
Πανεπιστήμιο
Πειραιά



Τμήμα
Πληροφορικής

Σχολή : Τεχνολογιών Πληροφορικής και Επικοινωνιών
Τμήμα: Πληροφορικής

Συστήματα Διαχείρισης Βάσεων Δεδομένων
Εργασία εξαμήνου

Ονοματεπώνυμο: Γκαρτζονίκα Αικατερίνη Π18026
Μπάικας Αναστάσιος Π20131

Επιβλέπων καθηγητής: Θεοδωρίδης Γιάννης

Issue #1

(a)

NOT) Η υλοποίηση του not έγινε με χρήση του πακέτου operator και της συνάρτησης του ne. Αυτή η συνάρτηση προστέθηκε στο dict ops του αρχείου misc.py και με αυτόν τον τρόπο μπορούμε να καλέσουμε ένα not statement. Τα statements είναι της μορφής “select * from student where name not brown”.

```
(smdb)> select * from student where name not brown
```

id (str)	#PK#	name (str)	dept_name (str)	tot_cred (int)
00128		zhang	comp. sci.	102
12345		shankar	comp. sci.	32
19991		brandt	history	80
23121		chavez	finance	110
44553		peltier	physics	56
45678		levy	physics	46
54321		williams	comp. sci.	54
55739		sanchez	music	38
70557		snow	physics	0
76653		aoi	elec. eng.	60
98765		bourikas	elec. eng.	98
98988		tanaka	biology	120

BETWEEN) Η υλοποίηση του between έγινε με την δημιουργία μίας συνάρτησης που ελέγχει εάν τηρείται η διάταξη ανάμεσα στην στήλη που επιλέξαμε και την συνθήκη. Για την λειτουργία αυτού του κατηγορήματος τροποποιήσαμε την συνάρτηση parse_condition ώστε να “σβήνει” τα κενά ανάμεσα στα ορίσματα, ώστε η συνθήκη να περνά τα υπόλοιπα στάδια της προ-επεξεργασίας ως ένα όρισμα, χωρίς να δημιουργείται λάθος. Ειδική μέριμνα έχει ληφθεί ώστε να γίνεται διαφορετικού είδους σύγκριση για αλφαριθμητικά πεδία και διαφορετική για αριθμητικά. Τα statements είναι της μορφής “select * from student where tot_cred between 50 and 100”.

```
(smdb)> select * from student where tot_cred between 50 and 100
```

id (str)	#PK#	name (str)	dept_name (str)	tot_cred (int)
19991		brandt	history	80
44553		peltier	physics	56
54321		williams	comp. sci.	54
76543		brown	comp. sci.	58
76653		aoi	elec. eng.	60
98765		bourikas	elec. eng.	98

(b) Τα δύο ερωτήματα υλοποιήθηκαν παράλληλα ,αφού χρησιμοποιήθηκε η ίδια λογική για την υλοποίηση τους. Η λογική βασίζεται στο ότι οι συνθήκες ανάμεσα στα and και τα or μπορούν να αντιμετωπισθούν ως επιμέρους queries και στην συνέχεια να συνδυαστούν τα αποτελέσματά τους. Για αυτόν τον λόγο υλοποιήθηκε η συνάρτηση get_result στο table.py η οποία εκτελεί την ίδια ακριβώς λειτουργία με την show, απλώς αντί να εκτυπώνει το αποτέλεσμα ,το επιστρέφει στην μορφή λίστας .Έπειτα η εκτέλεση των queries γίνεται από αριστερά προς τα δεξιά και όταν ολοκληρωθεί η διαδικασία έχουμε ως έξοδο το τελικό αποτέλεσμα. Τα statements είναι της μορφής “select * from student where dept_name="comp. sci." and tot_cred>100 or name="aoi"”

```
(smdb)> select * from student where dept_name="comp. sci." and tot_cred>100 or name="aoi"
['00128', 'zhang', 'comp. sci.', 102]
['76653', 'aoi', 'elec. eng.', 60]
(smdb)>
```

Issue #2

Hash-index search) Η υλοποίηση αυτή πραγματοποιήθηκε με ένα πίνακα κατακερματισμού 10 θέσεων (modulo 10). Στην συνέχεια η αναζήτηση γίνεται με βάση την διευθυνσιοδότηση που έγινε παραπάνω και εκτυπώνει τα αποτελέσματα. Η συνάρτηση που υλοποιεί την λειτουργία είναι η _select_where_with_hash_table στο αρχείο table.py η οποία καλείται όταν υπάρξει ερώτημα επί του primary key, το οποίο έχει να κάνει με ισότητα. Η συνάρτηση δουλεύει είτε με * είτε σε συγκεκριμένες στήλες. Παράδειγμα εκτέλεσης “select name,dept_name from student where id=12345”

Σε όλα τα παραδείγματα χρησιμοποιήθηκε το smallRelations database και οι λειτουργίες έχουν εφαρμοστεί στον πίνακα students.

Issue #3

α) Εφαρμόζουμε τους κανόνες της σχεσιακής άλγεβρας πάνω σε ένα dictionary.

Συγκεκριμένα σε αυτήν την περίπτωση στο `dic = interpret(line) (mdb.py)` όπου έχει την μορφή (με query: `select * from (select * from student where tot_cred=56) where dept_name = "physics";`)

```
{'select': '*',  
  'from':  
    {'select': '*',  
      'from': 'student',  
      'where': 'tot_cred=56',  
      'distinct': None,  
      'order by': None,  
      'limit': None,  
      'desc': None},  
    'where': 'dept_name = "physics"',  
    'distinct': None,  
    'order by': None,  
    'limit': None, 'desc': None}
```

```
C:\> Command Prompt - "C:\Anaconda3\condabin\conda.bat" activate mdb - python mdb.py
```

Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

```
C:\Users\táóóc>cd C:\Users\táóóc\Desktop\baseis\miniDB-master\miniDB-master  
C:\Users\táóóc\Desktop\baseis\miniDB-master\miniDB-master>conda activate mdb  
(mdb) C:\Users\táóóc\Desktop\baseis\miniDB-master\miniDB-master>python mdb.py
```

MiniDB 2022

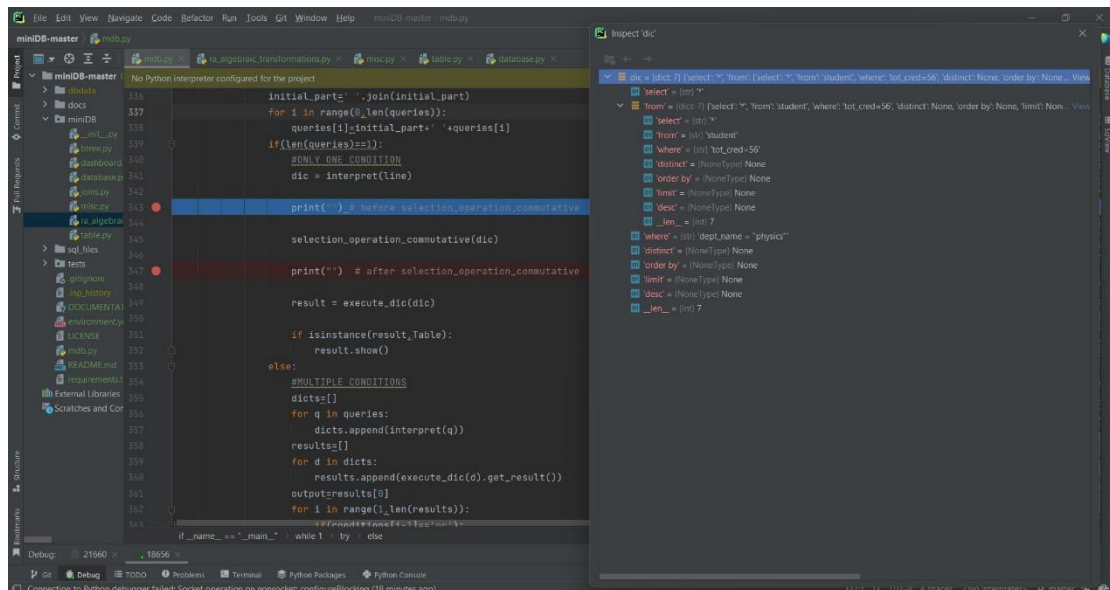
```
(None)> cdb smdb  
(smdb)> select * from (select * from student where tot_cred=56) where dept_name = "physics";  
-
```

Το συγκεκριμένο query σε σχεσιακή άλγεβρα ΣΑ έχει την μορφή $\sigma_{\theta_1}(\sigma_{\theta_2}(E))$.

Αν ο optimizer κρίνει πως πρέπει να εφαρμοσθεί ο κανόνας $\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$

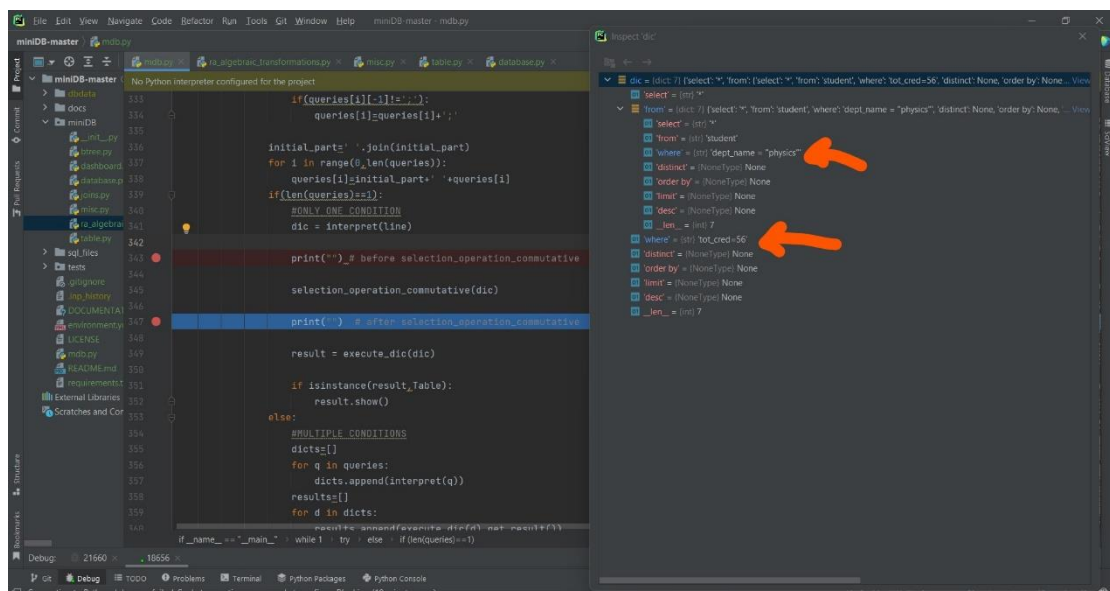
τότε καλεί την συνάρτηση `def selection_operation_commutative(dic)` (`ra_algebraic_transformations.py`). (Δεν υπάρχει optimizer γιατί δεν έχει υλοποιηθεί το ερώτημα 3β. Το παράδειγμα δείχνει την λογική πριν και μετά)

Όλη η διαδικασία της αντιμετάθεσης εφαρμόζεται πάνω στο `dic` πριν πάει στην συνάρτηση `result = execute_dic(dic)` (`mdb.py`).



```
def selection_operation_commutative(dic):  
    initial_parts = '.join(initial_part'  
    for i in range(0, len(queries)):  
        queries[i] = initial_part + 'queries[i]  
    if len(queries) == 1:  
        # ONLY ONE CONDITION  
        dic = interpret(line)  
        print("") # before selection_operation_commutative  
        selection_operation_commutative(dic)  
        print("") # after selection_operation_commutative  
        result = execute_dic(dic)  
        if isinstance(result, Table):  
            result.show()  
        else:  
            # MULTIPLE CONDITIONS  
            dicts = []  
            for q in queries:  
                dicts.append(interpret(q))  
            results = []  
            for d in dicts:  
                results.append(execute_dic(d).get_result())  
            outputs = results[0]  
            for i in range(1, len(results)):  
                outputs = outputs.union(results[i])  
            if __name__ == "__main__":  
                while 1:  
                    by = dic
```

```
dic = {dict: 7} (select: *, from: (select: *, from: student, where: tot_cred=56, distinct: None, order by: None, limit: None))  
select = (int) 7  
from = (dict: 7) (select: *, from: student, where: tot_cred=56, distinct: None, order by: None, limit: None)  
select = (int) 7  
where = (int) tot_cred=56  
distinct = (NoneType) None  
order by = (NoneType) None  
limit = (NoneType) None  
desc = (NoneType) None  
_len_ = (int) 7  
where = (int) dept_name = "physics"  
distinct = (NoneType) None  
order by = (NoneType) None  
limit = (NoneType) None  
desc = (NoneType) None  
_len_ = (int) 7
```



```
def selection_operation_commutative(dic):  
    if queries[i] != queries[i+1]:  
        queries[i] = queries[i+1] + '  
    initial_parts = '.join(initial_part'  
    for i in range(0, len(queries)):  
        queries[i] = initial_part + 'queries[i]  
    if len(queries) == 1:  
        # ONLY ONE CONDITION  
        dic = interpret(line)  
        print("") # before selection_operation_commutative  
        selection_operation_commutative(dic)  
        print("") # after selection_operation_commutative  
        result = execute_dic(dic)  
        if isinstance(result, Table):  
            result.show()  
        else:  
            # MULTIPLE CONDITIONS  
            dicts = []  
            for q in queries:  
                dicts.append(interpret(q))  
            results = []  
            for d in dicts:  
                results.append(execute_dic(d).get_result())  
            outputs = results[0]  
            for i in range(1, len(results)):  
                outputs = outputs.union(results[i])  
            if __name__ == "__main__":  
                while 1:  
                    by = dic
```

```
dic = {dict: 7} (select: *, from: (select: *, from: student, where: tot_cred=56, distinct: None, order by: None, limit: None))  
select = (int) 7  
from = (dict: 7) (select: *, from: student, where: dept_name = "physics", distinct: None, order by: None, limit: None)  
select = (int) 7  
from = (int) student  
where = (int) dept_name = "physics"  
distinct = (NoneType) None  
order by = (NoneType) None  
limit = (NoneType) None  
desc = (NoneType) None  
_len_ = (int) 7  
where = (int) tot_cred=56  
distinct = (NoneType) None  
order by = (NoneType) None  
limit = (NoneType) None  
desc = (NoneType) None  
_len_ = (int) 7
```

Όλοι οι επόμενοι κανόνες ΣΑ ακολουθούν ίδια λογική, αλλαγή στο `dic`.

Δεν έχουν γίνει τα ΣΑ με `union` γιατί δεν υποστηρίζονται στο συγκεκριμένο dbms.