Assignment-1Team:4

Bhulakshmi Makkena(12)

Harshi Priya Yarragonda(25)

Naga Mounica Dandlamudi(4)

1) Submit Cheat sheet generated through the In-Class Exercises. Identify the Best Indexing Techniques in Sequential and Parallel Manner

Single-level Ordered Indexes

Primary Indexes: The data file is ordered on a key field. Time complexity: O(n)

Clustering Indexes: Time complexity: O(n)

Secondary Indexes: A secondary index provides a secondary means for accessing a file.

Time complexity: O(n)

Multi-Level Indexes: Since a single-level index is an ordered file, we can create a primary index to the index itself. In this case, the original index file is called the first-level index and the index is called the *second-level index*.

Search Tree:

- Time complexity: O(log n)

Point Access Methods

B Tree: B tress are used for placing and locating the files in the database. It minimizes the number of transactions needed to search for a file.

	Average	**Worst case**
• **Space**	O(n)	O(n)
• **Search**	O(logn)	O(logn)
• **Insert**	O(logn)	O(logn)
• **Delete**	O(logn)	O(logn)

B+ Tree: B+ are similar like B trees, but they don't have key, value kind of relation like B trees. In B+ tress the data is stored in leaf nodes whereas intermediate nodes only store the key value.

	Average	**Worst case**
• **Space**	O(n)	O(n)
• **Search**	O(logn)	O(logn)
• **Insert**	O(logn)	O(logn)

• **Delete**	O(logn)	O(logn)

Multidimensional Hashing:

Grid File: It splits the space into multiple set of points called grid files. They help in minimizing the searches. Time complexity: O(n)

Hierarchical methods:

KD tree: It organizes the points in K dimensions. Useful when involved in multi-dimensional search.

	Average	**Worst case**
• **Space**	O(n)	O(n)
• **Search**	O(logn)	O(logn)
• **Insert**	O(logn)	O(logn)
• **Delete**	O(logn)	O(logn)

R tree: R trees are used in indexing multi-dimensional way like for co-ordinates and geographical representations to minimize the bounding regions**.**

Complexities:

• Search	O(log n)
• Insert	O(n)
• Delete	O(n)

Space Filling Curves:

Z-ordering

- Time Complexity: O(n)

Hilbert Curve

- Time Complexity: O(n)

The best indexing method would be R tree considering the complexities.

2) Indexing technique and implement it in spark map reduce using B+

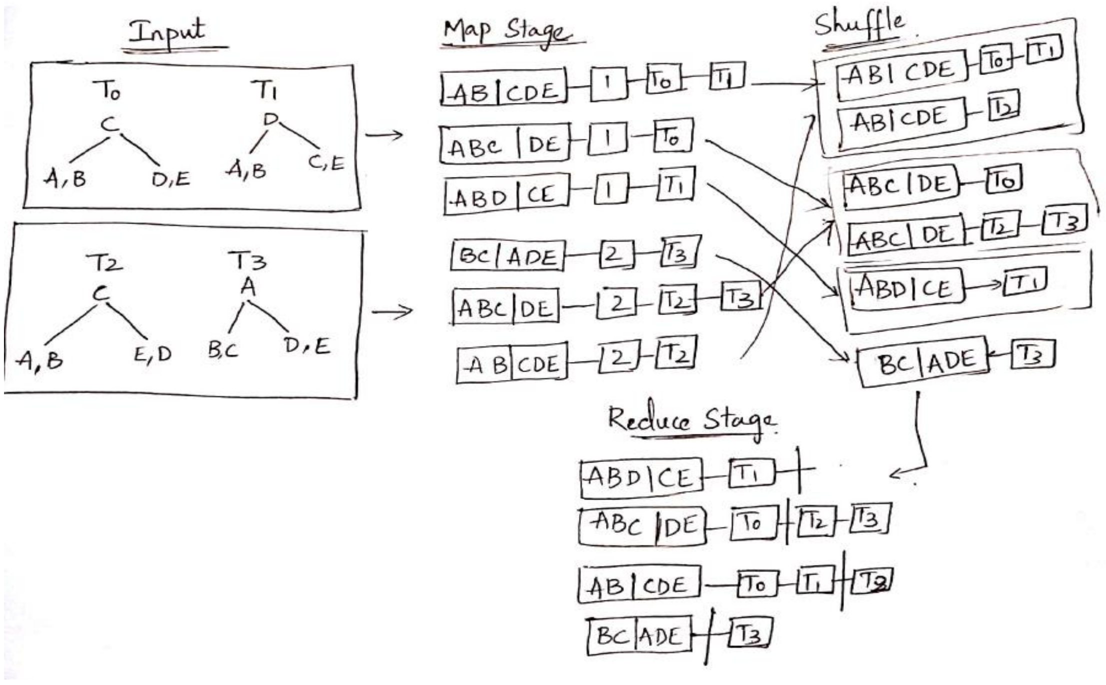
We consider two file that has two B+ trees in each file that has data in the nodes.

The two files are compared to get the hash table of both (i.e., each bipartition is the key and the values are the trees that contain that bipartition)

Two B+ trees can be compared and said to be equivalent when their roots are equivalent, each tree has same number of sub trees and ordered

In the fig, we took two files with two trees in each with nodes and internal nodes and trees in the first file are compared with the trees in the second file, the mapper emits the pairs like shown in fig and they are reduced based on the associated with a particular key.

Ex: The first reducer receives the list (AB,CDE)→(T0,T1);(AB,CDE)→T2 and outputs the final key value pairs (AB,CDE)→(T0,T1,T2)



Algorithm:

The mapper emits an intermediate key-value pair for each bipartition in the file

The reducer unions the trees for each bipartition

```
1. Class Mapper
    a. Method Map(file f,tree T):
        i. For all bipartition in tree t do
            1. EMIT(bipartition,(file,tree T))
2. Class Reducer
    a. Method Reduce(bipartition,Tree[T0,T1,…..])
        i. Val←empty
        ii. For all tree T belongs to Tree[T0,T1….] do
            1. Val←Val Union T
        EMIT(bipartition,Tree Val)
```

Complexity:

O((n(p+q))/m)

where O(n(p+q)) $\rightarrow$ total number of bipartitions that must be processed across p+q trees(p=(T0,T1 trees) , q=(T2,T3 trees)

m $\rightarrow$ number of mappers

Comparative evaluation with other techniques

Operations used for analysis are:

- Insertion of data files
- Deletion of data files
- Search of data files
  In addition to these, there are other conditions like the file grows beyond allocated size, overhead operations etc.

B+ tree is recommended but it is inefficient while handling duplicates since leaf nodes have data and intermediate nodes only have key values not the data values.

But compared to B tree, searching is faster in B+ trees so it is preferred.