



Πολυτεχνείο  
Κρήτης

## Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

### 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 PostgreSQL, 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 επιλέχθηκε 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 2<sup>η</sup> 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 των αλγορίθμων θα γίνει παρακάτω.

```
1 | explain analyse select *
2 | from "Student" s
3 | where s.surname = 'ΔΟΥΜΑΝΗΣ'
```

Data Output	Explain	Messages	Notifications
<b>QUERY PLAN</b>			
text			
1	Seq Scan on "Student" s (cost=0.00..4.39 rows=1 width=165) (actual time=0.043..0.060 rows=1 loops=1)		
2	Filter: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)		
3	Rows Removed by Filter: 110		
4	Planning Time: 0.550 ms		
5	Execution Time: 0.074 ms		

1	CREATE INDEX index_name ON "Student"
2	using btree(surname);
3	
4	
5	explain analyse select *
6	from "Student" s
7	where s.surname = 'ΔΟΥΜΑΝΗΣ'

Data Output	Explain	Messages	Notifications
<div> <div>QUERY PLAN</div> <div>text</div> </div> <div> <div>1</div> <div>Seq Scan on "Student" s (cost=0.00..4.39 rows=1 width=165) (actual time=0.060..0.074 rows=1 loops=1)</div> </div> <div> <div>2</div> <div>Filter: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)</div> </div> <div> <div>3</div> <div>Rows Removed by Filter: 110</div> </div> <div> <div>4</div> <div>Planning Time: 0.495 ms</div> </div> <div> <div>5</div> <div>Execution Time: 0.096 ms</div> </div>			

1	
2	
3	cluster "Student" using index_name;
4	explain analyse select *
5	from "Student" s
6	where s.surname = 'ΔΟΥΜΑΝΗΣ'

Data Output	Explain	Messages	Notifications
<div> <div>QUERY PLAN</div> <div>text</div> </div> <div> <div>1</div> <div>Seq Scan on "Student" s (cost=0.00..4.39 rows=1 width=165) (actual time=0.018..0.044 rows=1 loops=1)</div> </div> <div> <div>2</div> <div>Filter: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)</div> </div> <div> <div>3</div> <div>Rows Removed by Filter: 110</div> </div> <div> <div>4</div> <div>Planning Time: 0.879 ms</div> </div> <div> <div>5</div> <div>Execution Time: 0.058 ms</div> </div>			

1	-- create index index_name on "Student" (surname)
2	
3	explain analyze select *
4	from "Student" s
5	where s.surname = 'ΔΟΥΜΑΝΗΣ'

Data Output	Explain	Messages	Notifications
<div> <div>QUERY PLAN</div> <div>text</div> </div> <div> <div>1</div> <div>Seq Scan on "Student" s (cost=0.00..4.39 rows=1 width=165) (actual time=0.122..0.156 rows=1 loops=1)</div> </div> <div> <div>2</div> <div>Filter: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)</div> </div> <div> <div>3</div> <div>Rows Removed by Filter: 110</div> </div> <div> <div>4</div> <div>Planning Time: 4.313 ms</div> </div> <div> <div>5</div> <div>Execution Time: 0.252 ms</div> </div>			

1	CREATE INDEX index_name_2 ON "Student"
2	using hash(surname);
3	
4	
5	explain analyse select *
6	from "Student" s
7	where s.surname = 'ΔΟΥΜΑΝΗΣ'

Data Output	Explain	Messages	Notifications
<div> <div>QUERY PLAN</div> <div>text</div> </div> <div> <div>1</div> <div>Seq Scan on "Student" s (cost=0.00..4.39 rows=1 width=165) (actual time=0.038..0.051 rows=1 loops=1)</div> </div> <div> <div>2</div> <div>Filter: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)</div> </div> <div> <div>3</div> <div>Rows Removed by Filter: 110</div> </div> <div> <div>4</div> <div>Planning Time: 0.626 ms</div> </div> <div> <div>5</div> <div>Execution Time: 0.065 ms</div> </div>			

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.

```

1 explain analyze select *
2 from "Student" s
3 where s.surname = 'ΔΟΥΜΑΝΗΣ'
4

```

Data Output Explain Messages Notifications

QUERY PLAN	
1	Seq Scan on "Student" s (cost=0.00..374.39 rows=1 width=166) (actual time=0.020..4.478 rows=1 loops=1)
2	Filter: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
3	Rows Removed by Filter: 10110
4	Planning Time: 0.108 ms
5	Execution Time: 4.494 ms

```

1 explain analyze select *
2 from "Student" s
3 where s.surname = 'ΔΟΥΜΑΝΗΣ'
4

```

Data Output Explain Messages Notifications

QUERY PLAN	
1	Seq Scan on "Student" s (cost=0.00..374.39 rows=1 width=166) (actual time=0.021..2.782 rows=1 loops=1)
2	Filter: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
3	Rows Removed by Filter: 10110
4	Planning Time: 0.107 ms
5	Execution Time: 2.800 ms

```

1 explain analyze select *
2 from "Student" s
3 where s.surname = 'ΔΟΥΜΑΝΗΣ'
4

```

Data Output Explain Messages Notifications

QUERY PLAN	
1	Seq Scan on "Student" s (cost=0.00..374.39 rows=1 width=166) (actual time=0.031..2.224 rows=1 loops=1)
2	Filter: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
3	Rows Removed by Filter: 10110
4	Planning Time: 0.142 ms
5	Execution Time: 2.242 ms

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

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

1	-- create index index_name_def on "Student" (surname)
2	
3	explain analyze select *
4	from "Student" s
5	where s.surname = 'ΔΟΥΜΑΝΗΣ'
6	
<div> Data Output Explain Messages Notifications </div>	
<div> <div>QUERY PLAN</div> <div>text</div> <div>1</div> <div>Index Scan using index_name_def on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.073..0.074 rows=1 loops=1)</div> <div>2</div> <div>Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)</div> <div>3</div> <div>Planning Time: 0.713 ms</div> <div>4</div> <div>Execution Time: 0.104 ms</div> </div>	
1	-- create index index_name_def on "Student" (surname)
2	
3	explain analyze select *
4	from "Student" s
5	where s.surname = 'ΔΟΥΜΑΝΗΣ'
6	
<div> Data Output Explain Messages Notifications </div>	
<div> <div>QUERY PLAN</div> <div>text</div> <div>1</div> <div>Index Scan using index_name_def on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.029..0.030 rows=1 loops=1)</div> <div>2</div> <div>Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)</div> <div>3</div> <div>Planning Time: 0.227 ms</div> <div>4</div> <div>Execution Time: 0.045 ms</div> </div>	
1	-- create index index_name_def on "Student" (surname)
2	
3	explain analyze select *
4	from "Student" s
5	where s.surname = 'ΔΟΥΜΑΝΗΣ'
6	
<div> Data Output Explain Messages Notifications </div>	
<div> <div>QUERY PLAN</div> <div>text</div> <div>1</div> <div>Index Scan using index_name_def on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.038..0.039 rows=1 loops=1)</div> <div>2</div> <div>Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)</div> <div>3</div> <div>Planning Time: 0.161 ms</div> <div>4</div> <div>Execution Time: 0.059 ms</div> </div>	

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.

1	
2	explain analyze select *
3	from "Student" s
4	where s.surname = 'ΔΟΥΜΑΝΗΣ'
5	
<div> Data Output Explain Messages Notifications </div>	
<div> <div>QUERY PLAN</div> <div>text</div> <div>1</div> <div>Index Scan using index_name_btree on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.122..0.123 rows=1 loops=1)</div> <div>2</div> <div>Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)</div> <div>3</div> <div>Planning Time: 0.519 ms</div> <div>4</div> <div>Execution Time: 0.146 ms</div> </div>	

1  
2  
3  
4  
5

```
explain analyze select *
from "Student" s
where s.surname = 'ΔΟΥΜΑΝΗΣ'
```

Data Output

Explain

Messages

Notifications

QUERY PLAN

text

1  
2  
3  
4

```
Index Scan using index_name_btree on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.045..0.047 rows=1 loops=1)
Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
Planning Time: 0.189 ms
Execution Time: 0.071 ms
```

1  
2  
3  
4  
5

```
explain analyze select *
from "Student" s
where s.surname = 'ΔΟΥΜΑΝΗΣ'
```

Data Output

Explain

Messages

Notifications

QUERY PLAN

text

1  
2  
3  
4

```
Index Scan using index_name_btree on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.040..0.041 rows=1 loops=1)
Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
Planning Time: 0.191 ms
Execution Time: 0.066 ms
```

*Btree results.*

1  
2  
3  
4  
5  
6  
7

```
-- create index index_name_hash on "Student"
-- using hash(surname)

explain analyze select *
from "Student" s
where s.surname = 'ΔΟΥΜΑΝΗΣ'
```

Data Output

Explain

Messages

Notifications

QUERY PLAN

text

1  
2  
3  
4

```
Index Scan using index_name_hash on "Student" s (cost=0.00..8.02 rows=1 width=166) (actual time=0.056..0.056 rows=1 loops=1)
Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
Planning Time: 2.575 ms
Execution Time: 0.134 ms
```

1  
2  
3  
4  
5  
6  
7

```
-- create index index_name_hash on "Student"
-- using hash(surname)

explain analyze select *
from "Student" s
where s.surname = 'ΔΟΥΜΑΝΗΣ'
```

Data Output

Explain

Messages

Notifications

QUERY PLAN

text

1  
2  
3  
4

```
Index Scan using index_name_hash on "Student" s (cost=0.00..8.02 rows=1 width=166) (actual time=0.016..0.018 rows=1 loops=1)
Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
Planning Time: 0.189 ms
Execution Time: 0.090 ms
```

1	-- create index index_name_hash on "Student"
2	-- using hash(surname)
3	
4	explain analyze select *
5	from "Student" s
6	where s.surname = 'ΔΟΥΜΑΝΗΣ'
7	

Data Output	Explain	Messages	Notifications
<div> <div>QUERY PLAN</div> <div>text</div> <div> <div>1</div> <div>Index Scan using index_name_hash on "Student" s (cost=0.00..8.02 rows=1 width=166) (actual time=0.017..0.017 rows=1 loops=1)</div> <div>2</div> <div>Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ':bpchar)</div> <div>3</div> <div>Planning Time: 0.656 ms</div> <div>4</div> <div>Execution Time: 0.046 ms</div> </div> </div>			

Hash results.

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

1	-- create index index_name_def_cl on "Student" (surname)
2	-- cluster "Student" using index_name_def_cl
3	explain analyze select *
4	from "Student" s
5	where s.surname = 'ΔΟΥΜΑΝΗΣ'
6	

Data Output	Explain	Messages	Notifications
<div> <div>QUERY PLAN</div> <div>text</div> <div> <div>1</div> <div>Index Scan using index_name_def_cl on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.068..0.069 rows=1 loops=1)</div> <div>2</div> <div>Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ':bpchar)</div> <div>3</div> <div>Planning Time: 1.344 ms</div> <div>4</div> <div>Execution Time: 0.093 ms</div> </div> </div>			

1	-- create index index_name_def_cl on "Student" (surname)
2	-- cluster "Student" using index_name_def_cl
3	explain analyze select *
4	from "Student" s
5	where s.surname = 'ΔΟΥΜΑΝΗΣ'
6	

Data Output	Explain	Messages	Notifications
<div> <div>QUERY PLAN</div> <div>text</div> <div> <div>1</div> <div>Index Scan using index_name_def_cl on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.048..0.049 rows=1 loops=1)</div> <div>2</div> <div>Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ':bpchar)</div> <div>3</div> <div>Planning Time: 0.215 ms</div> <div>4</div> <div>Execution Time: 0.071 ms</div> </div> </div>			

1	-- create index index_name_def_cl on "Student" (surname)
2	-- cluster "Student" using index_name_def_cl
3	explain analyze select *
4	from "Student" s
5	where s.surname = 'ΔΟΥΜΑΝΗΣ'
6	

Data Output	Explain	Messages	Notifications
<div> <div>QUERY PLAN</div> <div>text</div> <div> <div>1</div> <div>Index Scan using index_name_def_cl on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.081..0.082 rows=1 loops=1)</div> <div>2</div> <div>Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ':bpchar)</div> <div>3</div> <div>Planning Time: 0.226 ms</div> <div>4</div> <div>Execution Time: 0.105 ms</div> </div> </div>			

Για clustering με btree:

```
1 -- create index index_name_btree_cl on "Student"
2 -- using btree(surname)
3 -- cluster "Student" using index_name_btree_cl
4 explain analyze select *
5 from "Student" s
6 where s.surname = 'ΔΟΥΜΑΝΗΣ'
7
```

Data Output Explain Messages Notifications

QUERY PLAN
text
1 Index Scan using index_name_btree_cl on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.271..0.277 rows=1 loops=1)
2 Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
3 Planning Time: 3.046 ms
4 Execution Time: 0.352 ms

```
1 -- create index index_name_btree_cl on "Student"
2 -- using btree(surname)
3 -- cluster "Student" using index_name_btree_cl
4 explain analyze select *
5 from "Student" s
6 where s.surname = 'ΔΟΥΜΑΝΗΣ'
7
```

Data Output Explain Messages Notifications

QUERY PLAN
text
1 Index Scan using index_name_btree_cl on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.060..0.060 rows=1 loops=1)
2 Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
3 Planning Time: 0.121 ms
4 Execution Time: 0.099 ms

```
1 -- create index index_name_btree_cl on "Student"
2 -- using btree(surname)
3 -- cluster "Student" using index_name_btree_cl
4 explain analyze select *
5 from "Student" s
6 where s.surname = 'ΔΟΥΜΑΝΗΣ'
7
```

Data Output Explain Messages Notifications

QUERY PLAN
text
1 Index Scan using index_name_btree_cl on "Student" s (cost=0.29..8.30 rows=1 width=166) (actual time=0.025..0.025 rows=1 loops=1)
2 Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
3 Planning Time: 0.116 ms
4 Execution Time: 0.039 ms

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 των indexes με ή χωρίς clustering. Η βέλτιστη επιλογή του, είναι αναζήτηση με hashing.




```

1  -- create index index_name_btree on "Student"
2  -- using btree(surname)
3  -- cluster "Student" using index_name_def_cl
4  explain analyze select *
5  from "Student" s
6  where s.surname = 'ΔΟΥΜΑΝΗΣ'
7

```

[Data Output](#) [Explain](#) [Messages](#) [Notifications](#)

 **QUERY PLAN**  
text

1	Index Scan using index_name_hash on "Student" s (cost=0.00..8.02 rows=1 width=166) (actual time=0.013..0.014 rows=1 loops=1)
2	Index Cond: (surname = 'ΔΟΥΜΑΝΗΣ'::bpchar)
3	Planning Time: 0.226 ms
4	Execution Time: 0.030 ms