Write Optimize B+ TREE:

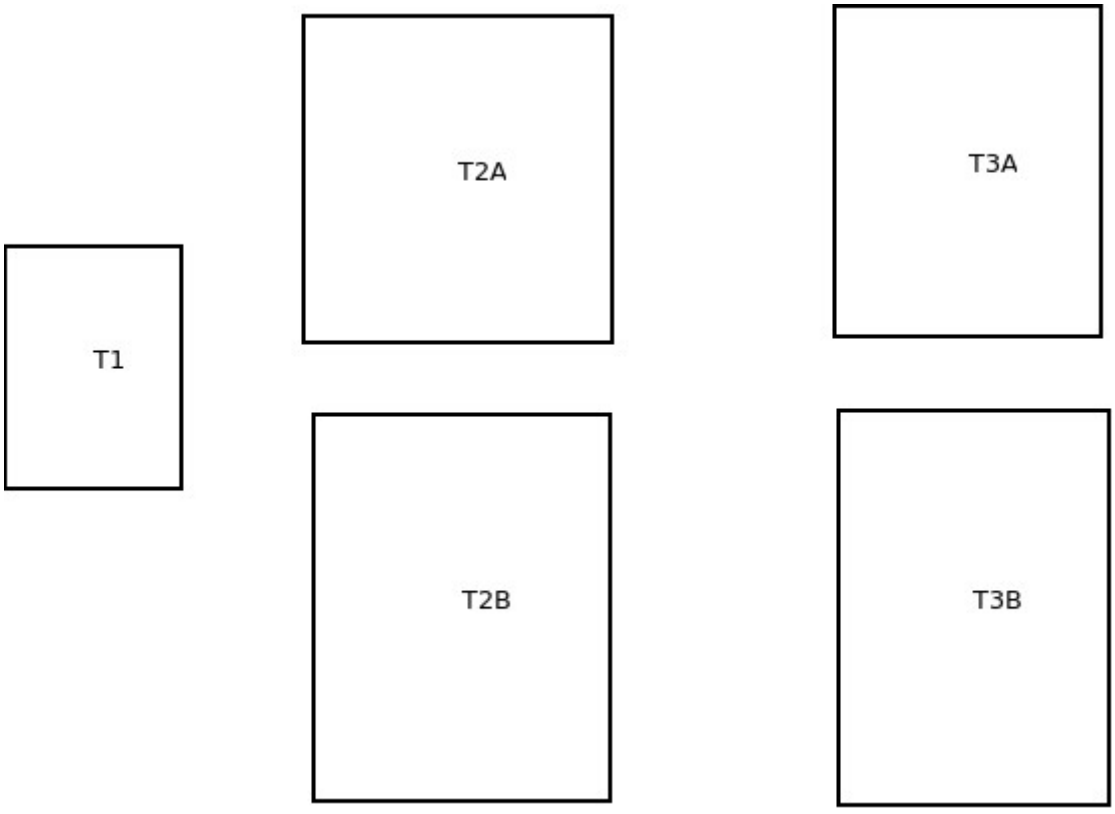
Submitted by:
Deepali Mittal – 153050016
Achala Bhati - 153050056
Sobha Singh - 153050065
Binapani Beria - 153050088

## Introduction:

In Today most of application require more insert than updates. So For this kind of application we can make write more optimize. In this we need make sure that writes should execute efficiently. We make sure that write is being done at immediate level we need not to take it from the next level before insertion. But In this case our select query can take time.

### What We should do?

So whenever a tuple comes we store the data in a table but after a certain table at level 1 will get filled so we need to partition it in level 2. Also we need to ensure that level 2 partition have large size than level.




So Each time when T1 get full then we need to send data to T2,then T3 and so on
Now to implement this we need use table inheritance.

## Table Inheritance:

so we can implement table inheritance for this purpose. In table inheritance when we insert into base table T, data actually gets inserted into either of its base tables.
Similarly for select queries, data is fetched not from T but from its corresponding child tables.

Implementation Details:

- Whenever user submits a query on base table, we will be storing his insert data into child table which inherits the properties of base table. But before inserting it into child table we are verifying if the size of table has reached a threshhold size or not.
  If size reached threshhold then we are copying the records of that base table into another base table at next level and deleting the records from first base table or merging the records of previous level table and next level table into second next level table (since 2 tables are present at next level)
  It is being assumed that each level has 2 tables.
- Consider level 2- if T1+T2.a OR T1+T2.b size is more than available in tables then do the above functionality for level 3 now and records will now be entered in T2.a or T2.b instead of T1.
- Similarly continue at every level.

## What function do we need to change in postgres:

1. As we are handling insert so we make sure any insert on base table T should go to table T_1 and if T_1 get full then we need to delete and insert into T_2A. And when T2_A got full then we will merge T1 and T2_A into T1

```
identify_insert(query_string): indentify insert query from other query
```

After this we will call T1 to check if there is space using select query on it. So if there is a space we will insert tuple in it.

```
exec_simple_query_select(new_str_select);:
```

```
PortalRunSelect :(use to get no of tuples so that we can find size of the table
before insertion)
```

```
 Ralation_name() we wrote our own fuction to find the table name from the user
query this we included in the postgres.c
```

Major changes has been done in postgres.c only.