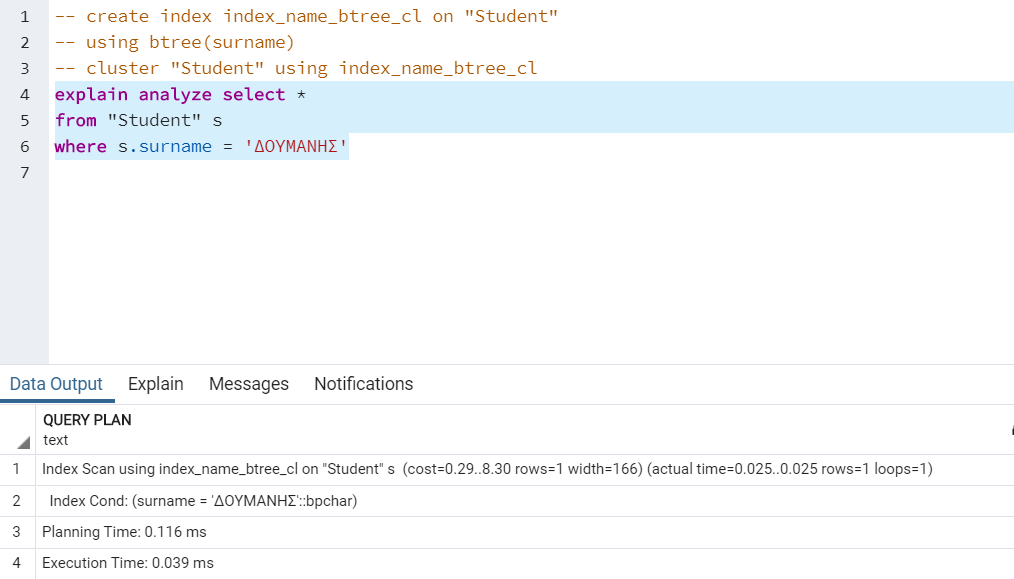




Για clustering με btree:







Είναι φανερό, πως η χρήση οποιουδήποτε ευρετηρίου, έχει σημαντική βελτίωση στο χρόνο εκτέλεσης της αναζήτησης. Συγκεκριμένα για το πρόβλημα αυτό, καθώς πρόκειται για πρόβλημα αναζήτησης ισότητας, η βέλτιστη μέθοδος είναι η hash search( η οποία μάλιστα δεν μπορεί να εφαρμόσει ομαδοποίηση), btree, θα μπορούσαμε να είχαμε επιλέξει αν το πρόβλημα αφορούσε την εύρεση κάποιου διαστήματος τιμών ή κάποιας ανισότητας. Το σίγουρο είναι ότι για μεγάλο όγκο δεδομένων, η σειριακή αναζήτηση δεν είναι καλή λύση, σε αντίθεση με μικρό αριθμό δεδομένων, όπου η εφαρμογή κάποιου άλλου αλγορίθμου δεν θα είχε κάποια σημαντική βελτιστοποίηση και μάλιστα θα αύξανε το planning time.

Παρακάτω, είναι η βέλτιστη επιλογή του compiler της sql, με την παρουσία όλων τον indexes με ή χωρίς clustering. Η βέλτιστη επιλογή του, είναι αναζήτηση με hashing.

