Swapnali Vijay Kadam – 202091929

Priyanka Maru - 202091536

COMP 6908 – Database Tech and Applications

Description of the Directory Structure

svkadam_pmaru
- Program
  - bTreeOperations.py
  - buildTree.py
  - display.py
  - query.py
  - relAlg.py
  - remove.py
- Data
  - Products
    - page00.txt
    - page01.txt
    - …….
    - page17.txt
    - pageLink.txt
  - ` Suppliers
    - page00.txt
    - page01.txt
    - …….
    - page12.txt
    - pageLink.txt
  - Supply
    - page00.txt
    - page01.txt
    - …….
    - Page53.txt
    - pageLink.txt
  - pagePool.txt
  - schemas.txt
- Index
  - directory.txt
  - pagePool.txt
  - pg00.txt
  - pg01.txt
  - ……
  - Pgn.txt
- treePic

Suppliers_sid.txt
Supply_pid.txt

queryOutput

queryResult.txt

Files in Program Folder – It contains Five python files

1. **buildTree.py** –

   In this file we are using function build(rel, att, od) where od is an order of B+ tree, att is a search key attribute and rel is relation to create two B+ trees – one on sid of Suppliers relation and another B+ tree on pid of Supply relation. We have written detailed function definitions in bTreeOperations.py and are calling them in buildTree.py file to make it more readable. The overview of the functionality that is implemented in the buildTree file are :

   -Using the pages from pagePool to build the B+tree
   -Reading schemas.txt file to get schema of the table
   -Retrieving attribute list for B+ tree.

   We are inserting node in a B+ tree with the help of node, key and value parameters. We are using write function to get the sibling node i.e. left and right nodes, leaf nodes or Intermediate nodes of b+ tree.

1.2 **bTreeOperations.py**-

   We are using this file to write logic of functions called in buildTree.py. Here we have written functions to assign get pages from pagepool.txt, assign new root, find leaf, or intermediate node, if exceed node capacity then split node of B+ tree. We are importing bTreeOperations.py file in buildTree.py to use functions written in a bTreeOperations.py

2. **display.py-**

   In this file we are displaying structure of the B+ tree with root file fname. We are using display table function with relation and fname parameters to read pages from pagelink and display result of queries in queryResult.txt under queryOutput folder.
   We have also written logic here to display tree pic on sid of Suppliers relation and pid of Supply relation under treePic folder in Suppliers_sid.txt and Supply_pid.txt files respectively.

3. **remove.py-**

   This file contains two main functions removeTable(*) and removeTree(*)
   removeTable(rel) takes in the relation and is used for removing the relation from the database system. In this function , we need check if a path on the particular relation exists if so, we read the pages from pagelink and pagepool and dump the back to the page_pool file and remove a temporary path that was created. Once the relation is removed the pagelinks and pages for that relation will be removed.
   The removeTree(rel,att) function takes in the relation rel and the attribute att on which the tree needs to be removed. Before removal of a node , a search function is

used for getting all the nodes that needs to be removed . Once the tree is removed the pages are also released and dump it back to the page pool.

4. **query.py-**

This file contains 5 relational algebra queries given in problem statement and displays result by calling display function from display.py file

In query A as we need to get the name for the supplier 's23' when a B+_tree exists on *Suppliers.sid,* so we first select all the tuples based on the select condition i.e. sid=23 and then perform projection on the selected relation to get only the names of the supplier.

Similary, in Query B we have removed the tree on suppliers.sid and again checked the same condition as done in Query A. (i.e sid=23).

In query C ,we are first selecting on the given condition pid= p15, then we perform join this relation with the supply relation on the common attribute "sid", finally we project on the resultant data to get the address of the Suppliers.

In Query D , we perform select operation on Suppliers and Supply relation using the condition "sname = Kiddie" and "pid=p20" respectively. Next, we perform the join operation on both the relation and finally project on the resulting relation to get the cost .

Similarly , In query E , we perform select to get the tuples whose cost is >= 47.00 on Supply relation , then join the resulting relation with Products relations on attribute "pid" and again join the previous resulting relation with Suppliers relation on "sid". Finally , we project on the result we get after performing the two join operations to get the supplier's name , product's name and cost , whose cost is higher or equal to 47.

5. **relAlg.py**-

This file contains all the 3 operations - select, project, and join.

1)select(self,rel, att, op, val):

In Select operation we match the condition based on different operators '<','<=','=','>','>=' and given attributes and it returns name of the resulting relation.

2) project(self, rel, attList):

This operation takes in the relation and attribute List i.e the list of string and returns the tuples for a particular column. The schema of the projected relation will consists the list of the given attributes. It returns name of resulting relation.

3) join(self, rel1, att1, rel2, att2):

This operation takes in two relations and two attributes and returns a new joined relation with tuples that satisfy the join condition. The schema of the resulting relation would be the combination of both the relations except one of the attributes i.e. attr1 or attr2 .

Files in Index Folder

1. Directory.txt
   Directory.txt is used to keep track of root page of the relation . After running buildTree.txt, a page is pushed which is referred as the root node to further build the B+tree.

2. pagePool.txt
   pagePool.txt is a collection of available pages to build the B+ tree. After removing the B+ tree, pages are released to the pagePool.txt. It is used to keep track of page names like pg01.txt,pg02.txt..,etc.

3. page1…pagen
   These are the pages that are occupied by B+ tree which are nothing but B+ tree nodes. After removing the B+ tree, pages are removed.

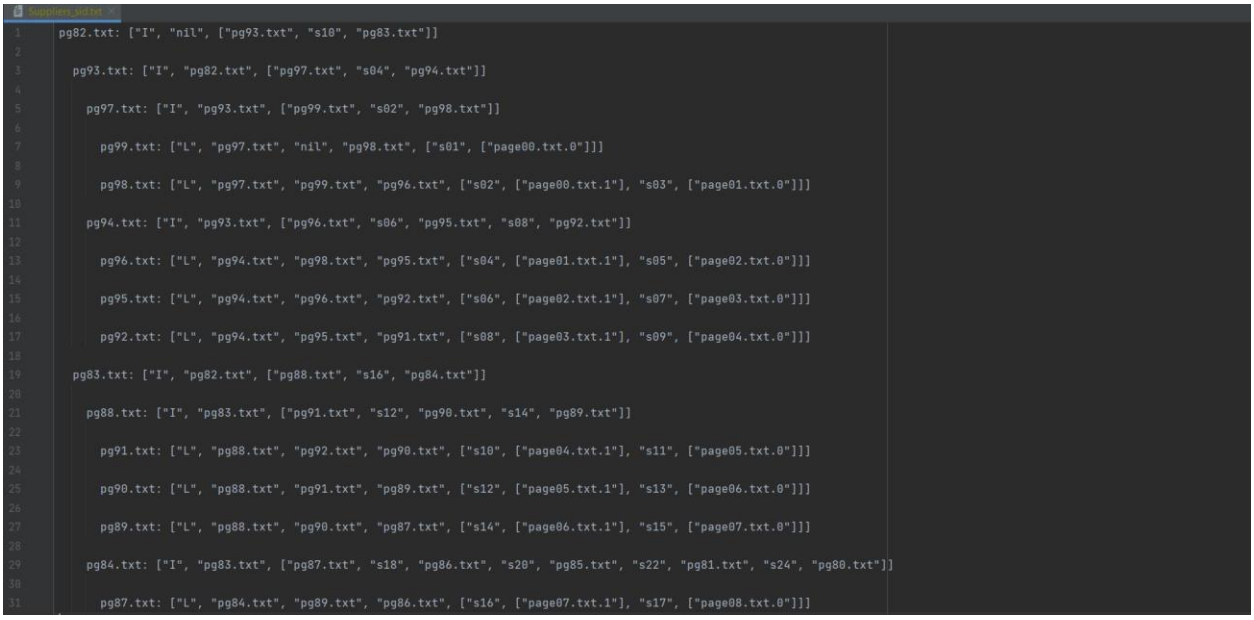
How to Run Project:

- Project requirements : Install python3.8 and Install pandas .
- Run the BuildTree.py file to build the tree on sid and pid.
- Run the query.py file and check the queryResult.txt in the queryOutput folder
- Run the display.py file to get the treePic on Suppliers.sid and Supply.pid
    - Note : The root node in function displayTreePic() of main method of display.py should be same as root node in directory.txt.
      i.e. display().displayTreePic("pg82.txt").
      display().displayTreePic("pg73.txt")

      i.e. [["Suppliers", "sid", "pg82.txt"],
      ["Supply", "pid", "pg73.txt"]]

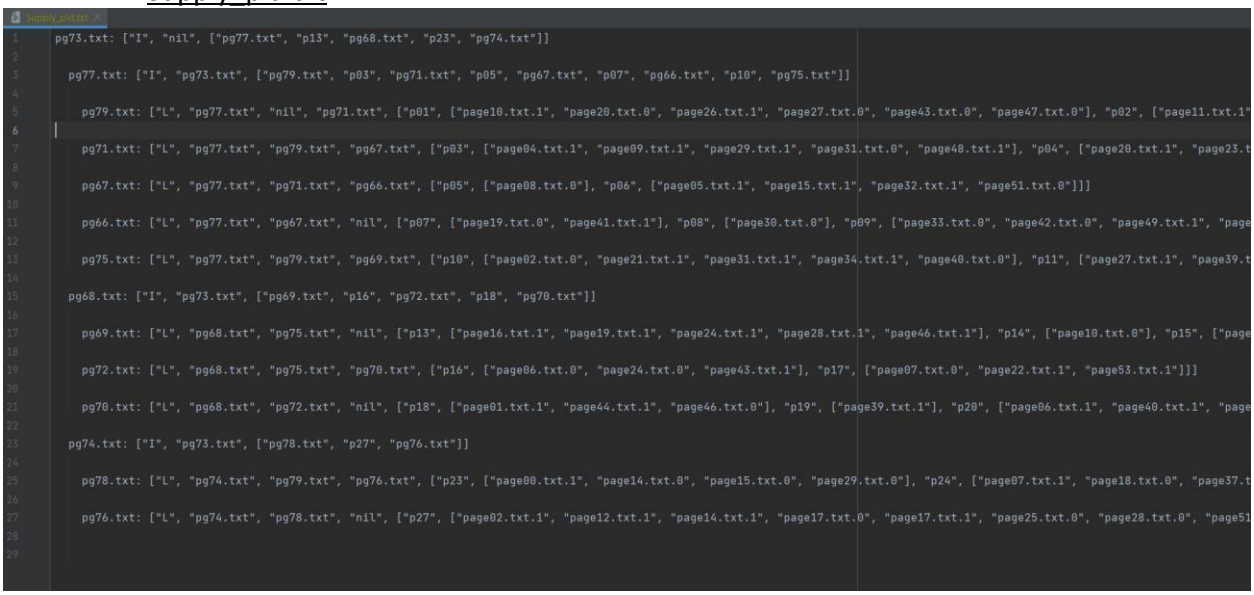
Files in treePic Folder

1. Suppliers_sid.txt



```
pg82.txt: ["I", "nil", ["pg93.txt", "s10", "pg83.txt"]]

  pg93.txt: ["I", "pg82.txt", ["pg97.txt", "s04", "pg94.txt"]]

    pg97.txt: ["I", "pg93.txt", ["pg99.txt", "s02", "pg98.txt"]]

      pg99.txt: ["L", "pg97.txt", "nil", "pg98.txt", ["s01", ["page00.txt.0"]]]

      pg98.txt: ["L", "pg97.txt", "pg99.txt", "pg96.txt", ["s02", ["page00.txt.1"], "s03", ["page01.txt.0"]]]

    pg94.txt: ["I", "pg93.txt", ["pg96.txt", "s06", "pg95.txt", "s08", "pg92.txt"]]

      pg96.txt: ["L", "pg94.txt", "pg98.txt", "pg95.txt", ["s04", ["page01.txt.1"], "s05", ["page02.txt.0"]]]

      pg95.txt: ["L", "pg94.txt", "pg96.txt", "pg92.txt", ["s06", ["page02.txt.1"], "s07", ["page03.txt.0"]]]

      pg92.txt: ["L", "pg94.txt", "pg95.txt", "pg91.txt", ["s08", ["page03.txt.1"], "s09", ["page04.txt.0"]]]

  pg83.txt: ["I", "pg82.txt", ["pg88.txt", "s16", "pg84.txt"]]

    pg88.txt: ["I", "pg83.txt", ["pg91.txt", "s12", "pg90.txt", "s14", "pg89.txt"]]

      pg91.txt: ["L", "pg88.txt", "pg92.txt", "pg90.txt", ["s10", ["page04.txt.1"], "s11", ["page05.txt.0"]]]

      pg90.txt: ["L", "pg88.txt", "pg91.txt", "pg89.txt", ["s12", ["page05.txt.1"], "s13", ["page06.txt.0"]]]

      pg89.txt: ["L", "pg88.txt", "pg90.txt", "pg87.txt", ["s14", ["page06.txt.1"], "s15", ["page07.txt.0"]]]

    pg84.txt: ["I", "pg83.txt", ["pg87.txt", "s18", "pg86.txt", "s20", "pg85.txt", "s22", "pg81.txt", "s24", "pg80.txt"]]

      pg87.txt: ["L", "pg84.txt", "pg89.txt", "pg86.txt", ["s16", ["page07.txt.1"], "s17", ["page08.txt.0"]]]
```

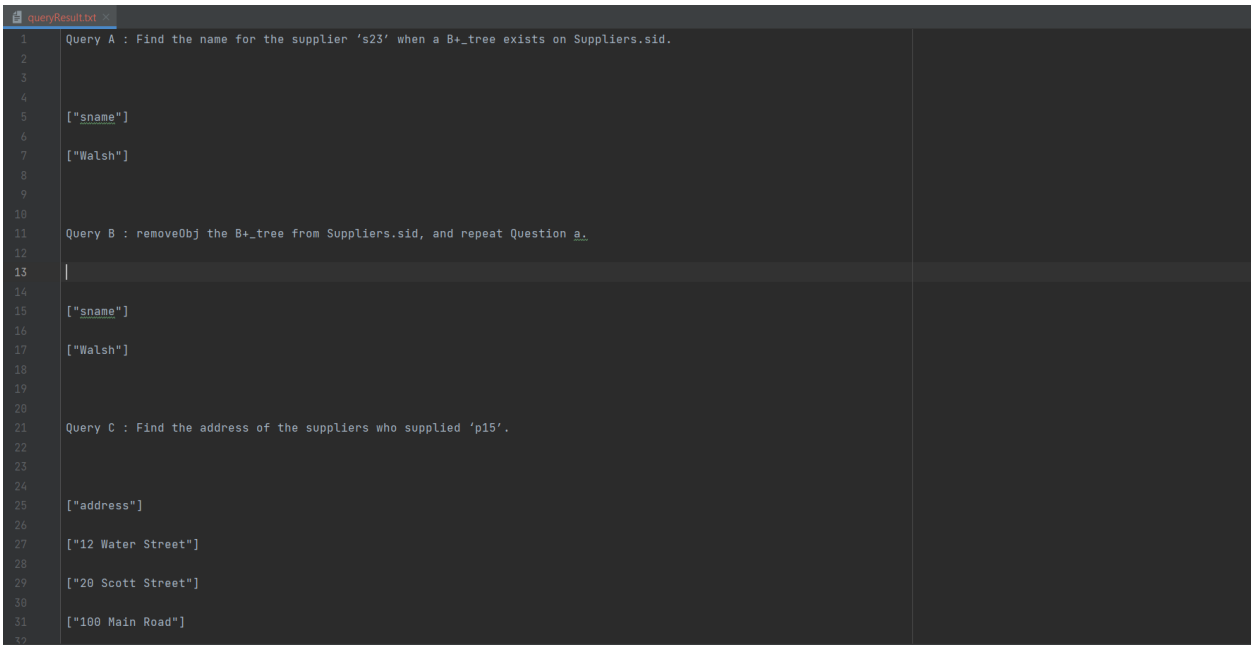
2. Supply_pid.txt



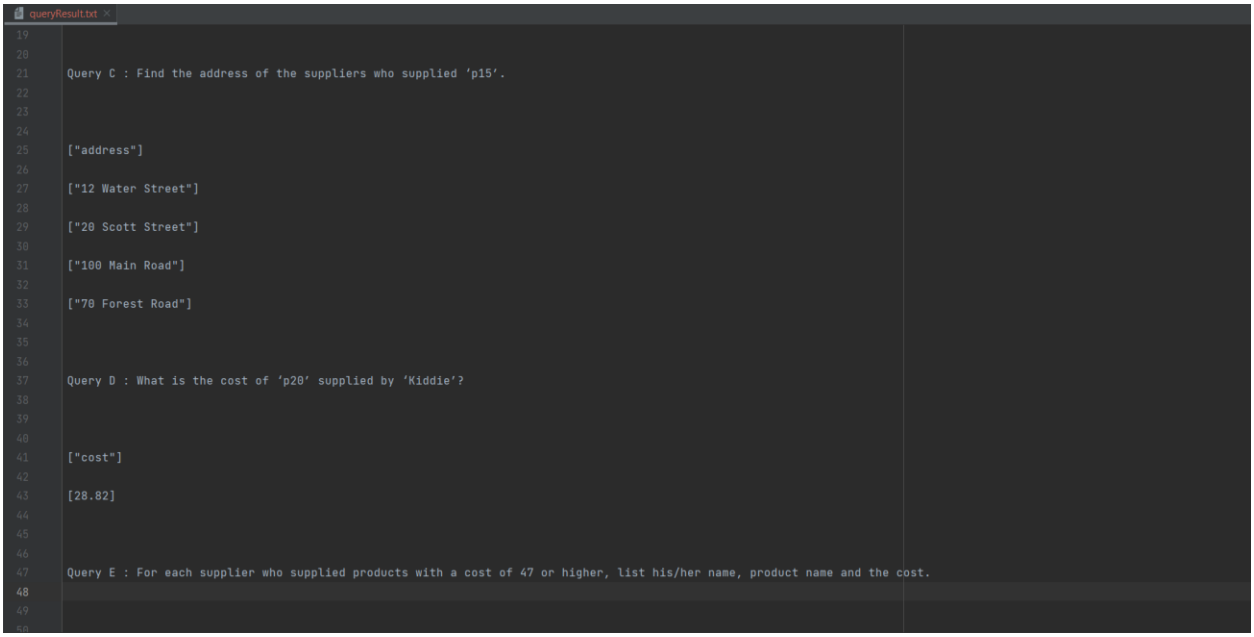
```
pg73.txt: ["I", "nil", ["pg77.txt", "p13", "pg68.txt", "p23", "pg74.txt"]]

  pg77.txt: ["I", "pg73.txt", ["pg79.txt", "p03", "pg71.txt", "p05", "pg67.txt", "p07", "pg66.txt", "p10", "pg75.txt"]]

    pg79.txt: ["L", "pg77.txt", "nil", "pg71.txt", ["p01", ["page10.txt.1", "page20.txt.0", "page26.txt.1", "page27.txt.0", "page43.txt.0", "page47.txt.0"], "p02", ["page11.txt.1"

    pg71.txt: ["L", "pg77.txt", "pg79.txt", "pg67.txt", ["p03", ["page04.txt.1", "page09.txt.1", "page29.txt.1", "page31.txt.0", "page48.txt.1"], "p04", ["page20.txt.1", "page23.t

    pg67.txt: ["L", "pg77.txt", "pg71.txt", "pg66.txt", ["p05", ["page08.txt.0"], "p06", ["page05.txt.1", "page15.txt.1", "page32.txt.1", "page51.txt.0"]]]

    pg66.txt: ["L", "pg77.txt", "pg67.txt", "nil", ["p07", ["page19.txt.0", "page41.txt.1"], "p08", ["page30.txt.0"], "p09", ["page33.txt.0", "page42.txt.0", "page49.txt.1", "page

    pg75.txt: ["L", "pg77.txt", "pg79.txt", "pg69.txt", ["p10", ["page02.txt.0", "page21.txt.1", "page31.txt.1", "page34.txt.1", "page40.txt.0"], "p11", ["page27.txt.1", "page39.t

  pg68.txt: ["I", "pg73.txt", ["pg69.txt", "p16", "pg72.txt", "p18", "pg70.txt"]]

    pg69.txt: ["L", "pg68.txt", "pg75.txt", "nil", ["p13", ["page16.txt.1", "page19.txt.1", "page24.txt.1", "page28.txt.1", "page46.txt.1"], "p14", ["page10.txt.0"], "p15", ["page

    pg72.txt: ["L", "pg68.txt", "pg75.txt", "pg70.txt", ["p16", ["page06.txt.0", "page24.txt.0", "page43.txt.1"], "p17", ["page07.txt.0", "page22.txt.1", "page53.txt.1"]]]

    pg70.txt: ["L", "pg68.txt", "pg72.txt", "nil", ["p18", ["page01.txt.1", "page44.txt.1", "page46.txt.0"], "p19", ["page39.txt.1"], "p20", ["page06.txt.1", "page40.txt.1", "page

  pg74.txt: ["I", "pg73.txt", ["pg78.txt", "p27", "pg76.txt"]]

    pg78.txt: ["L", "pg74.txt", "pg79.txt", "pg76.txt", ["p23", ["page08.txt.1", "page14.txt.0", "page15.txt.0", "page29.txt.0"], "p24", ["page07.txt.1", "page18.txt.0", "page37.t

    pg76.txt: ["L", "pg74.txt", "pg78.txt", "nil", ["p27", ["page02.txt.1", "page12.txt.1", "page14.txt.1", "page17.txt.0", "page17.txt.1", "page25.txt.0", "page28.txt.0", "page51
```

Files in queryOutput Folder

1. queryResult.txt



```
Query A : Find the name for the supplier 's23' when a B+_tree exists on Suppliers.sid.



["sname"]

["Walsh"]



Query B : removeObj the B+_tree from Suppliers.sid, and repeat Question a.

["sname"]

["Walsh"]



Query C : Find the address of the suppliers who supplied 'p15'.



["address"]

["12 Water Street"]

["20 Scott Street"]

["100 Main Road"]
```



```
Query C : Find the address of the suppliers who supplied 'p15'.



["address"]

["12 Water Street"]

["20 Scott Street"]

["100 Main Road"]

["70 Forest Road"]



Query D : What is the cost of 'p20' supplied by 'Kiddie'?



["cost"]

[28.82]



Query E : For each supplier who supplied products with a cost of 47 or higher, list his/her name, product name and the cost.
```

queryResult.txt

```
Query E : For each supplier who supplied products with a cost of 47 or higher, list his/her name, product name and the cost.

["sname", "pname", "cost"]
["Brown", "shovel", 48.78]
["Brown", "grill", 49.85]
["Wang", "siding", 49.39]
["Carew", "paint", 48.57]
["Carew", "air conditioner", 49.55]
["Carter", "vacuum", 47.3]
["Evoy", "matherboard", 49.26]
["Evans", "switch", 47.3]
["Carter", "screwdriver", 47.27]
["Lee", "fan", 47.81]
["kielly", "kettle", 48.01]
["Zhang", "flashlight", 49.45]
["Edward", "matherboard", 49.01]
```

queryResult.txt

```
["Evans", "switch", 47.3]
["Carter", "screwdriver", 47.27]
["Lee", "fan", 47.81]
["kielly", "kettle", 48.01]
["Zhang", "flashlight", 49.45]
["Edward", "matherboard", 49.01]
["Edward", "monitor", 47.62]
["Newwll", "air conditioner", 49.66]
["Hayley", "usb", 47.42]
["Hayley", "flashlight", 49.62]
["Hayley", "switch", 48.76]
["Kiddie", "sander", 47.69]
["Kiddie", "usb", 47.12]
["Kieley", "matherboard", 47.03]
["Kieley", "chair", 48.15]
["Walsh", "sofa", 48.82]
```

Screenshots of the IO Cost of the executed Queries :

query

```
C:\Users\swapn\AppData\Local\Programs\Python\Python38\python.exe "C:/Swapnali/MUN/Fall 2021/Databse applications/Database Final Project/COMP-6908-001 (Database Tech & Applications
With B+_tree, the cost of searching the attribute : sid ,operator : =, value : s23 on relation : Suppliers is 3 pages
Without B+_tree, the cost of searching attribute : sid , operator : =, value : s23 on relation : Suppliers is 13 pages
With B+_tree, the cost of searching the attribute : pid ,operator : =, value : p15 on relation : Supply is 8 pages
Without B+_tree, the cost of searching attribute : sname , operator : =, value : Kiddie on relation : Suppliers is 13 pages
With B+_tree, the cost of searching the attribute : pid ,operator : =, value : p20 on relation : Supply is 10 pages
Without B+_tree, the cost of searching attribute : cost , operator : >=, value : 47.0 on relation : Supply is 54 pages

Process finished with exit code 0
```