#ASLI ENGINEERING

Why databases use B+ Trees

SWIPE

BY

ARPIT BHAYANI

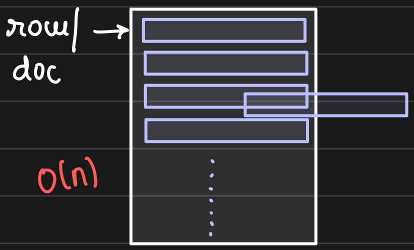
Why databases use B+ trees to hold data?

SQL Databases are known to use B+ trees to hold the data

* even non-relational databases leverage them to store data !!

let's start simple

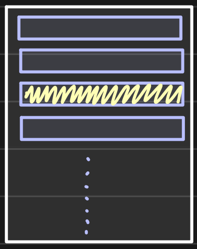
Say our table/records are stored in one file sequentially



O(n)

Insert

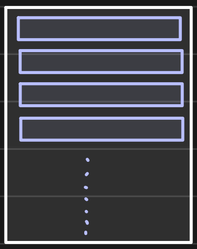
cannot efficiently insert
in the middle



O(n)

Update

can override within
the same width



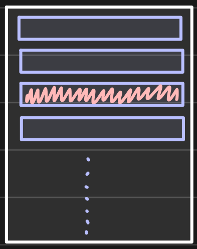
O(n)

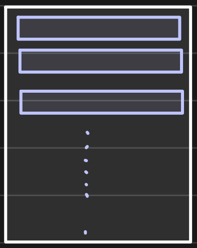
Find One

Linear scan

Range Queries

Possible only when
rows are ordered by it





O(n)

Delete

create a new file
without that entry/row

ARPIT BHAYANI

O(n) complexity for every operation is far too much !!

So, can we do them in O(logn) ?

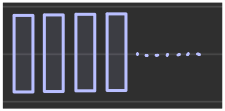
B+ Trees

Rows or documents of a table are clubbed in B+ tree nodes

eg: if 1 B+ tree node is 4KB big

and row/document size is 40 B

then each node will hold max ~100 rows



* size of B+ tree node ≈ disk block size

↳ in one disk read we read 1 node ≈ 100 rows

Thus, one table is just a collection B+ tree nodes on the disk

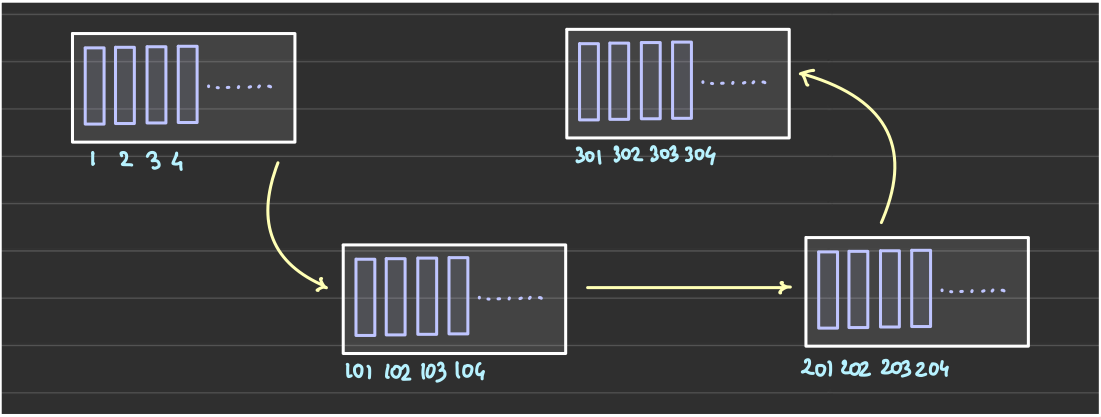
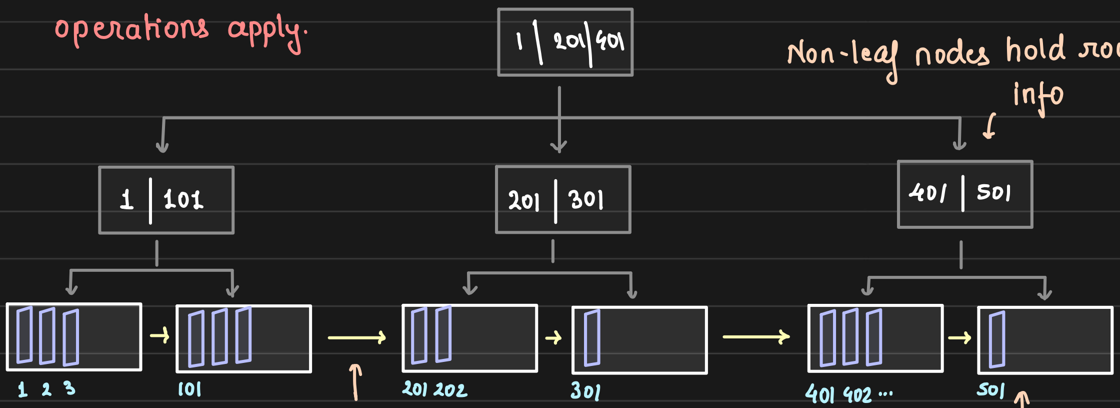


Table is always arranged by its Primary Key and hence

the B+ tree nodes (leaf) are connectd accordingly

Table as B+ Tree

Standard B+ tree operations apply.



Every B+ tree node is serialized and stored on disk

Non-leaf nodes hold routing info

leaf nodes are linked so as to enable linear traversal.

leaf nodes hold the actual rows

Find One By ID

Traverse from the root node, reach the leaf, read the leaf, and extract

↳ Read each node from disk, understand, and act

Insert Find a leaf node where value/row/doc fits, update. and flush

Update Find leaf that holds the row, read block, update, and flush

Delete Find leaf that holds the row, read block, remove, and flush

Range eg: id in (100, 600)

Find leaf that holds row 100. traverse linearly until you reach row 600.

ARPIT BHAYANI