Date: 05/09/2021 — 19. B and B+ Tree — DBMS Notes Written by Abhishek Sharma (GFG course)

Q) What are B and B+ Trees are ??

Ans :. They are n ways search Trees.

* Every Node of a Tree has some no. of values. and the no. of values that a node can contain is defined by **Branch factor**.

*

If Branch factor is b, then maximum b children are in the node and minimum $\lceil b/2 \rceil$ children.

Ex. if Branch factor = 4.

then minimum no. of children = $\lceil \frac{4}{2} \rceil$ = 2 children.

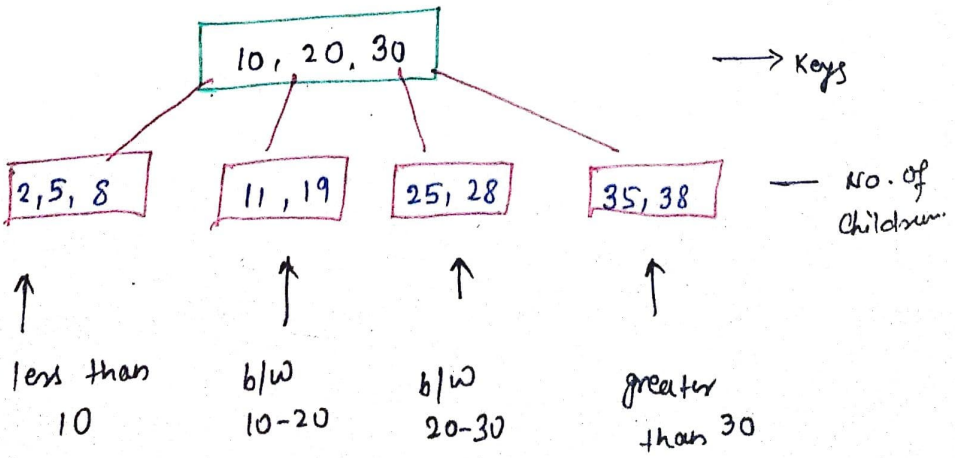
maximum no. of children = 4 children.

* No. of keys present in ~~a node~~ in n ways search Tree node = **(no. of children − 1)**

Example:

if a Trees has 4 children then No. of keys = (4−1) = 3

Ex.



* B and B^+ Trees are self Balanced Search Trees.

Ex. If There are millions of Record and millions of hight then they manage them by themselves.

* They make sure that hight never goes beyond $O(\log n)$ if there are n records to store.

Q) How B and B^+ Trees maintain their Height ??

Ans we will see via this example.

Given: 10, 30, 25, 80, 90, 100

15, 18, 20, 60, 110, 120.

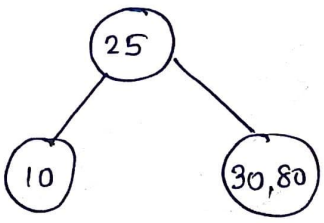
Branch factor = 4.

We have to insert these values in the Tree. Let's see.

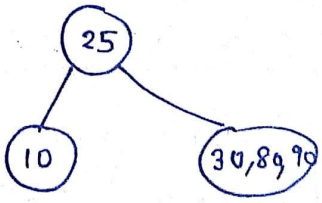
$\lceil b/2 \rceil$ minimum children formula is not applicable for root.

Now if we try to insert Here 80, then it violents the maximum children property. it has $b = 4$ children now.

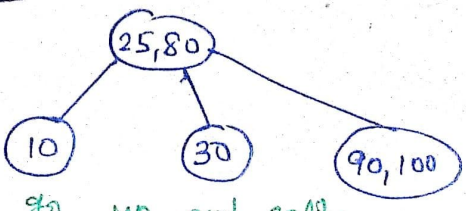
so we will split it.



Now we will insert 90

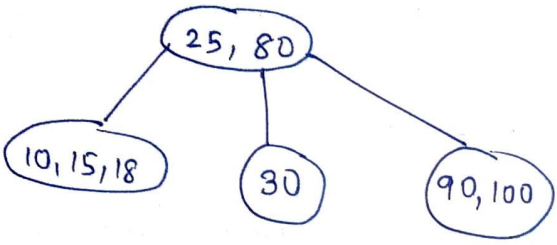


if we add 100 same problem



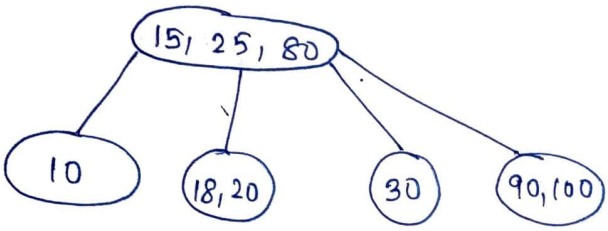
middle key will go up and split.

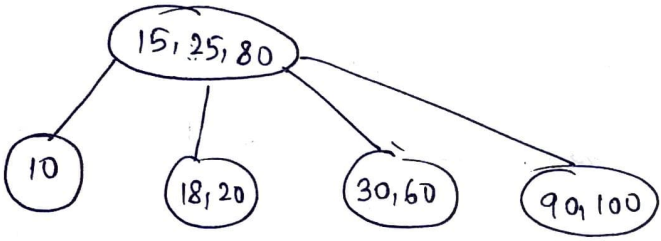
Now if we add 15, 18



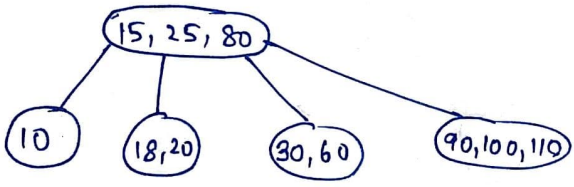
Now to add 20. and 60

// If we insert 20 then Rules violeted so we split it and 15 (middle key) go upward and splitting done.





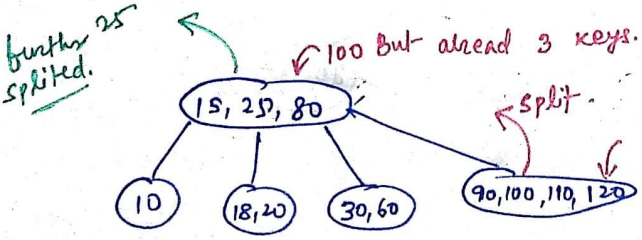
Now we have to add 110.



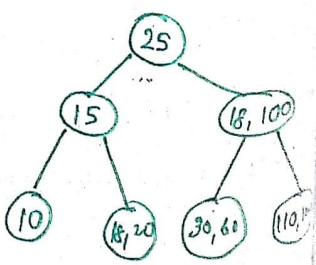
Now we have to add 120.

Mechanism · How we will add 120.

Finally Tree will be



Finally.



This is Our B Tree.

Abhishek Sharma Notes.

Note: B Trees Height grows upward. unlike the normal Binary Search Trees where Height grows downward.

in BST we ~~add~~ insert always from the leaf and that's why height always grows downwards.

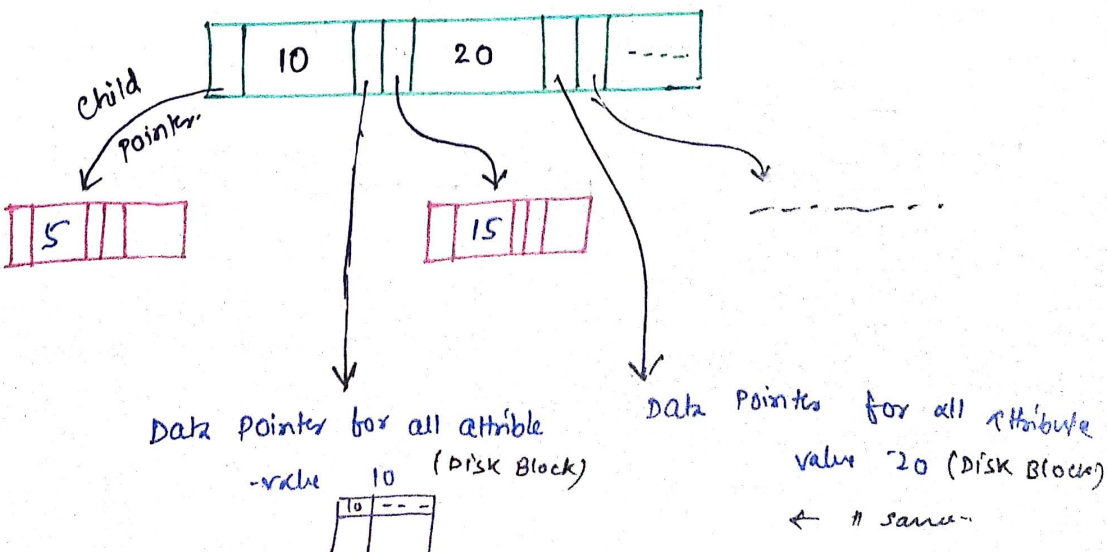
But Here in B and B+ Trees we always insert from root and Height growing from the root ↑

In the previous examples we saw that if the ~~Height~~ no. of keys increases Height grows upward.

* Similarly Deletion Operation, it (B and B+ Trees) shrinks the Height when no. of keys removed or deleted.

* B Trees are used To represent index files.
  - So They have pointers of actual records. actual ~~Records~~ of the disk where the records saves. and blocks
  - They also have pointers to children nodes.
  - So B Trees has 2 Pointers. i) Child pointer and ii) Data pointer for (Disk Block)

Q. How the B Trees store data. ??



Abhishek Sharma notes.

B+ Tree.

It is the improved version of B Tree.

It is the most used Tree in database for indexing like Oracle, Microsoft Sql Server. uses this (B+ Tree)

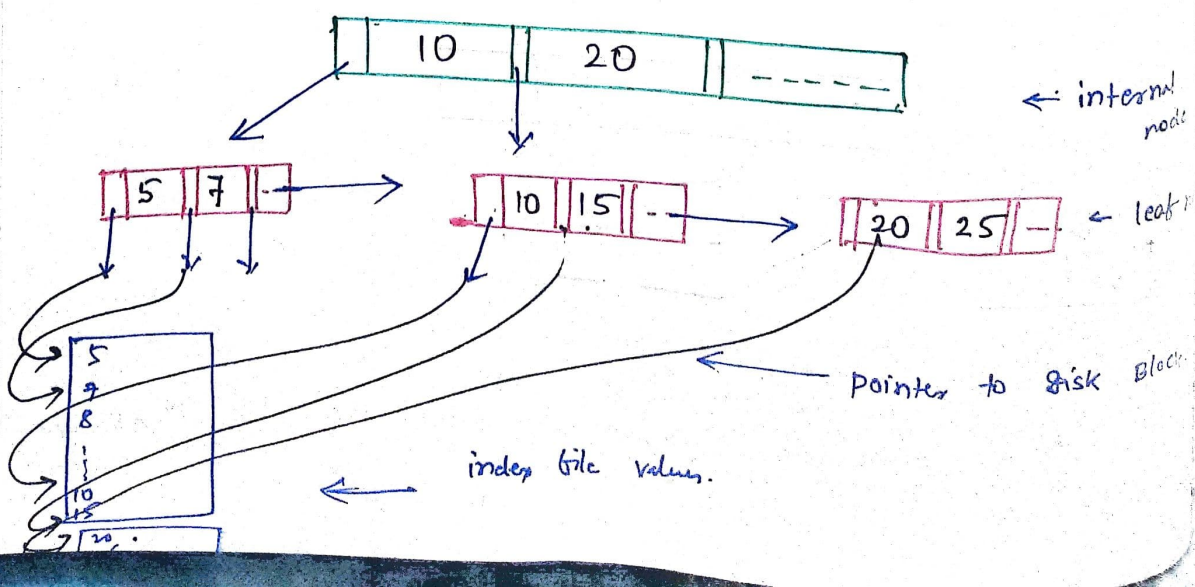
Idea of B+ Tree :-

All the leaf nodes are connected with each other and Leaf nodes allows you to sequencially access the data.

Since leaf nodes allows us to sequencially access the data so, every data item has to be present in the leaf nodes. Whatever present in the internal node is also to be present in the leaf nodes. so that we can access all the table entries in a sequencial manner.

* B+ Trees are index blocks.

B+ Trees.



* Advantages of B^+ Trees over B Trees.

you do not need 2 pointers. you only need one pointer with every key attributes also we have sequencial Access.

* Drawbacks of B^+ Trees.

This might happen that key value might be present Two times. (i.e. all the key value that is present in the internal nodes will be also be present in the leaf nodes also).

* B^+ Tree is the fantastic Data structure that works upon multilevel indexing. They grow and shrinks automatically. They have only one pointer. bcz. your internal nodes have block pointers pointing to disk blocks. which have index file values. (as we saw in the diagram).

* Both B and B^+ Trees insure that every leaf node present at the same level.

* Height of these (B and B^+) Trees is actually $\log n$.

Abhishek Sharma Notes.

20. ACID properties.

Transactions and concurrency control (ACID property).

Transaction: A set of logical instructions for the Database that should happen together. Either all of them Happen or non of them will happen.

Example: In a Bank data of Two people x and y

x has initial balance 500 Rs. and y has 200 Rs.

Our task is to send 100 Rs. from x to y.

So the instructions are.

Before Transactions:

x : 500
y : 200

Read (x)

x = x - 100

Write (x) : 400

Read (y)

y = y + 100

write (y) : 300

After Transaction.

x : 400
y : 300

This is the whole process for Sending 100 Rs. from x to y.

So Transaction says either all steps will happen or nothing will happen.

Bich में stop नहीं करेंगे Step.

* Most of the databases (DBMS) have the syntax available for Transaction

You have to give instructions to DBMS that this is the Transaction, either it should happen completely or should not happen at all.

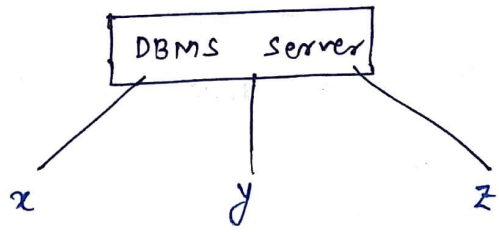
Transaction is one problem that your DBMS should solve

Another problem that comes with Transaction is concurrency.

Concurrency: concurrency says that if a DBMS server is doing multiple transactions. Then they all should be doing in concurrent way.

Same as operating system i.e. when a process that is doing IO, you bring another process that can use CPU.

Example.



⇐ DBMS Server

Multiple.

⇐ Transactions

Transaction 1.

x = 500 Rs.

y = 200 Rs.

Transfering 100 Rs. from x to y.

Transaction 2.

z = 200 Rs.

Transfering 200 Rs to x account.

Let's Discuss if a Transaction is happen in Transaction 1. and it stops at Deduction Rs 100 from x account and at the same time z's account Transfer the money to x account

So, we might not get our Desireed Result.

i.e. These are the problems happen in Transaction and concurrency.

To get Rid of these types of problem we have ACID properties. in our DBMS.

These are 4 certain things that will insure Transaction and Concurrency in the DBMS.

These are. ACID properties.

1) Atomicity
2) Consistency
3) Isolation
4) Durability.

Atomicity: Either the whole Transaction happen or nothing will happen.

Consistency: your Database should be in consistent state After the Transaction has happened.

Example: In a Bank, two people x and y have money 500 and 400 respectively. After any Transaction happen. then the Sum of them remain Constant. It should not change.

x = 500	= Sum = 900
y = 400	

At Befor Transaction.

if any Transaction Happen

Sum will be 900.

This is consistency says.

Money should not go anywhere. It should be constant before and after the Transaction. This is consistency says. your DBMS should be in consistent state.

Isolation: There are multiple transaction that is doing by your DBMS server. Isolation says that every Transaction feel like that it is running alone. Any Transaction should not be impacted by another Transaction. All the Transaction should be running in alone ~~seperately~~ and in a Sequencial manner.

Ex: If you are doing Internet Banking. and at the same time anyone sending you money. So Here are Two Transaction is happening at the same time. But your internet Banking Transaction will be alone and safe. It will not an effect of anyone sending you money. at the same time.

Duarability: Duarability says that once your transaction is done, whatever will be the changes. it should be in ↓ Stay your DBMS. It should not be lost. Changes should be durable. It should not be lost.