ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ: Ιωάννης Θεοδωρίδης

ΒΟΗΘΟΙ: Ιωάννης Κοντούλης, Γεώργιος Θεοδωρόπουλος

DOCUMENTATION ΤΕΛΙΚΗΣ ΕΡΓΑΣΙΑΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ (miniDB)

ΓΚΟΛΕΜΙ ΚΡΙΣΤΙΑΝ, Π18029

ΜΙΧΑΗΛ ΚΑΤΣΟΥΛΑΣ, Π18071

ΓΙΩΡΓΟΣ ΜΑΡΚΟΖΑΝΗΣ, Π18098

IMPLEMENTATION OF TASK 1.4 (JOINS)

Η υλοποίηση του task που ανατέθηκε στην ομάδα μας έχει γίνει στα κύρια αρχεία της εφαρμογής και συγκεκριμένα στα table.py και database.py. Έγινε προσπάθεια για υλοποίηση σε εξωτερικά αρχεία, ωστόσο, λόγω πολλαπλών errors (name_errors, circular imports etc.) κατά το testing, η τελική υλοποίηση συμπεριλήφθηκε στα ήδη υπάρχοντα αρχεία.

LEFT OUTER JOIN

```
# left outer join returns the matching rows as well as the rows which are in the left table but not in the right table
    def _left_outer_join(self, table_right: Table, condition):
        '''
        Join table (left) with a supplied table (right) where condition is met.
        '''
        # get columns and operator
        column_name_left, operator, column_name_right = self._parse_condition(condition, join=True)
        # try to find both columns, if you fail raise error
        try:
            column_index_left = self.column_names.index(column_name_left)
            column_index_right = table_right.column_names.index(column_name_right)
        except:
            raise Exception(f'Columns dont exist in one or both tables.')

        # get the column names of both tables with the table name in front
        # ex. for left -> name becomes left_table_name_name etc
        left_names = [f'{self._name}_{colname}' for colname in self.column_names]
        right_names = [f'{table_right._name}_{colname}' for colname in table_right.column_names]

        # define the new tables name, its column names and types
        join_table_name = f'{self._name}__left_outer_join_{table_right._name}'
        join_table_colnames = left_names+right_names
        join_table_coltypes = self.column_types+table_right.column_types
        join_table = Table(name=join_table_name, column_names=join_table_colnames, column_types= join_table_coltypes)

        # in left_outer_join we need the matching rows of the two tables, as well as the values of the left_table that
        # don't exist in the right_table
        # the values that do not exist in the right_table are replaced with null values
        # exists is a boolean variable which is True if the value of the left_table exists in the right_table too
        # count the number of operations (<,> etc)
        no_of_ops = 0
        null_values = []
        for row_left in self.data:
            null_values.clear()
            left_value = row_left[column_index_left]
            exists = False
            for row_right in table_right.data:
                right_value = row_right[column_index_right]
                no_of_ops+=1
                if get_op(operator, left_value, right_value): #EQ_OP
                    exists = True
            if exists = True:
                join_table._insert(row_left + row_right)
            elif exists = False:
                for i in range(table_right._no_of_columns):
                    null_values.append(0)
                join_table._insert(row_left + null_values)

        print(f'## Select ops no. -> {no_of_ops}')
        print(f'# Left table size -> {len(self.data)}')
        print(f'# Right table size -> {len(table_right.data)}')

        return join_table
```

```
def left_outer_join(self, left_table_name, right_table_name, condition, save_as=None, return_object=False):
    '''
    Join two tables that are part of the database where condition is met.
    left_table_name -> left table's name (needs to exist in database)
    right_table_name -> right table's name (needs to exist in database)
    condition -> a condition using the following format :
                 'column[<,<=,==,>=,>]value' or
                 'value[<,<=,==,>=,>]column'.

                 operatores supported -> (<,<=,==,>=,>)
    save_as -> The name that will be used to save the resulting table in the database. Def: None (no save)
    return_object -> If true, the result will be a table object (usefull for internal usage). Def: False (the result will be printed)
    '''
    self.load(self.savedir)
    if self.is_locked(left_table_name) or self.is_locked(right_table_name):
        print(f'Table/Tables are currently locked')
        return

    res = self.tables[left_table_name]._left_outer_join(self.tables[right_table_name], condition)
    if save_as is not None:
        res._name = save_as
        self.table_from_object(res)
    else:
        if return_object:
            return res
        else:
            res.show()
```

Παραπάνω, παρατείθεται ο κώδικας για την υλοποίηση του left_outer_join, στα αρχεία table.py και database.py, αντίστοιχα. Ο κώδικας στο αρχείο table.py είναι βασισμένος μέχρι ένα σημείο στην ήδη υπάρχουσα υλοποίηση του inner join με βάση τη συνάρτηση _inner_join(), ενώ η left_outer_join() είναι wrapper function της _left_outer_join() και είναι επίσης βασισμένη στην inner_join().

ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ _left_outer_join():

Τα αρχικά στάδια για την εύρεση των columns και του operator, είναι ίδια με εκείνα της ήδη υπάρχουσας συνάρτησης _inner_join(). Έπειτα, ανατρέχουμε τα στοιχεία του αριστερού πίνακα και για κάθε στοιχείο του ψάχνουμε τον δεξί πίνακα. Σε περίπτωση που ικανοποιείται η συνάρτηση get_op, με ορίσματα τα values που δόθηκαν, η μεταβλητή exists γίνεται True, ενώ αν δεν ικανοποιηθεί παραμένει στην αρχική της τιμή False. Στη συνέχεια, με ένα if condition, κάνουμε έλεγχο σχετικά με την Boolean μεταβλητή exists. Εάν η exists = True, τότε προστίθονται στον πίνακα join_table οι τιμές που βρέθηκαν από τον αριστερό και τον δεξί πίνακα, ενώ εάν exists = False, τότε κάνουμε append στη λίστα null_values[] όσα μηδενικά, όσα και το πλήθος των στηλών του δεξιού πίνακα και τελικά προσθέτουμε στον πίνακα join_table, τα στοιχεία του αριστερού πίνακα και

τη λίστα null_values. Τα μηδενικά αναπαριστούν null τιμές, που κανονικά θα έπρεπε να προστίθονται στη λίστα null_values[] με την τιμή None, ωστόσο με αυτόν τον τρόπο προέκυπταν errors για τα οποία δεν καταφέραμε να βρούμε λύση.

RIGHT OUTER JOIN

```
def _right_outer_join(self, table_right: Table, condition):
    '''
    Join table (left) with a supplied table (right) where condition is met.
    '''
    # get columns and operator
    column_name_left, operator, column_name_right = self._parse_condition(condition, join=True)
    # try to find both columns, if you fail raise error
    try:
        column_index_left = self.column_names.index(column_name_left)
        column_index_right = table_right.column_names.index(column_name_right)
    except:
        raise Exception(f'Columns dont exist in one or both tables.')

    # get the column names of both tables with the table name in front
    # ex. for left -> name becomes left_table_name_name etc
     left_names = [f'{self._name}_{colname}' for colname in self.column_names]
    right_names = [f'{table_right._name}_{colname}' for colname in table_right.column_names]

    # define the new tables name, its column names and types
    join_table_name = f'{self._name}_right_outer_join_{table_right._name}'
    join_table_colnames = left_names+right_names
    join_table_coltypes = self.column_types+table_right.column_types
    join_table = Table(name=join_table_name, column_names=join_table_colnames, column_types= join_table_coltypes)

    # in right_outer_join we need the matching rows of the two tables, as well as the values of the right_table that
    # don't exist in the left_table
    # the values that do not exist in the left_table are replaced with null values
    # exists is a boolean variable which is True if the value of the right_table exists in the left_table too
    # count the number of operations (<,> etc)
    no_of_ops = 0
    null_values = []
    for row_left in self.data:
        null_values.clear()
        left_value = row_left[column_index_left]
        exists = False
        for row_right in table_right.data:
            right_value = row_right[column_index_right]
            no_of_ops+=1
            if get_op(operator, left_value, right_value): #EQ_OP
                exists = True
        if exists = True:
            join_table._insert(row_left + row_right)
        elif exists = False:
            for i in range(self._no_of_columns):
                null_values.append(0)
            join_table._insert(row_right + null_values)

    print(f'## Select ops no. -> {no_of_ops}')
    print(f'# Left table size -> {len(self.data)}')
    print(f'# Right table size -> {len(table_right.data)}')

    return join_table
```

```
def right_outer_join(self, left_table_name, right_table_name, condition, save_as=None, return_object=False):
    '''
    Join two tables that are part of the database where condition is met.
    left_table_name -> left table's name (needs to exist in database)
    right_table_name -> right table's name (needs to exist in database)
    condition -> a condition using the following format :
                 'column[<,<=,==,>=,>]value' or
                 'value[<,<=,==,>=,>]column'.

                 operatores supported -> (<,<=,==,>=,>)
    save_as -> The name that will be used to save the resulting table in the database. Def: None (no save)
    return_object -> If true, the result will be a table object (usefull for internal usage). Def: False (the result will be printed)
    '''
    self.load(self.savedir)
    if self.is_locked(left_table_name) or self.is_locked(right_table_name):
        print(f'Table/Tables are currently locked')
        return

    res = self.tables[left_table_name]._right_outer_join(self.tables[right_table_name], condition)
    if save_as is not None:
        res._name = save_as
        self.table_from_object(res)
    else:
        if return_object:
            return res
        else:
            res.show()
```

Παραπάνω παρατίθεται ο κώδικας για το right outer join στα αρχεία table.py και database.py, αντίστοιχα. Η υλοποίηση είναι πανομοιότυπη με εκείνη για το left outer join με τη μόνη διαφορά να είναι η προσθήκη των στοιχείων στον πίνακα στην περίπτωση που η μεταβλητή exists = False, οπού σε αυτή την περίπτωση προστίθενται στον πίνακα null values και τα στοιχεία του δεξιού πίνακα, σε αντίθεση με το left outer join, όπου προστίθονταν null values και τα στοιχεία του αριστερού πίνακα.

FULL OUTER JOIN

Η υλοποίηση του full outer join, δεν χρειάζεται επεξήγηση καθώς στην ουσία είναι η λειτουργία των συναρτήσεων _left_outer_join(), _right_outer_join(), η μια μετά την άλλη. Παρακάτω παρατίθεται ενδεικτικά ο κώδικας του full outer join, μαζί με την wrapper function που βρίσκεται στο αρχείο database.py.

```
def _full_outer_join(self, table_right: Table, condition):
    '''
    Join table (left) with a supplied table (right) where condition is met.
    '''
    # get columns and operator
    column_name_left, operator, column_name_right = self._parse_condition(condition, join=True)
    # try to find both columns, if you fail raise error
    try:
        column_index_left = self.column_names.index(column_name_left)
        column_index_right = table_right.column_names.index(column_name_right)
    except:
        raise Exception(f'Columns dont exist in one or both tables.')

    # get the column names of both tables with the table name in front
    # ex. for left -> name becomes left_table_name_name etc
    left_names = [f'{self._name}_{colname}' for colname in self.column_names]
    right_names = [f'{table_right._name}_{colname}' for colname in table_right.column_names]

    # define the new tables name, its column names and types
    join_table_name = f'{self._name}__full_outer_join_{table_right._name}'
    join_table_colnames = left_names+right_names
    join_table_coltypes = self.column_types+table_right.column_types
    join_table = Table(name=join_table_name, column_names=join_table_colnames, column_types= join_table_coltypes)

    # in order to perform outer_join we actually have to perform left_outer_join and right_outer_join
    # count the number of operations (<,> etc)
    no_of_ops = 0
    for row_left in self.data:
        null_values.clear()
        left_value = row_left[column_index_left]
        exists = False
        for row_right in table_right.data:
            right_value = row_right[column_index_right]
            no_of_ops+=1
            if get_op(operator, left_value, right_value): #EQ_OP
                exists = True
        # if records exist in both tables
        if exists = True:
            join_table._insert(row_left+row_right)
        elif exists = False:
            null_values = []
            for i in range(table_right._no_of_columns):
                null_values.append(0)
            join_table._insert(row_left + null_values)

    for row_left_new in self.data:
        null_values.clear()
        left_value_new = row_left_new[column_index_left]
        exists = False
        for row_right_new in table_right.data:
            right_value_new = row_right_new[column_index_right]
            no_of_ops+=1
            if get_op(operator, left_value_new, right_value_new): #EQ_OP
                exists = True
        # if records exist in both tables
        if exists = True:
            join_table._insert(row_left_new + row_right_new)
        elif exists = False:
            null_values = []
            for i in range(self._no_of_columns):
                null_values.append(0)
            join_table._insert(row_right_new + null_values)

    print(f'## Select ops no. -> {no_of_ops}')
    print(f'# Left table size -> {len(self.data)}')
    print(f'# Right table size -> {len(table_right.data)}')

    return join_table
```

```python
def full_outer_join(self, left_table_name, right_table_name, condition, save_as=None, return_object=False):
    '''
    Join two tables that are part of the database where condition is met.
    left_table_name -> left table's name (needs to exist in database)
    right_table_name -> right table's name (needs to exist in database)
    condition -> a condition using the following format :
                 'column[<,<=,==,>=,>]value' or
                 'value[<,<=,==,>=,>]column'.

                 operatores supported -> (<,<=,==,>=,>)
    save_as -> The name that will be used to save the resulting table in the database. Def: None (no save)
    return_object -> If true, the result will be a table object (usefull for internal usage). Def: False (the result will be printed)
    '''
    self.load(self.savedir)
    if self.is_locked(left_table_name) or self.is_locked(right_table_name):
        print(f'Table/Tables are currently locked')
        return

    res = self.tables[left_table_name]._full_outer_join(self.tables[right_table_name], condition)
    if save_as is not None:
        res._name = save_as
        self.table_from_object(res)
    else:
        if return_object:
            return res
        else:
            res.show()
```

SORT MERGE JOIN

Για την υλοποίηση του sort merge join έχουν δημιουργηθεί δύο συναρτήσεις, η _sort_merge_join() στο αρχείο table.py και η sort_merge_join() στο αρχείο database.py, η οποία είναι wrapper function της _sort_merge_join(). Η υλοποίηση του sort_merge_join() στο database.py είναι πανομοιότυπη με τις υπόλοιπες συναρτήσεις που αναλύσαμε παραπάνω, ενώ η επεξήγηση της κύριας συνάρτησης ακολουθεί παρακάτω:

Αρχικά, ο αλγόριθμος που έχει ακολουθεί είναι ο παρακάτω:

- Sort the left table on the column given in the condition.
- Sort the right table on the column given in the condition.
- “Merge-like” concurrently scan the two sorted tables.

Αρχικά, ταξινομούμε τους δύο πίνακες βασιζόμενοι στην υλοποίηση της ήδη υπάρχουσας συνάρτησης _sort, ενώ στη συνέχεια ακολουθεί η ίδια διαδικασία για την εύρεση των columns και την ονοματοδοσία των σχέσεων, όπως και στις προηγούμενες συναρτήσεις.

Στη συνέχεια, αρχικοποιούμε δύο μεταβλητές inner_counter, outer_counter, τις οποίες χρησιμοποιούμε ως indexes για να σκανάρουμε τους δύο πίνακες. Ξεκινώντας τη διαδικασία του merge ορίζουμε ένα condition μέσα στο οποίο θα γίνονται οι συγκρίσεις για όσο οι δείκτες δεν έχουν φτάσει στο τέλος της κάθε λίστας. Αν τα στοιχεία που συγκρίνουμε είναι ίδια, τότε τα προσθέτουμε στη λίστα join_table και αυξάνουμε τον outer_counter κατά μια μονάδα συνεχίζοντας το σκανάρισμα στον δεξί πίνακα αλλά κρατώντας το ίδιο στοιχείο του αριστερού πίνακα. Εν συνεχεία ακολουθούν οι δύο ακόμη περιπτώσεις σχετικά με τα στοιχεία που συγκρίνονται (το ένα να είναι μικρότερο ή μεγαλύτερο από το άλλο), αυξάνοντας ή μειώνοντας τους απαραίτητους δείκτες κάθε φορά, ώστε να συνεχίσει το σκανάρισμα του κάθε πίνακα με τον σωστό τρόπο. Παρακάτω παρατίθεται ο κώδικας του sort merge join στα αρχεία database.py και table.py αντίστοιχα.

```
def sm_join(self, left_table_name, right_table_name, condition, save_as=None, return_object=False):
    '''
    Join two tables that are part of the database where condition is met.
    left_table_name -> left table's name (needs to exist in database)
    right_table_name -> right table's name (needs to exist in database)
    condition -> a condition using the following format :
                 'column[<,<=,==,>=,>]value' or
                 'value[<,<=,==,>=,>]column'.

                 operatores supported -> (<,<=,==,>=,>)
    save_as -> The name that will be used to save the resulting table in the database. Def: None (no save)
    return_object -> If true, the result will be a table object (usefull for internal usage). Def: False (the result will be printed)
    '''
    self.load(self.savedir)
    if self.is_locked(left_table_name) or self.is_locked(right_table_name):
        print(f'Table/Tables are currently locked')
        return

    res = self.tables[left_table_name]._sm_join(self.tables[right_table_name], condition)
    if save_as is not None:
        res._name = save_as
        self.table_from_object(res)
    else:
        if return_object:
            return res
        else:
            res.show()
```

```python
# 1) sort both tables on the condition given
# 2) merge-like concurrently scan the two tables
# 3) return the join_table containing the matching rows, found through merging the tables
def _sm_join(self, table_right: Table, condition):
    '''
    Join table (left) with a supplied table (right) where condition is met.
    '''
    # get columns and operator
    column_name_left, operator, column_name_right = self._parse_condition(condition, join=True)

    # sort table left and table right using the implementation of _sort def that already exists
    # sort left table
    column = self.columns[self.column_names.index(column_name_left)]
    idx = sorted(range(len(column)), key=lambda k: column[k])
    self.data = [self.data[i] for i in idx]
    self._update()
    # sort right table
    column = self.columns[self.column_names.index(column_name_right)]
    idx = sorted(range(len(column)), key=lambda k: column[k])
    table_right.data = [table_right.data[i] for i in idx]
    table_right._update()

    # try to find both columns, if you fail raise error
    try:
        column_index_left = self.column_names.index(column_name_left)
        column_index_right = table_right.column_names.index(column_name_right)
    except:
        raise Exception(f'Columns dont exist in one or both tables.')

    # get the column names of both tables with the table name in front
    # ex. for left -> name becomes left_table_name_name etc
    left_names = [f'{self._name}_{colname}' for colname in self.column_names]
    right_names = [f'{table_right._name}_{colname}' for colname in table_right.column_names]

    # define the new tables name, its column names and types
    join_table_name = f'{self._name}_sm_join_{table_right._name}'
    join_table_colnames = left_names+right_names
    join_table_coltypes = self.column_types+table_right.column_types
    join_table = Table(name=join_table_name, column_names=join_table_colnames, column_types= join_table_coltypes)

    # merge the two tables now that they are sorted
    # initialize indexes to keep track of the table scanning while merging
    # inner_counter is the index for self table
    inner_counter = 0
    # outer_counter is the index for table_right
    outer_counter = 0

    # while loop checking whether the indexes have reached the end of the tables
    while inner_counter <= len(self.columns)-1 and outer_counter <= len(table_right.columns)-1:
        # if the two values are the same, then insert in the join_table and then increase the outer_counter by 1
        if self.data[inner_counter][column_index_left] == table_right.data[outer_counter][column_index_right]:
            join_table._insert(self.data[inner_counter] + table_right.data[outer_counter])
            outer_counter+=1
        # else if self table's value is larger than right table's value, increase the outer counter by 1 again, but this time don't insert the values in the join_table
        elif self.data[inner_counter][column_index_left] > table_right.data[outer_counter][column_index_right]:
            outer_counter+=1
        # else if self table's value is smaller than right table's value, increase the inner counter by 1 and again don't insert the values in the join_table
        elif self.data[inner_counter][column_index_left] < table_right.data[outer_counter][column_index_right]:
            inner_counter+=1
            # if table_right's values are larger than self_table's data then decrease the outer_counter by 1 so that the values can be appropriately merged
            while outer_counter >= 1:
                if self.data[inner_counter][column_index_left] <= table_right.data[outer_counter][column_index_right]:
                    outer_counter-=1
                else:
                    break

    # tried to merge the two tables using pandas library but it's not working for the moment
    # now that the tables are sorted we can merge the two tables using the pandas library
    # merged_table = pd.merge(self, table_right, left_on="column_name_left", right_on="column_name_right")
    # for row_left in merged_table.data:
        # for row_right in merged_table.data:
            # no_of_ops += 1
            # join_table._insert(row_left + row_right)

    # find where to place no_of_ops sos
    # print(f'## Select ops no. -> {no_of_ops}')
    print(f'# Left table size -> {len(self.data)}')
    print(f'# Right table size -> {len(table_right.data)}')
    return join_table
```

INDEX NESTED LOOP JOIN

Παρακάτω παρατίθεται ο κώδικας για την υλοποίηση του index nested loop join, στα αρχεία table.py και database.py αντίστοιχα. Για τη συγκεκριμένη συνάρτηση έγινε θεωρητική υλοποίηση του αλγορίθμου, ωστόσο δεν καταφέραμε να την ολοκληρώσουμε πρακτικά ώστε να έχουμε το επιθυμητό αποτέλεσμα, λόγω errors των οποίων η λύση δεν βρέθηκε. Θα εκτιμούσαμε να ληφθεί υπόψιν ως ένα σημείο η θεωρητική ανάπτυξη αυτή του αλγορίθμου και ακόμη αργότερα να λαμβάναμε feedback σχετικά με το πώς θα έπρεπε να υλοποιηθεί η συνάρτηση index nested loop join, ίσως μέσω εργασιών άλλων ομάδων που θα γίνουν merge στην κύρια εφαρμογή.

```
def inl_join(self, left_table_name, right_table_name, condition, save_as=None, return_object=False):
    '''
    Join two tables that are part of the database where condition is met.
    left_table_name -> left table's name (needs to exist in database)
    right_table_name -> right table's name (needs to exist in database)
    condition -> a condition using the following format :
                 'column[<,<=,==,>=,>]value' or
                 'value[<,<=,==,>=,>]column'.

                 operatores supported -> (<,<=,==,>=,>)
    save_as -> The name that will be used to save the resulting table in the database. Def: None (no save)
    return_object -> If true, the result will be a table object (usefull for internal usage). Def: False (the result will be printed)
    '''
    self.load(self.savedir)
    if self.is_locked(left_table_name) or self.is_locked(right_table_name):
        print(f'Table/Tables are currently locked')
        return

    res = self.tables[left_table_name]._inl_join(self.tables[right_table_name], condition)
    if save_as is not None:
        res._name = save_as
        self.table_from_object(res)
    else:
        if return_object:
            return res
        else:
            res.show()
```

```python
def _inl_join(self, table_right: Table, condition):
    from database import Database
    # create an instance of the Database class
    database = Database()
    '''
    Join table (left) with a supplied table (right) where condition is met.
    '''
    # get columns and operator
    column_name_left, operator, column_name_right = self._parse_condition(condition, join=True)
    # try to find both columns, if you fail raise error
    try:
        column_index_left = self.column_names.index(column_name_left)
        column_index_right = table_right.column_names.index(column_name_right)
    except:
        raise Exception(f'Columns dont exist in one or both tables.')

    # get the column names of both tables with the table name in front
    # ex. for left -> name becomes left_table_name_name etc
    left_names = [f'{self._name}_{colname}' for colname in self.column_names]
    right_names = [f'{table_right._name}_{colname}' for colname in table_right.column_names]

    # define the new tables name, its column names and types
    join_table_name = f'{self._name}__inl_join_{table_right._name}'
    join_table_colnames = left_names+right_names
    join_table_coltypes = self.column_types+table_right.column_types
    join_table = Table(name=join_table_name, column_names=join_table_colnames, column_types= join_table_coltypes)

    '''
    INDEX NESTED LOOP JOIN ALGORITHM
    '''
    # creating an index for the inner table
    index_name = f'{table_right._name}_index'
    # create an index using the _create_index function
    database._create_index(self, table_right._name, index_name, index_type = 'Btree')

    # trying to create an index based on the implementation of _create_index function that already exists
    if Database.tables[table_right._name].pk._idx is None:
        print("Can't create index because table has no primary key!")
        return
    if index_name not in Database.tables['meta_indexes'].index_name:
        print("Creating Btree index")
        Database.tables['meta_indexes']._insert([table_right._name, index_name])
        Database._construct_index(table_right._name, index_name)
        Database.save()
    else:
        print("Index already exists, can't create a new one!")
        return

    # now for each value of the left table, we need to search in the index we created and check if it exists
    # if the value we are searching exists, it should be appended to the join_table
    # create an empty list to store in it the values that match
    results = []

    for row_left in self.data:
        left_value = row_left[column_index_left]
    for left_value in self.data:
        no_of_ops += 1
        leaf_idx, ops = self._search(left_value, True)
        target_node = self.nodes[leaf_idx]
        # if the element we are searching for exists in the Btree, append in the list results[], else pass and return
        try:
            results.append(target_node.ptrs[target_node.values.index(left_value)])
            print("The value was found in the Btree")
            join_table._insert(results)
        except:
            print("Not found")
            pass

    print(f'# Left table size -> {len(self.data)}')
    print(f'# Right table size -> {len(table_right.data)}')


    return join table
```

ΠΑΡΑΔΕΙΓΜΑΤΑ ΛΕΙΤΟΥΓΡΙΑΣ ΣΥΝΑΡΤΗΣΕΩΝ ΠΟΥ ΥΛΟΠΟΙΗΘΗΚΑΝ

LEFT OUTER JOIN

```
>>> db.left_outer_join('instructor', 'teaches', 'ID==ID')
## Select ops no. -> 180
# Left table size -> 12
# Right table size -> 15

## instructor__left_outer_join_teaches ##
  instructor_ID (str)  instructor_name (str)    instructor_dept_name (str)      instructor_salary (int)    teaches_ID (str)  teaches_course_id (str)      teaches_sec_id (str)  teaches_semester (str)
      teaches_year (int)
---------------------  ----------------------  --------------------------  ------------------------  ------------------  -----------------------  ----------------------  ----------------------
--  --------------------
               10101  Srinivasan              Comp. Sci.                                     65000               98345  EE-181                                         1  Spring
                   2009
               12121  Wu                      Finance                                        90000               98345  EE-181                                         1  Spring
                   2009
               15151  Mozart                  Music                                          40000               98345  EE-181                                         1  Spring
                   2009
               22222  Einstein                Physics                                        95000               98345  EE-181                                         1  Spring
                   2009
               32343  El Said                 History                                        60000               98345  EE-181                                         1  Spring
                   2009
               33456  Gold                    Physics                                        87000                   0  0                                              0  0
                   0
               45565  Katz                    Comp. Sci.                                     75000               98345  EE-181                                         1  Spring
                   2009
               58583  Califieri               History                                        62000                   0  0                                              0  0
                   0
               76543  Singh                   Finance                                        80000                   0  0                                              0  0
                   0
               76766  Crick                   Biology                                        72000               98345  EE-181                                         1  Spring
                   2009
               83821  Brandt                  Comp. Sci.                                     92000               98345  EE-181                                         1  Spring
                   2009
               98345  Kim                     Elec. Eng.                                     80000               98345  EE-181                                         1  Spring
                   2009
```

RIGHT OUTER JOIN

```
>>> db.right_outer_join('instructor', 'teaches', 'ID==ID')
## Select ops no. -> 180
# Left table size -> 12
# Right table size -> 15

## instructor_right_outer_join_teaches ##
  instructor_ID (str)  instructor_name (str)    instructor_dept_name (str)      instructor_salary (int)    teaches_ID (str)  teaches_course_id (str)      teaches_sec_id (str)  teaches_semester (str)
      teaches_year (int)
---------------------  ----------------------  --------------------------  ------------------------  ------------------  -----------------------  ----------------------  ----------------------
--  --------------------
               10101  Srinivasan              Comp. Sci.                                     65000               98345  EE-181                                         1  Spring
                   2009
               12121  Wu                      Finance                                        90000               98345  EE-181                                         1  Spring
                   2009
               15151  Mozart                  Music                                          40000               98345  EE-181                                         1  Spring
                   2009
               22222  Einstein                Physics                                        95000               98345  EE-181                                         1  Spring
                   2009
               32343  El Said                 History                                        60000               98345  EE-181                                         1  Spring
                   2009
                   0  0                       0                                                  0               98345  EE-181                                         1  Spring
                   2009
               45565  Katz                    Comp. Sci.                                     75000               98345  EE-181                                         1  Spring
                   2009
                   0  0                       0                                                  0               98345  EE-181                                         1  Spring
                   2009
                   0  0                       0                                                  0               98345  EE-181                                         1  Spring
                   2009
               76766  Crick                   Biology                                        72000               98345  EE-181                                         1  Spring
                   2009
               83821  Brandt                  Comp. Sci.                                     92000               98345  EE-181                                         1  Spring
                   2009
               98345  Kim                     Elec. Eng.                                     80000               98345  EE-181                                         1  Spring
                   2009
```

FULL OUTER JOIN

```
>>> db.full_outer_join('instructor', 'teaches', 'ID==ID')
## Select ops no. -> 360
# Left table size -> 12
# Right table size -> 15

## instructor__full_outer_join_teaches ##
  instructor_ID (str)  instructor_name (str)    instructor_dept_name (str)      instructor_salary (int)    teaches_ID (str)  teaches_course_id (str)      teaches_sec_id (str)  teaches_semester (str)
      teaches_year (int)
---------------------  ----------------------  --------------------------  ------------------------  ----------------  -----------------------  --------------------  ----------------------
--  --------------------
               10101  Srinivasan              Comp. Sci.                                    65000             98345  EE-181                                       1  Spring
                   2009
               12121  Wu                      Finance                                       90000             98345  EE-181                                       1  Spring
                   2009
               15151  Mozart                  Music                                         40000             98345  EE-181                                       1  Spring
                   2009
               22222  Einstein                Physics                                       95000             98345  EE-181                                       1  Spring
                   2009
               32343  El Said                 History                                       60000             98345  EE-181                                       1  Spring
                   2009
               33456  Gold                    Physics                                       87000                 0  0                                            0  0
                      0
               45565  Katz                    Comp. Sci.                                    75000             98345  EE-181                                       1  Spring
                   2009
               58583  Califieri               History                                       62000                 0  0                                            0  0
                      0
               76543  Singh                   Finance                                       80000                 0  0                                            0  0
                      0
               76766  Crick                   Biology                                       72000             98345  EE-181                                       1  Spring
                   2009
               83821  Brandt                  Comp. Sci.                                    92000             98345  EE-181                                       1  Spring
                   2009
               98345  Kim                     Elec. Eng.                                    80000             98345  EE-181                                       1  Spring
                   2009
               10101  Srinivasan              Comp. Sci.                                    65000             98345  EE-181                                       1  Spring
                   2009
               12121  Wu                      Finance                                       90000             98345  EE-181                                       1  Spring
                   2009
               15151  Mozart                  Music                                         40000             98345  EE-181                                       1  Spring
                   2009
               22222  Einstein                Physics                                       95000             98345  EE-181                                       1  Spring
                   2009
               32343  El Said                 History                                       60000             98345  EE-181                                       1  Spring
                   2009
                   0  0                       0                                                 0             98345  EE-181                                       1  Spring
                   2009
               45565  Katz                    Comp. Sci.                                    75000             98345  EE-181                                       1  Spring
                   2009
                   0  0                       0                                                 0             98345  EE-181                                       1  Spring
                   2009
                   0  0                       0                                                 0             98345  EE-181                                       1  Spring
                   2009
               76766  Crick                   Biology                                       72000             98345  EE-181                                       1  Spring
                   2009
               83821  Brandt                  Comp. Sci.                                    92000             98345  EE-181                                       1  Spring
                   2009
               98345  Kim                     Elec. Eng.                                    80000             98345  EE-181                                       1  Spring
                   2009
```

SORT MERGE JOIN

```
>>> db.sm_join('instructor', 'teaches', 'ID==ID')
# Left table size -> 12
# Right table size -> 12

## instructor_sm_join_teaches ##
  instructor_ID (str)  instructor_name (str)    instructor_dept_name (str)      instructor_salary (int)    teaches_ID (str)  teaches_course_id (str)      teaches_sec_id (str)  teaches_semester (str)
      teaches_year (int)
---------------------  ----------------------  --------------------------  ------------------------  ----------------  -----------------------  --------------------  ----------------------
--  --------------------
               10101  Srinivasan              Comp. Sci.                                    65000             10101  CS-101                                       1  Fall
                   2009
               10101  Srinivasan              Comp. Sci.                                    65000             10101  CS-315                                       1  Spring
                   2010
               10101  Srinivasan              Comp. Sci.                                    65000             10101  CS-347                                       1  Fall
                   2009
               12121  Wu                      Finance                                       90000             12121  FIN-201                                      1  Spring
                   2010
               15151  Mozart                  Music                                         40000             15151  MU-199                                       1  Spring
                   2010
```

Λόγω προβλήματος με την ανάλυση της οθόνης χρησιμοποιώντας το virtual box, το αποτέλεσμα της σύζευξης των πινάκων είναι ‘’παραμορφωμένο’’ καθώς η τελευταία στήλη εμφανίζεται κάτω από την 1^{η}.