

Εργασία για Συστήματα Διαχείρισης Βάσεων Δεδομένων

ΜΠΟΧΤΗΣ ΧΑΡΙΣΗΣ Π04094

ΣΦΑΛΑΓΚΑΚΟΣ ΓΙΩΡΓΟΣ Π20241

1.AND, OR , NOT implementations

And:

Αρχικά δημιουργήσαμε το index1 και index που χρησιμοποιούμε και για το AND και για το OR. Κάνουμε ένα if statement για να δούμε αν υπάρχει "and" μέσα στο condition_list, αν υπάρχει τότε αλλάζει τον index1 μεταβλητή σε True και συνεχίζει στο if index1 == True: (INSERT PICTURE).

```
# cast the value with the specified column's type and
isnot = True
index1 = False
index2 = False
index = []
condition_list = condition.split()
...

we look for 'and' within the condition_list, if we find
something similar happens with the elif line but instead
...

if "and" in condition_list :
    index = condition_list.index('and')
    index1 = True
elif "or" in condition_list:
    index = condition_list.index('or')
    index2 = True
...
```

Στην συνέχεια τρέχουμε δυο εντολές που χωρίζουν το condition_list σε δυο κομμάτια. Το κομμάτι πριν το AND και το κομμάτι μετά το and, και στην συνέχεια δημιουργούμε άλλους δυο μεταβλητές και χρησιμοποιούμε την method "self._parse_condition". μετά κάνουμε assign το left_value[-1] και right_value[-1] στο x. και μετά συνεχίζουμε κάνοντας assign τα ονόματα των 2 column στους μεταβλητές y και w

```
if index1 == True: #THIS IS AND
    left_condition = " ".join(condition_list[:index])
    right_condition = " ".join(condition_list[index+1:])
    left_value = self._parse_condition(left_condition)
    right_value = self._parse_condition(right_condition)
    left_value = left_value[:-1]
    right_value = right_value[:-1]
    coltype = self.column_types[self.column_names.index(left_value[0])]
    x = (left_value[-1], right_value[-1])
    y = left_value[1]
    w = (right_value[0])
    op = "="
    isnot = True
    return y,w, op, coltype(x), isnot
```

Στην συνέχεια

αφού καλέσουμε το `self._parse_condition(condition)` στην `_select_where()` μέθοδο χωρίζουμε το `condition` πάλι με `divider` το `and` και το κάνουμε `assign` σε μεταβλητές. Παίρνουμε τους μεταβλητές και κάνουμε `strip` όλα τα κενά ώστε να έχουμε μια λίστα για κάθε `condition`, κάθε λίστα να έχει 3 items πχ(`'credits', '=', 4`) όπου το `w/u` να είναι τα `values` που ψάχνουμε, τα `operator1/operator2` να είναι τα `operators` που θέλουμε να χρησιμοποιήσουμε. Αν και τα δυο είναι `True` τότε αποθηκεύουμε τα `index` των σωστών σειρών στο `rows` και το εκτυπώνουμε.

```
(smdb)> select * from student where dept_name=physics and tot_cred=46
these are the correct Indexes for the rows [5]
  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
      76543   brown      comp. sci.         58
      76653   aoi        elec. eng.         60
      98765   bourikas   elec. eng.         98
      98988   tanaka     biology          120

(smdb)> |
```

OR:

Το `Or` δουλεύει παρομοια με το `AND` όπου πρώτα βλέπουμε αν υπάρχει το `OR` μέσα στο `condition`, αν ναι τωπά μπαίνει στο `if statement` που έχουμε δημιουργήσει, δημιουργούμε άλλες δυο μεταβλητές και χρησιμοποιούμε την `method "self._parse_condition"`. μετά κάνουμε `assign` το `left_value[-1]` και `right_value[-1]` στο `x`. και μετά συνεχίζουμε κάνοντας `assign` τα ονόματα των 2 `column` στις μεταβλητές `y` και `w`.

```
...
we look for 'and' within the condition_list,
something similar happens with the elif line
...
if "and" in condition_list :
    index = condition_list.index('and')
    index1 = True
elif "or" in condition_list:
    index = condition_list.index('or')
    index2 = True
...
here we check if there is 'and', 'or' stateme
```

```

elif index2 == True:# THIS IS OR
    left_condition = " ".join(condition_list[:index])
    right_condition = " ".join(condition_list[index+1:])
    left_value = self._parse_condition(left_condition)
    right_value = self._parse_condition(right_condition)
    left_value= left_value[:-1]
    right_value= right_value[:-1]
    coltype = self.column_types[self.column_names.index(left_value[0])]
    x = (left_value[-1], right_value[-1])
    y = left_value[1]
    w = (right_value[0])
    op = "="
    isnot = True
    return y,w, op, coltype(x), isnot

```

Μετά όταν μπαίνουμε στο `_select_where()` και καλούμε την `self._parse_condition(condition)`. χωρίζουμε το `condition` πάλι με `divider` το `and` και το κάνουμε `assign` σε μεταβλητές, κάνουμε `strip` όλα τα κενά ώστε να έχουμε μια λίστα για κάθε `condition`. κάθε λίστα να έχει 3 items `πχ('dept_name', '=', 'student')` οπότε το `w/u` να είναι τα `values` που ψάχνουμε, τα `operator1/operator2` να είναι τα `operators` που θέλουμε να χρησιμοποιήσουμε. Αν ένα από τα `duo rows(rows1, rows2)` είναι `True` τότε αποθηκεύουμε τα `index` των σωστών σειρών στο `rows` και το εκτυπώνουμε και τα εκτυπώνουμε.

```

elif isnot == True and len(condition.split('or'))>1:
    left_condition, right_condition = condition.split('or')
    left_value, right_value = self._parse_condition(left_condition.strip()), self._parse_condition(right_condition.strip())
    left_value, right_value = left_value[:-1], right_value[:-1]
    operator1, operator2 = left_value[-2], right_value[-2]
    w, u = left_value[-1], right_value[-1]
    rows1 = [ind for ind, x in enumerate(column1) if get_op(operator1, x, w)]
    rows2 = [ind for ind, y in enumerate(column2) if get_op(operator2, y, u)]
    rows = list(set(rows1) | set(rows2)) #compares rows1 and rows2 and prints either one if it is correct
    print("these are the correct Indexes for the rows", rows)

```

```

(smdb)> select * from course where dept_name=biology OR credits=4
these are the correct Indexes for the rows [0, 1, 2, 3, 4, 12]

```

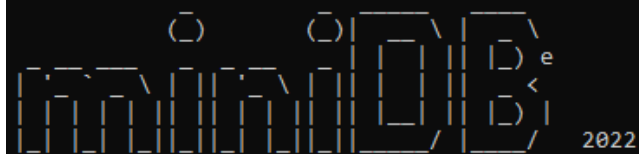
course_id (str)	#PK#	title (str)	dept_name (str)	credits (int)
bio-101		intro. to biology	biology	4
bio-301		genetics	biology	4
bio-399		computational biology	biology	3
cs-101		intro. to computer science	comp. sci.	4
cs-190		game design	comp. sci.	4
cs-315		robotics	comp. sci.	3
cs-319		image processing	comp. sci.	3
cs-347		database system concepts	comp. sci.	3
ee-181		intro. to digital systems	elec. eng.	3
fin-201		investment banking	finance	3
his-351		world history	history	3
mu-199		music video production	music	3
phy-101		physical principles	physics	4

NOT

Στο table.py στη μεθοδο parse_condition ορίσαμε με τη βοήθεια της split condition αριστερο και δεξι μέρος , βάλαμε μια μεταβλητη default false και μετα ελεγχουμε το μεγεθος της result να είναι μεγαλύτερο του 1 για να μην είναι κενή η συθηκη και αποδιδουμε την δευτερη τιμη του πινακα στο left

```
        return y,w, op, coltype(x), isnot
    elif index1 == False and index2 == False:
        left, op, right = split_condition(condition)
        result1 = left.split(" ")
        isnot = False
        if len(result1) > 1 and result1[0] == "not":#THIS IS NOT
            left = result1[1]
            isnot = True
            x = left
            y = right
        if left not in self.column_names:
            raise ValueError(f'Condition is not valid (cant find column name)')
        coltype = self.column_types[self.column_names.index(left)]
        x = left
        y = left
        return x,y, op, coltype(right), isnot
```

Και μετα καλειται στο select_where οπου προσθεσα ενα not στο rows που προυπηρχε μιας και ετσι λειτουργει το not. Για να βγαζει τα παντα εκτος απο αυτο που γραφουμε στο sql query:
Παραδειγμα στην επομενη photo.



```
(smdb)> select * from course
course_id (str) #PK#    title (str)                dept_name (str)    credits (int)
-----
bio-101                intro. to biology    biology            4
bio-301                genetics             biology            4
bio-399                computational biology biology            3
cs-101                 intro. to computer science comp. sci.        4
cs-190                 game design          comp. sci.        4
cs-315                 robotics             comp. sci.        3
cs-319                 image processing     comp. sci.        3
cs-347                 database system concepts comp. sci.        3
elec-181               intro. to digital systems elec. eng.        3
Fin-201                investment banking   finance            3
his-351                world history        history            3
mu-199                 music video production music              3
phy-101                physical principles  physics            4

(smdb)> select * from course where not credits<4
course_id (str) #PK#    title (str)                dept_name (str)    credits (int)
-----
bio-101                intro. to biology    biology            4
bio-301                genetics             biology            4
cs-101                 intro. to computer science comp. sci.        4
cs-190                 game design          comp. sci.        4
phy-101                physical principles  physics            4

(smdb)> _
```

2a BTREE

Αρχικά φτιάξαμε δυο κλάσεις row και table στο btree.py

```
class Row:
    def __init__(self, table, values):
        self.table = table
        self.values = values

class Table:
    def __init__(self, name, columns):
        self.name = name
        self.columns = columns
        self.rows = []
        self.primary_key_index = None
        self.unique_indexes = {}

    def insert(self, values):
        if len(values) != len(self.columns):
            raise ValueError("the number of values is not the same as columns number")
        row = Row(self, values)
        for column in self.columns:
            if column.is_primary_key:
                if self.primary_key_index is None:
                    self.primary_key_index = Btree(column)
                    self.primary_key_index.insert(row)
            elif column.is_unique:
                if column.name not in self.unique_indexes:
                    self.unique_indexes[column.name] = Btree(column)
                    self.unique_indexes[column.name].insert(row)
                self.rows.append(row)

    def select(self, column_names=None, where=None):
        if where is None:
            rows = self.rows
        else:
            rows = []
            for row in self.rows:
                if where.matches(row):
                    rows.append(row)
        if column_names is None:
            return [row.values for row in rows]
        else:
            indices = [self.get_index(column_name) for column_name in column_names]
            result = []
            for row in rows:
                result.append([row.values[index] for index in indices])
            return result
```

Στη συνέχεια φτιάξαμε create table users με unique columns

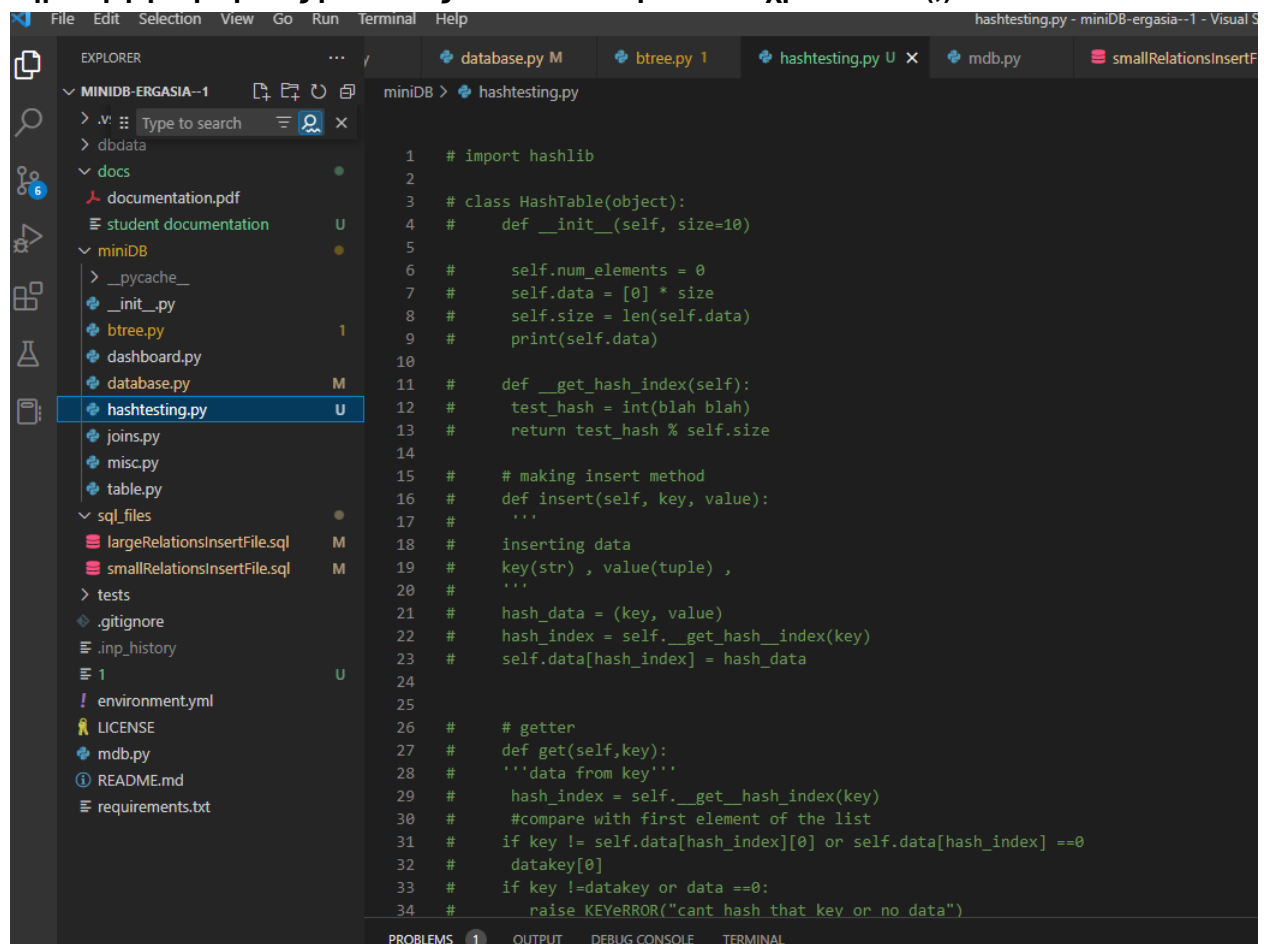
```
11 create table prereq (course_id str, prereq_id str);
12 CREATE TABLE users (id int primary key,username str UNIQUE,email str UNIQUE,age int);
13 insert into classroom values (Packard,101,500);
```

Και στη συνέχεια κάνουμε insert into values

```
150 insert into prereq values (1,101,101);
151 insert into users values (1, 'harisbohtis', 'haris@gmail.com', 35);
152 insert into users values (2, 'georgesfalagakos', 'george@gmail.com', 20);
153 insert into users values (3, 'mariapapadopoulou', 'mariap@yahoo.com', 25);
```

2.b

Τελος δοκιμάσαμε το Hash index over PK or unique columns αλλα εξαιτίας έλλειψης χρόνου και επειδή ήμασταν 2 άτομα και όχι 3 φτιάξαμε ενα γενικο πλάνο που θα ακολουθούσαμε το οποίο και παραθέτουμε σαν comments μέσα στο hashtesting.py που δημιουργήσαμε με τις μεθόδους που πιστεύαμε ότι θα χρειαστούν(;



```
miniDB > hashtesting.py
1  # import hashlib
2
3  # class HashTable(object):
4  #     def __init__(self, size=10)
5
6  #         self.num_elements = 0
7  #         self.data = [0] * size
8  #         self.size = len(self.data)
9  #         print(self.data)
10
11 #     def __get_hash_index(self):
12 #         test_hash = int(blah blah)
13 #         return test_hash % self.size
14
15 #     # making insert method
16 #     def insert(self, key, value):
17 #         '''
18 #         inserting data
19 #         key(str) , value(tuple) ,
20 #         '''
21 #         hash_data = (key, value)
22 #         hash_index = self.__get_hash_index(key)
23 #         self.data[hash_index] = hash_data
24
25
26 #     # getter
27 #     def get(self, key):
28 #         '''data from key'''
29 #         hash_index = self.__get_hash_index(key)
30 #         #compare with first element of the list
31 #         if key != self.data[hash_index][0] or self.data[hash_index] == 0
32 #         datakey[0]
33 #         if key != datakey or data == 0:
34 #             raise KEYERROR("cant hash that key or no data")
```

