Πανεπιστήμιο Πειραιώς
Εργαστήριο Επιστήμης Δεδομένων
www.datastories.org

---

miniDB Documentation

v.1.0 (Nov. 2020)

---

1. Εισαγωγή - η βασική λειτουργικότητα

Η miniDB είναι μια πλατφόρμα διαχείρισης σχεσιακών δεδομένων, γραμμένη εξ ολοκλήρου σε Python. Ο βασικότερος σκοπός της είναι να αποδώσει μέρος της λειτουργικότητας των μεγάλων και ευρέως διαδεδομένων RDBMS, όπως η PostgreSQL, με ένα απλό και εύκολο στην κατανόηση και τον περαιτέρω εμπλουτισμό κώδικα. Έτσι, η miniDB απευθύνεται κυρίως σε φοιτητές και ερευνητές που επιθυμούν να δουλέψουν με ένα εργαλείο το οποίο μπορούν να παραμετροποιήσουν όπως θέλουν, γρήγορα και εύκολα, αλλά και στο οποίο μπορούν να εντάξουν νέες υλοποιήσεις, εξίσου εύκολα και γρήγορα.

Παρακάτω, εξηγούμε βήμα βήμα τις εντολές που υποστηρίζει η miniDB, τις οποίες και τις παραλληλίζουμε με τις αντίστοιχες της γλώσσας SQL, με σκοπό ένας χρήστης να κατανοήσει πώς μπορεί να διαχειριστεί μια ΒΔ όπως θα την διαχειριζόταν με την βοήθεια ενός SQL-enabled εργαλείου.

SQL	miniDB code (python3)
Εντολές δημιουργίας, διαγραφής, φόρτωσης, αποθήκευσης ΒΔ	
CREATE DATABASE name	db_object = Database(name, load=False)
DROP DATABASE name	db_object.drop_db(); del db_object
LOAD DATABASE name	db_object = Database(name, load=True)
SAVE DATABASE name	db_object.save() # no name needed
Εντολές δημιουργίας, διαγραφής, αλλαγής, φόρτωσης, εξαγωγής πίνακα	
CREATE TABLE name (column_name1 column_type1, …)	db_object.create_table(name, [column_names], [column_types])

DROP TABLE name	db_object.drop_table(name)
ALTER TABLE name ALTER COLUMN column TYPE INTEGER(int)	db_object.cast_column(name, column, int)
CREATE TABLE name (column_name1 column_type1, …); COPY name FROM csv;	db_object.table_from_csv(name, column_names, column_types)
EXPORT TABLE name TO csvname.csv;	db_object.table_to_csv(name, csvname.csv)
Εντολές δημιουργίας, διαγραφής ευρετηρίου	
CREATE INDEX name ON table(column);	db_object.create_index(table, type, name) # type = ‘Btree’; no column needed, only pk column is valid
DROP INDEX name;	db_object.drop_index(name)
Εντολές εισαγωγής, διαγραφής, τροποποίησης εγγραφών πίνακα	
INSERT INTO table VALUES (row);	db_object.insert(table, row)
DELETE FROM table WHERE condition;	db_object.delete(table, condition) # condition is of type “column op value”
UPDATE table SET column=value WHERE condition;	db_object.update(table, value, column, condition)
Εντολές αναζήτησης σε πίνακα(-ες)	
SELECT columns/* FROM table WHERE condition [TOP k] [ORDER BY column [ASC|DESC]] [SAVE AS table2];	db_object.select(table, [columns/’*’], condition, order_by=column, asc=False/True, top_k=k, save_as=table2)
SELECT * FROM table1 INNER JOIN table2 ON condition;	db_object.inner_join(table1,table2, condition)
SELECT columns FROM table1 INNER JOIN table2 ON condition1 WHERE condition2 [TOP k] [ORDER BY column [ASC|DESC]] [SAVE AS table3];	db_object.inner_join(table1,table2, condition, return_object=True)._select_where(columns, condition, order_by=column, asc=False/True, top_k=k, save_as=table3)
Εντολές κλειδώματος / ξεκλειδώματος πινάκων	
BEGIN WORK; LOCK TABLE name IN EXCLUSIVE MODE; ... COMMIT WORK;	db_object.lockX_table(name) ... db_object.unlock_table(name)

2. Ένα παράδειγμα χρήσης της miniDB

Σε αυτή την ενότητα, θα δείξουμε πώς μπορούμε να φορτώσουμε, και κατόπιν να επεξεργαστούμε, μία ενδεικτική ΒΔ με βάση τα παραδείγματα smallRelationsInsertFile.sql και largeRelationsInsertFile.sql που περιλαμβάνονται στο συνοδευτικό υλικό του βιβλίου Silberschatz et al., Database System Concepts, 6/e (εδώ). Αυτό που επιχειρούμε δηλαδή είναι να μετατρέψουμε τον κώδικα SQL που υπάρχει εκεί σε Python ώστε να λειτουργήσει στη MiniDB.

Ενδεικτικά, ο κώδικας που δημιουργεί τη ΒΔ με όνομα “vsmdb” (VerySMallDataBase), δημιουργεί τον πίνακα “classroom” στον οποίο κατόπιν εισάγει 5 εγγραφές είναι ο παρακάτω:

```
from database import Database
# create db with name "vsmdb"
db = Database('vsmdb', load=False)
# create a single table named "classroom"
db.create_table('classroom', ['building', 'room_number', 'capacity'],
[str,str,int])
# insert 5 rows into classroom
db.insert('classroom', ['Packard', '101', '500'])
db.insert('classroom', ['Painter', '514', '10'])
db.insert('classroom', ['Taylor', '3128', '70'])
db.insert('classroom', ['Watson', '100', '30'])
db.insert('classroom', ['Watson', '120', '50'])
```

Με τον ίδιο τρόπο μπορούμε να συμπληρώσουμε και το υπόλοιπο σχήμα της ΒΔ.

Αρχικά, μπορούμε να δούμε το περιεχόμενο του πίνακα *'classroom'* με την παρακάτω select εντολή:

```
>>> db.select('classroom', '*')

## classroom ##
building (str)         room_number (str)    capacity (int)
---------------     -------------------  ----------------
Packard                            101                500
Painter                            514                 10
Taylor                            3128                 70
Watson                             100                 30
Watson                             120                 50
```

Μπορούμε επίσης να επιλέξουμε μόνο της αίθουσες με capacity>100 με την παρακάτω εντολή:

```
>>> db.select('classroom', '*', 'capacity>100')

## classroom ##
building (str)      room_number (str)     capacity (int)
---------------  -------------------   ----------------
Packard                          101                500
```

Έπειτα, μπορούμε να διαγράψουμε τις εγγραφές με capacity>100 και να ελέγξουμε τόσο την νέα κατάσταση του πίνακα και των meta πινάκων insert_stack και length (η λειτουργικότητα των meta-πινάκων θα περιγραφεί στη συνέχεια).

```
>>> db.delete('classroom', 'capacity>100')
del 0
Deleted 1 rows
>>> db.select('classroom', '*')

## classroom ##
building (str)         room_number (str)   capacity (int)
---------------    -------------------  ----------------
Painter                             514                10
Taylor                             3128                70
Watson                              100                30
Watson                              120                50

>>> db.meta_insert_stack.show()

## meta_insert_stack ##
table_name (str)    indexes (list)
------------------  ----------------
classroom           [0]

>>> db.meta_length.show()

## meta_length ##
table_name (str)      no_of_rows (int)
------------------  ------------------
classroom                            4
```

Τέλος, μπορούμε να εισάγουμε μια νέα εγγραφή και να ελέγξουμε τόσο την νέα κατάσταση του πίνακα και των meta πινάκων insert_stack και length.

```
>>> db.insert('classroom', ['Hewlett', '102', 420])
>>> db.select('classroom', '*')

## classroom ##
building (str)        room_number (str)    capacity (int)
----------------    ------------------  ----------------
Hewlett                            102                420
Painter                            514                 10
Taylor                            3128                 70
Watson                             100                 30
Watson                             120                 50

>>> db.meta_insert_stack.show()

## meta_insert_stack ##
table_name (str)    indexes (list)
------------------  ----------------
classroom           []

>>> db.meta_length.show()

## meta_length ##
table_name (str)      no_of_rows (int)
------------------  ------------------
classroom                           5
```

Όπως συμβαίνει και στα πραγματικά ΣΔΒΔ, παράλληλα με τους πίνακες το σύστημα συντηρεί και μετα-πίνακες (metatables). Η miniDB υποστηρίζει τους εξής metatables:

- meta_length: τηρεί πληροφορία σχετικά με τις εγγραφές των πινάκων
- meta_insert_stack: τηρεί πληροφορία σχετικά με τις “κενές” εγγραφές των πινάκων (λόγω διαγραφών που έχουν προηγηθεί)
- meta_locks: τηρεί πληροφορία σχετικά με τα κλειδώματα των πινάκων

- meta_indexes: τηρεί πληροφορία σχετικά με τα ευρετήρια των πινάκων

Παρακάτω βλέπουμε τα metatables που αφορούν τη vsmdb, αφού έχουμε δημιουργήσει και φορτώσει με εγγραφές τον πίνακα classroom.

```
## meta_indexes ##
table_name (str)    index_name (str)
------------------  ------------------


## meta_insert_stack ##
table_name (str)    indexes (list)
------------------  ----------------
classroom           []


## meta_length ##
table_name (str)      no_of_rows (int)
------------------  ------------------
classroom                           5


## meta_locks ##
table_name (str)    locked (bool)
------------------  ---------------
classroom           False
```

3. Αναλυτική τεκμηρίωση της miniDB

Παρακάτω εξηγούμε τις λειτουργίες των μεθόδων από τις οποίες αποτελείται ο κώδικας της miniDB. Για κάθε μέθοδο, παρατίθεται το όνομά της, τα ορίσματά της και μια σύντομη περιγραφή της λειτουργίας της. Οι μέθοδοι οργανώνονται ως εξής:

- MiniDB.database (ενότητα 3.1): περιλαμβάνει μεθόδους οι οποίες αφορούν το I/O της βάσης (π.χ. select, delete κλπ) (ενότητα 3.1.1) και την εσωτερική διαχείριση αυτής (ενότητα 3.1.2)
- MiniDB.table (ενότητα 3.2): περιλαμβάνει μεθόδους οι οποίες αφορούν το αντικείμενο table.
- MiniDB.btree (ενοτητα 3.3): περιλαμβάνει μεθόδους οι οποίες αφορούν τα αντικείμενα btree και node (btree node).
- MiniDB.misc (ενότητα 3.4): περιλαμβάνει βοηθητικές μεθόδους.

3.1. MiniDB.database

Σε αυτή την ενότητα παρουσιάζονται οι μέθοδοι που σχετίζονται με τη λειτουργικότητα σε επίπεδο ΒΔ (αρχικοποίηση, τήρηση meta-tables κλπ.).

3.1.1. I/O functions

database.cast_column(self, table_name, column_name, cast_type)

Wrapper function της table._cast_column. Φορτώνει μέσω της database.load το τελευταίο checkpoint της βάσης, ελέγχει αν ο πίνακας με όνομα table_name είναι κλειδωμένος και αν δεν είναι, τον κλειδώνει και καλεί την table._cast_column. Τέλος, ξεκλειδώνει τον πίνακα table_name και αποθηκεύει ένα νέο checkpoint της βάσης.

database.insert(self, table_name, row)

Wrapper function της table._insert. Φορτώνει μέσω της database.load το τελευταίο checkpoint της βάσης, καθώς και το insert_stack του πίνακα table_name (μέσω της database._get_insert_stack_for_table). Ελέγχει αν ο πίνακας με όνομα table_name είναι κλειδωμένος και αν δεν είναι, τον κλειδώνει και καλεί την table._insert. Τέλος, ξεκλειδώνει τον πίνακα table_name και αποθηκεύει ένα νέο checkpoint της βάσης.

database.update(self, table_name, set_value, set_column, condition)

Wrapper function της table._update_row. Φορτώνει μέσω της database.load το τελευταίο checkpoint της βάσης, ελέγχει αν ο πίνακας με όνομα table_name είναι κλειδωμένος και αν δεν είναι, τον κλειδώνει και καλεί την table._update_row. Τέλος, ξεκλειδώνει τον πίνακα table_name και αποθηκεύει ένα νέο checkpoint της βάσης.

database.delete(self, table_name, condition)

Wrapper function της table._delete_where. Φορτώνει μέσω της database.load το τελευταίο checkpoint της βάσης, ελέγχει αν ο πίνακας με όνομα table_name είναι κλειδωμένος, και αν δεν είναι, τον κλειδώνει και καλεί την table._delete_where. Η table._delete_where επιστρέφει τις θέσεις των εγγραφών που διέγραψε, οι οποίες προστίθεται στο insert stack του πίνακα table_name (μέσω της database._add_to_insert_stack). Έτσι, ένα index μπορεί να είναι consistent, αφού σε καμία περίπτωση ένα row δεν θα αλλάξει θέση (index), κάτι που θα αχρήστευε το υπάρχον ευρετήριο (βλ. Btree). Π.Χ.: Στην λίστα [0,1,2,3], κάθε value είναι και το index της στην λίστα (στην python μετράμε από το 0). Αν κάνουμε pop το 1, η λίστα θα γίνει [0,2,3]. Συνεπώς τα στοιχεία 2,3 θα έχουν index 1 και 2. Αυτό το φαινόμενο δημιουργεί inconsistencies και είναι απαγορευτικό στην περίπτωσή μας.Τέλος, ξεκλειδώνει τον πίνακα table_name και αποθηκεύει ένα νέο checkpoint της βάσης.

database.select(self, table_name, columns, condition, order_by, asc,top_k, save_as, return_object)

Wrapper function της table._select_where ή της table._select_where_with_btree (αν ο πίνακας table_name έχει index στο primary key του). Φορτώνει μέσω της database.load το τελευταίο checkpoint της βάσης και ελέγχει αν ο πίνακας με όνομα table_name είναι κλειδωμένος. Αν δεν είναι, τον κλειδώνει και καλεί την κατάλληλη select (με ή χωρίς index). Επιστρέφει και κάνει show τον πίνακα αν return_object=False, αλλιώς επιστρέφει τον πίνακα σαν αντικείμενο table. Αν save_as!=None, αποθηκεύει τον πίνακα στη βάση με το όνομα save_as. Τέλος, ξεκλειδώνει τον πίνακα table_name και αποθηκεύει ένα νέο checkpoint της βάσης.

database.show_table(self, table_name, no_of_rows)

Wrapper function της table.show. Φορτώνει μέσω της database.load το τελευταίο checkpoint της βάσης και καλεί την table.show(). Σημείωση: η database.show_table δεν υπακούει στα κλειδώματα (χρήσιμο π.χ. για live dashboard).

database.sort(self, table_name, column_name, asc)

Wrapper function της table._sort. Φορτώνει μέσω της database.load το τελευταίο checkpoint της βάσης, ελέγχει αν ο πίνακας με όνομα table_name είναι κλειδωμένος και αν δεν είναι, τον κλειδώνει και καλεί την table._sort. Τέλος, ξεκλειδώνει τον πίνακα table_name και αποθηκεύει ένα νέο checkpoint της βάσης.

database.inner_join(self, left_table_name, right_table_name, condition, save_as, return_object)

Wrapper function της table._inner_join. Φορτώνει μέσω της database.load το τελευταίο checkpoint της βάσης και ελέγχει αν ο πίνακας με όνομα table_name είναι κλειδωμένος. Αν δεν είναι, καλεί την table_inner_join. Επιστρέφει και κάνει show τον πίνακα αν return_object=False, αλλιώς επιστρέφει τον πίνακα σαν αντικείμενο table. Αν save_as!=None, αποθηκεύει τον πίνακα στη βάση με το όνομα save_as. Τέλος, ξεκλειδώνει τον πίνακα table_name και αποθηκεύει ένα νέο checkpoint της βάσης.

3.1.2. Database management functions

database.__init__(self, dbname, load)

Function που τρέχει όταν αρχικοποιούμε το database object. Δημιουργεί τα directories dbdata και dbdata/{dbname}_db, αν δεν υπάρχουν. Αν υπάρχει directory με το ίδιο όνομα (dbdata/{dbname}_db) και load=True, φορτώνει την ήδη υπάρχουσα βάση. Αν load=False, την αντικαθιστά. Τέλος, αρχικοποιεί τους meta-πίνακες meta_length, meta_locks, meta_insert_stack και meta_indexes. Αυτοί οι meta-πίνακες έχουν την παρακάτω λειτουργικότητα:

- meta_length: πίνακας που αποθηκεύει τα ονόματα των πινάκων της ΒΔ και τον αριθμό των εγγραφών (rows) τους.
- meta_locks: πίνακας που αποθηκεύει τα ονόματα των πινάκων της ΒΔ και, για τον καθένα ένα boolean που δείχνει αν ο συγκεκριμένος πίνακας είναι exclusively locked (X lock) ή όχι.

- meta_insert_stack: πίνακας που αποθηκεύει τα ονόματα των πινάκων της ΒΔ και, για τον καθένα μια λίστα (μορφής στοίβας) με τον δείκτη (ptr) των rows που είναι “κενά” (επισημασμένα με την ειδική τιμή None σε κάθε column).
- meta_indexes: πίνακας που αποθηκεύει τα ονόματα των πινάκων της ΒΔ και τα ονόματα των αντιστοιχων ευρετηρίων (τύπου Btree index), αν υπάρχουν. Κάθε πίνακας μπορεί να διαθέτει το πολύ ένα ευρετήριο, αφού index μπορεί να φτιαχτεί μόνο στο primary key του πίνακα.

database._save_locks()
Αποθηκεύει τον πίνακα meta_locks στο αρχείο meta_locks.pkl, στο directory dbdata/{dbname}_db.

database._update()
Ενημερώνει τους πίνακες meta_length, meta_locks, meta_insert_stack εκτελώντας τις αντίστοιχες functions _update_meta_length, _update_meta_locks, _update_meta_insert_stack.

database._update_meta_length(self)
Ενημερώνει τον πίνακα meta_length εκτελώντας την αντίστοιχη function _update_meta_length. Συγκεκριμένα, εισάγει εγγραφές για τυχόν νέους πίνακες και έπειτα, για τον καθένα υπολογίζει και αποθηκεύει τον αριθμό εγγραφών που δεν είναι επισημασμένοι ως deleted ([None, None, ..]).

database._update_meta_locks(self)
Ενημερώνει τον πίνακα meta_locks εκτελώντας την αντίστοιχη function _update_meta_locks. Συγκεκριμένα, εισάγει εγγραφές για τυχόν νέους πίνακες στον πίνακα meta_locks.

database._update_meta_insert_stack(self)
Ενημερώνει τον πίνακα meta_insert_stack εκτελώντας την αντίστοιχη function _update_meta_insert_stack. Συγκεκριμένα, εισάγει εγγραφές για τυχόν νέους πίνακες στον πίνακα meta_insert_stack.

database._add_to_insert_stack(self, table_name, ptr)
Προσθέτει (append) τα δεδομένα της λίστας ptr στο insert stack του πίνακα table_name.

database._get_insert_stack_for_table(self, table_name)
Επιστρέφει το insert stack του πίνακα table_name

database._update_insert_stack_for_tb(self, table_name, new_stack)
Αντικαθιστά (replace) το insert stack του πίνακα table_name με το new_stack.

database._construct_index(self, table_name, index_name)
Δημιουργεί ένα ευρετήριο B+tree με όνομα index_name πάνω στο primary key column του πίνακα table_name.

database._has_index(self, table_name)
Επιστρέφει true/false αν ο πίνακας table_name έχει / δεν έχει index πάνω στο primary key του.

database._save_index(self, table_name, index_name, index)
Δημιουργεί το directory dbdata/{dbname}_db/indexes, αν δεν υπάρχει, και αποθηκεύει το ευρετήριο “index” (object) στο αρχείο meta_{table_name_index_name}_index.pkl. Επειδή τεχνικά δεν επιτρέπεται δύο διαφορετικά ευρετήρια μέσα στη ΒΔ να έχουν το ίδιο όνομα συνίσταται η ονομασία ενός ευρετηρίου να είναι της μορφής tablename_columnname

database._load_idx(self, table_name, index_name)
Φορτώνει το index με όνομα table_name_index_name από το directory dbdata/{dbname}_db/indexes.

3.2. MiniDB.table

Σε αυτή την ενότητα παρουσιάζονται οι μέθοδοι που σχετίζονται με τη λειτουργικότητα σε επίπεδο πίνακα ΒΔ (ενημέρωση, επιλογή, σύνδεση κλπ.).

table.__init__(self, name, column_names, column_types, primary_key, load)
Function που τρέχει όταν αρχικοποιούμε το table object. Δημιουργεί ένα table object στο οποίο φορτώνει ένα ήδη υπάρχον dictionary ή ένα αρχείο pkl αν η παράμετρος load δεν ειναι None. Αν είναι None, αρχικοποιεί παραμέτρους του table object όπως το όνομα του πίνακα και τα ονόματα των columns (table.name, table.column1, …), την λίστα στην οποία αποθηκεύονται τα δεδομένα (table.data), στην οποία κάθε στοιχείο είναι μια λίστα που αναπαριστά ένα row του πίνακα και το index του column που έχει δηλωθεί ως primary key (αν υπάρχει).

table._update(self)
Ενημερώνει την λίστα columns με τα στοιχεία του εκάστοτε column για κάθε row στην λίστα data. Η λίστα columns αποθηκεύει τα στοιχεία του κάθε column στην θέση 0 για το 1ο, 1 για το 2ο κτλ. Έτσι, είναι εφικτή η προσπέλαση του column με το όνομα του, διατηρώντας το consistency του πίνακα.

table._cast_column(self, column_name, cast_type)
Αντικαθιστά τα στοιχεία του πίνακα που αντιστοιχούν στο column_name, με το cast_type(στοιχείο) για το κάθε στοιχείο. Π.χ., το στοιχείο ‘2’ με cast_type int αντικαθίσταται με το int(‘2’).

table._insert(self, row, insert_stack)
Εισάγει την λίστα row στα στοιχεία του πίνακα (table.data). Το κάθε στοιχείο του row αντικαθίσταται με το column_type του column στο οποίο εισάγεται (π.χ. αν το ‘1’ εισάγεται σε

ένα column με column_type=int, το '1' αντικαθίσταται με το int('1'). Το τελικό row (με τα σωστά types) εισάγεται στην τελευταία θέση (ptr) του insert stack αν αυτό δεν είναι κενό, αλλιώς εισάγεται στο τέλος.

table._update_row(self, set_value, set_column, condition)
Για κάθε row του πίνακα στο οποίο η συνθήκη condition είναι αληθής, αντικαθιστά το στοιχείο του set_column με το set_value. Για πληροφορίες για το condition, συμβουλευτείτε το misc.split_condition.

table._delete_where(self, condition)
Αντικαθιστά κάθε row του πίνακα στο οποίο η συνθήκη condition είναι αληθής με ένα row της μορφής <None, None, …>. Τέλος, επιστρέφει τις θέσεις (ptr) των rows που "διαγράφηκαν" με σκοπό αυτά να ενταχθούν στο insert_stack του πίνακα.

table._select_where(self, return_columns, condition, order_by, asc, top_k)
Επιστρέφει ένα table object το οποίο περιέχει τα columns που υπάρχουν στη λίστα return_columns (αν return_columns='*', επιστρέφει όλες τις στήλες) και τα rows που ικανοποιούν τη συνθήκη condition. Η μέθοδος που έχει υλοποιηθεί για την υλοποίηση της αναζήτησης είναι η γραμμική αναζήτηση (linear search). Αν η παράμετρος order_by οριστεί με το όνομα ενός column, κάνει διάταξη με βάση αυτό το column (asc=True/False σημαίνει αύξουσα / φθίνουσα σειρά). Αν έχει οριστεί η παράμετρος top_k, επιστρέφει τα πρώτα k rows, αλλιώς τα επιστρέφει όλα.

table._select_where_with_btree(self, return_columns, bt, condition, order_by, asc, top_k)
Όμοια με την table._select_where, με την διαφορά ότι κάνει χρήση ενός btree object (bt) για να εντοπίσει τα rows που ικανοποιούν τη συνθήκη condition. Εκτυπώνει τα αποτελέσματα (indexes όπου condition is true) και των δύο τεχνικών (με/χωρις btree) καθώς και τον αριθμό συνολικών συγκρίσεων μεταξύ στοιχείων, δείχνοντας την διαφορά στην απόδοση.

table.order_by(self, column_name, asc)
Επιστρέφει τον πίνακα με τα rows διατεταγμένα σε αύξουσα / φθίνουσα σειρά με βάση το column_name, αν asc=True/False.

table._sort(self, column_name, asc)
Όμοιο με το table.order_by(self, column_name, asc), το αποτέλεσμα ωστόσο αντικαθιστά τον υπάρχοντα πίνακα. Πρακτικά δηλαδή αλλάζει η σειρά των rows.

table._inner_join(self, table_right, condition)
Επιστρέφει ένα table_object το οποίο αποτελεί το προϊόν της εσωτερικής σύνδεσης (inner join) του πίνακα με τον πίνακα table_right, με βάση τη συνθήκη condition (που στη συγκεκριμένη περίπτωση πρέπει υποχρεωτικά να είναι της μορφής column1 op column2, όπου τα columns υπάρχουν στον table και τον table_right, αντίστοιχα). Η μέθοδος που έχει υλοποιηθεί για την υλοποίηση του inner join είναι η Nested-Loop-Join (NLJ).

table.show(self, no_of_rows, is_locked)
Εμφανίζει το περιεχόμενο του πίνακα σε μια φιλική στο χρήστη tabular μορφή. Εμφανίζει τα πρώτα no_of_rows αν η παράμετρος έχει τιμή, αλλιώς τα εμφανίζει όλα. Η παράμετρος

is_locked μπορεί να περαστεί ώς όρισμα ώστε να δείχνει δίπλα από τον εκάστοτε πίνακα αν αυτός είναι locked (## table_name (locked) ##)

table._parse_condition(self, condition, join)
Αυτή η μέθοδος δέχεται ένα condition με μορφή string και επιστρέφει το/τα columns που αυτή αφορά (ένα σε περιπτώσεις επιλογής, δύο σε περιπτώσεις inner join), τον τελεστή op (οι τελεστές που υποστηρίζονται είναι οι: <, <=, ==, >=, >) και, αν υπάρχει, το value (που μπορεί να είναι string, int κτλ) με το οποίο συγκρίνεται το column. Η σύνταξη των conditions που υποστηρίζονται είναι της μορφής:

- column op value, με ή χωρίς τα κενά
- column op column2, με ή χωρίς κενά, αν το όρισμα join=True

Τα values που επιστρέφονται γίνονται cast με τον τύπο του column (δεν παραμένουν δηλαδη πάντα string).

table._load_from_file(self, filename)
Φορτώνει ένα ήδη υπάρχον table_object από ένα αρχείο pkl (filename).

3.3. MiniDB.btree

Σε αυτή την ενότητα παρουσιάζονται οι μέθοδοι που σχετίζονται με τη λειτουργικότητα του ευρετηρίου Btree (κατασκευή ευρετηρίου, αναζήτηση κλπ.).

node.init(self, b, values, ptrs, left_sibling, right_sibling, parent, is_leaf)
Αρχικοποιεί το node abstraction, το οποίο συμβολίζει έναν Btree κόμβο (node). Οι παράμετροι είναι:

- b: branching factor. Όταν ένα node έχει ήδη b στοιχεία, τότε είναι γεμάτο και πρέπει να γίνει διάσπαση (split).
- values: οι τιμές που εισάγουμε στο node.
- ptrs: δείκτες (pointers) προς τα rows του πίνακα αν το node είναι φύλλο, δείκτες σε άλλα nodes αν το node δεν είναι φύλλο.
- left_sibling, right_sibling: (μόνο για φύλλα) δείκτης προς τον αριστερό και τον δεξιό, αντίστοιχα, αδελφό-κόμβο (sibling). None αν δεν έχει sibling ή αν δεν ειναι φύλλο.
- parent: (για όλα τα nodes εκτός της ρίζας) δείκτης τον πατέρα-κόμβο (parent) του node. None αν δεν εχει parent (δηλαδή, η ρίζα).
- is_leaf: true / false αν το node είναι / δεν είναι φύλλο.

node.find(self, value, return_ops)
Επιστρέφει το ptr προς τον επόμενο κόμβο στον οποίο θα πρέπει να συνεχιστεί η αναζήτηση για το value το οποίο ψάχνουμε. Αν το node είναι leaf (δηλαδή μετά από διαδοχικές αναζητήσεις έχουμε καταλήξει σε αυτό το φύλλο) δεν επιστρέφει τίποτα. Αν θέλουμε να μετρήσουμε τον αριθμό των συγκρίσεων (χρήσιμο για σύγκριση με sequential μεθόδους για παράδειγμα), ορίζουμε το return_ops σε True. Έτσι μαζί με το αποτέλεσμα του find, θα μας επιστραφεί και αριθμός των operations.

node.insert(self, value, ptr, ptr1)
Εισάγει το value και το ptr του στο node (αναζητά τη σωστή θέση και το εισάγει σε αυτή). Για περιπτώσεις που πρέπει να εισαχθούν 2 ptrs (π.χ. μετά από split), η μέθοδος που καλεί την node.insert μπορεί να ορίσει και την παράμετρο ptr1 (το 2ο ptr).

node.show(self)
Εκτυπώνει βασικά στοιχεία για το node. Συγκεκριμένα εκτυπώνει:

1. Τις τιμές των στοιχείων του.
2. Τα pointers των τιμών του.
3. Το pointer του γονέα του.
4. Το pointer του αριστερού του αδελφού.
5. Το pointer του δεξιού του αδελφού.

Btree.init(self, b)
Αρχικοποιεί το tree abstraction. Οι παράμετροι είναι:

- b: branching factor, - η μέγιστη χωρητικότητα ενός node είναι b-1 τιμές.
- nodes: λίστα που περιέχει όλα τα nodes (node objects) του δέντρου
- root: το ptr του root node (ρίζα)

Btree.insert(self, value, key, rkey)
Εισάγει το value και το/τα ptr/s του στο δεντρο. Αν το δέντρο είναι άδειο, εισάγει το 1ο node και ορίζει το ptr αυτού ως το root. Έπειτα, βρίσκει και (προσπαθεί να) εισαγει το value και το/τα ptrs στο σωστό node. Αν το node έχει ήδη αριθμό στοιχείων ίσο με b, καλεί την split για το συγκεκριμένο node.

Btree._search(self, value, return_ops)
Επιστρέφει το node στο οποίο υπάρχει ή θα έπρεπε να υπάρχει το value. Αν θέλουμε να μετρήσουμε το αριθμό των συγκρίσεων (χρήσιμο για σύγκριση με sequential μεθόδους για παράδειγμα), ορίζουμε το return_ops σε True. Έτσι μαζί με το αποτέλεσμα του _search, θα μας επιστραφεί και αριθμός των operations.

Btree.split(self, node_id)
Τρέχει όταν ένα node αποκτήσει αριθμό στοιχείων = b. Σπάει το node αυτό και φροντίζει να διατηρήσει την σωστή δομή του δέντρου.

Btree.show(self)
Εκτυπώνει βασικά χαρακτηριστικά για το κάθε node (node.show()) του δέντρου, με σειρά από πάνω προς τα κάτω και αριστερά προς τα δεξιά.

Btree.plot(self)
Εμφανίζει την οπτικοποίηση του δέντρου

Btree.find(self, operator, value)
Επιστρέφει τα ptrs των στοιχείων που ικανοποιούν την συνθήκη <btree_value op value>.

3.4. MiniDB.misc

Σε αυτή την ενότητα παρουσιάζονται οι μέθοδοι που σχετίζονται με διάφορες βοηθητικές λειτουργίες της miniDB.

misc.get_op(op, a, b):
Μεταφράζει το operator string (π.χ. '<=') σε μια python function (operator.le (less equal) εν προκειμένω) και επιστρέφει το αποτέλεσμα αυτής με τελεστέους τα a, b.
Π.χ.:

- '2<4' -> True
- 'Nick<John' -> False

misc.split_condition(condition):
"Σπάει" το string condition (π.χ. 'Nick<John') με βάση τον τελεστή σε 1ο τελεστέο, τελεστή και 2ο τελεστέο.
Π.χ.:

- '2<4' -> '2', '<', '4'
- 'Nick<John' -> 'Nick', '<', 'John'

3.5. MiniDB.dashboard

Dashboard script. Σκοπός του είναι να εκτελείται σε συνδυασμό με την unix εντολή watch ώστε εκτυπώνει σε πραγματικό χρόνο τους πίνακες της βάσης το ονομα της οποίας ο χρήστης πρέπει να περάσει ως πρώτη παράμετρο. Αν ο χρήστης προσθέσει και την λέξη "meta" ως δεύτερη παράμετρο, τότε θα εμφανιστούν μόνο τα meta tables της βάσης.

```
All tables
watch -n1 python3 dashboard.py {db_name}

Only meta tables
watch -n1 python3 dashboard.py {db_name} meta
```

3.6. MiniDB.btree_test

Btree testing script. Ο χρήστης επιλέγει πόσους τυχαίους αριθμούς (0 έως 100 - όχι διπλότυπους) θα εντάξει στο B+tree καθώς και το branching factor του (B).

```
B+tree με B=5 και 100 στοιχεία
python3 btree_test.py 100 5
```