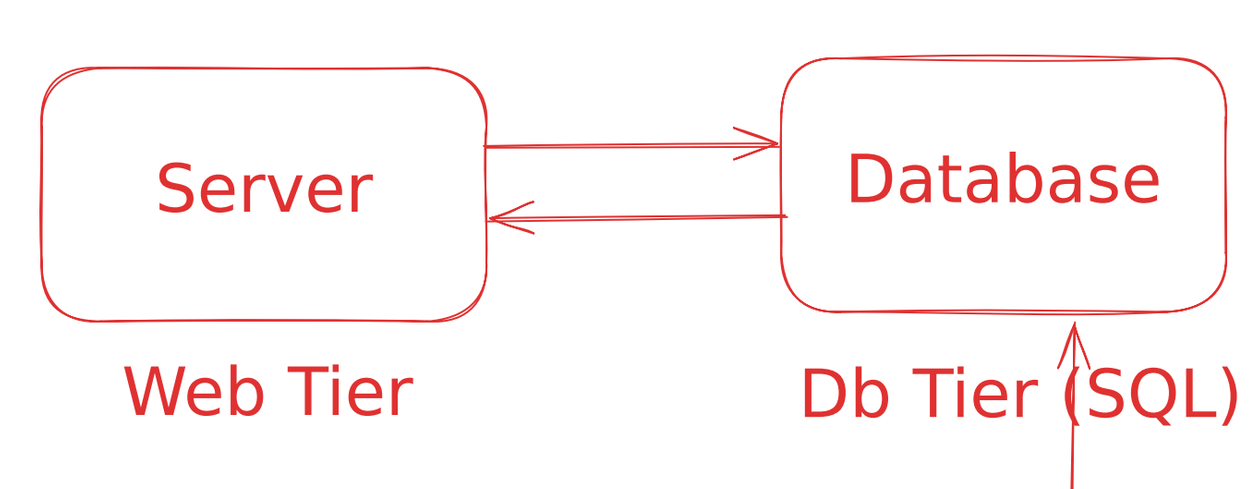
SQL / Relational Architecture



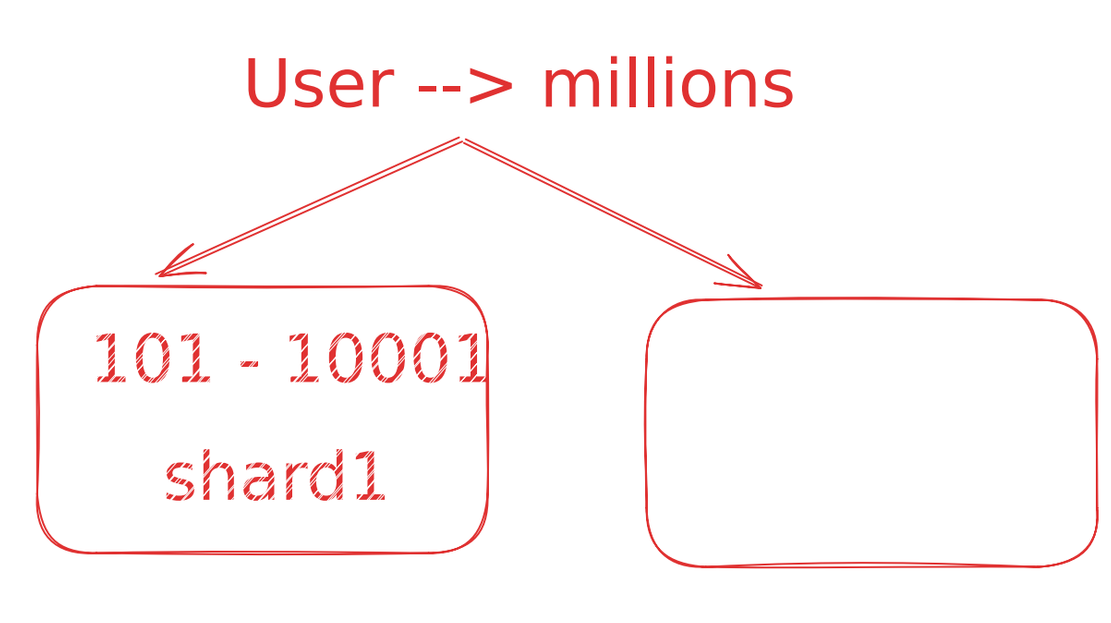
Master-Slave Architecture

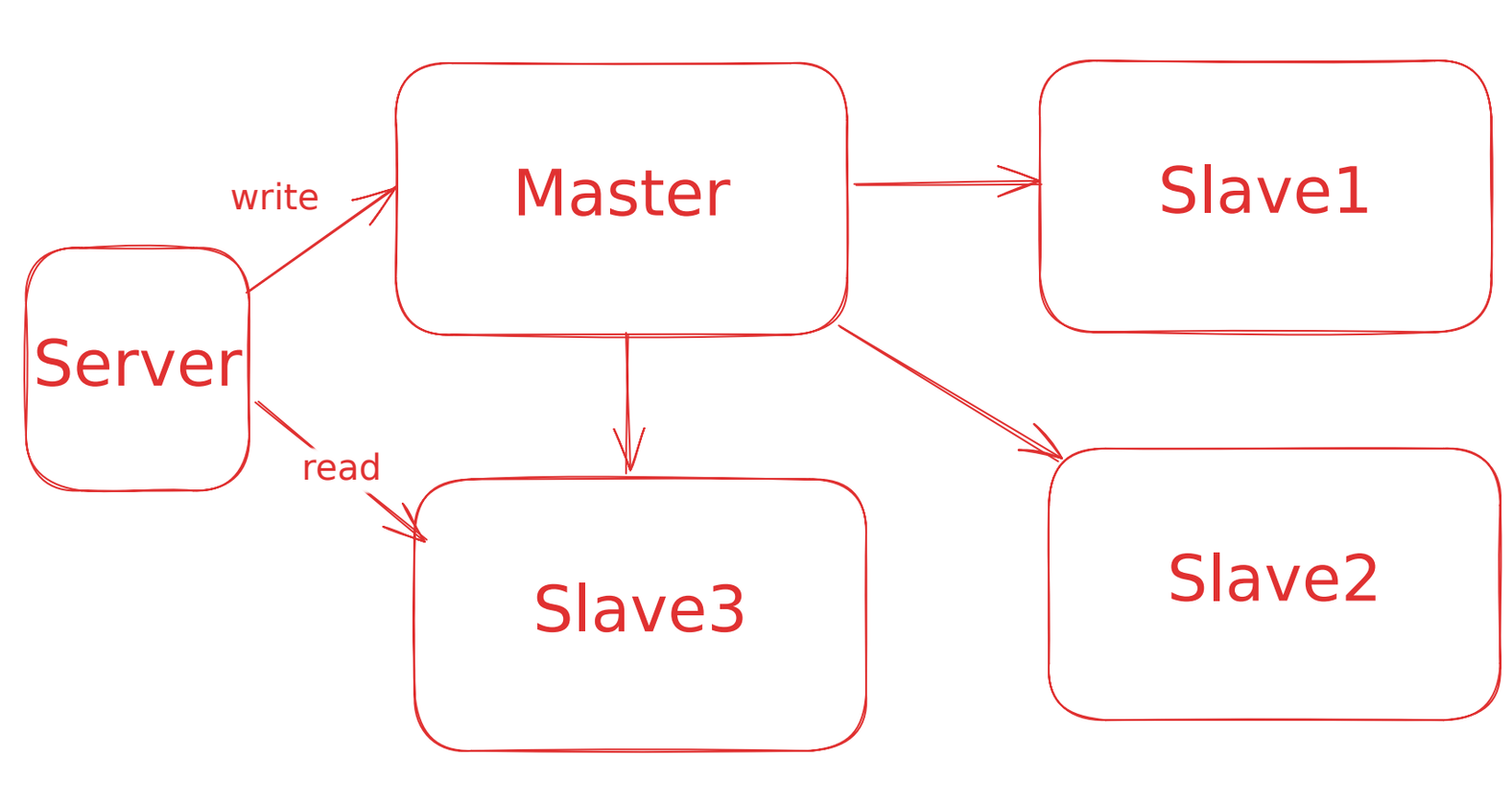
PK

User

userId	name	email
101	Aditya	xyz
102	Rohit	ert
103	Mukesh	wer

Horizontal Scaling --> Sharding





Views. --> Virtual Tables

SQL DB

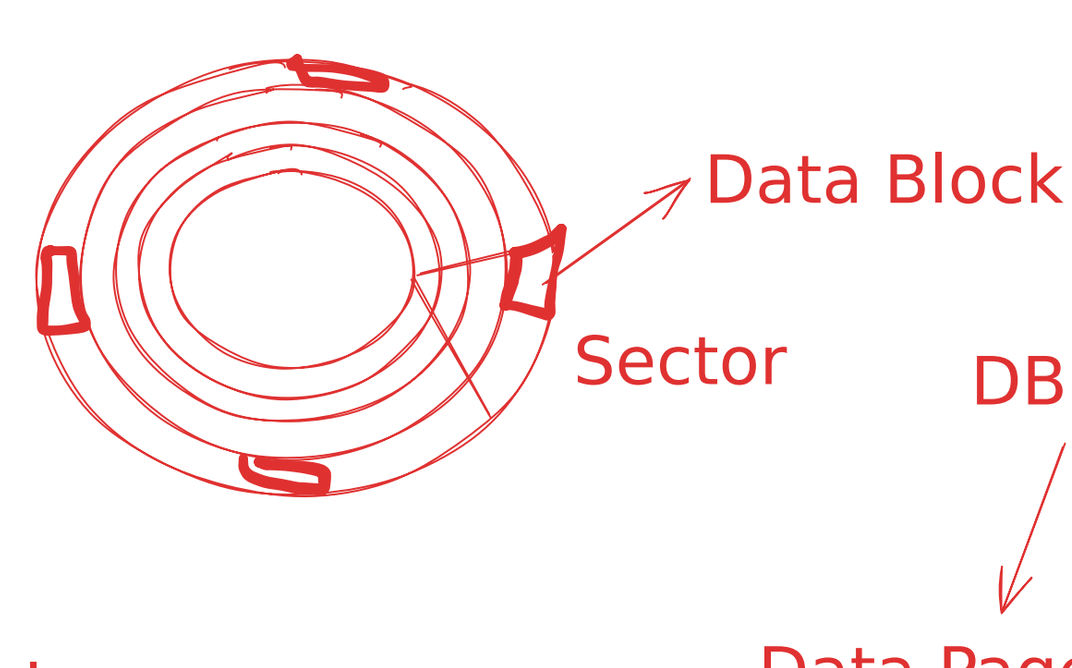
PK

User

userId	name	email
101	Aditya	xyz
102	Rohit	ert
103	Mukesh	wer

Relational DB. --> Rows and Columns
Logical Arrangement
Stored not in Volatile memory

Stored in permanent memory (Disc)



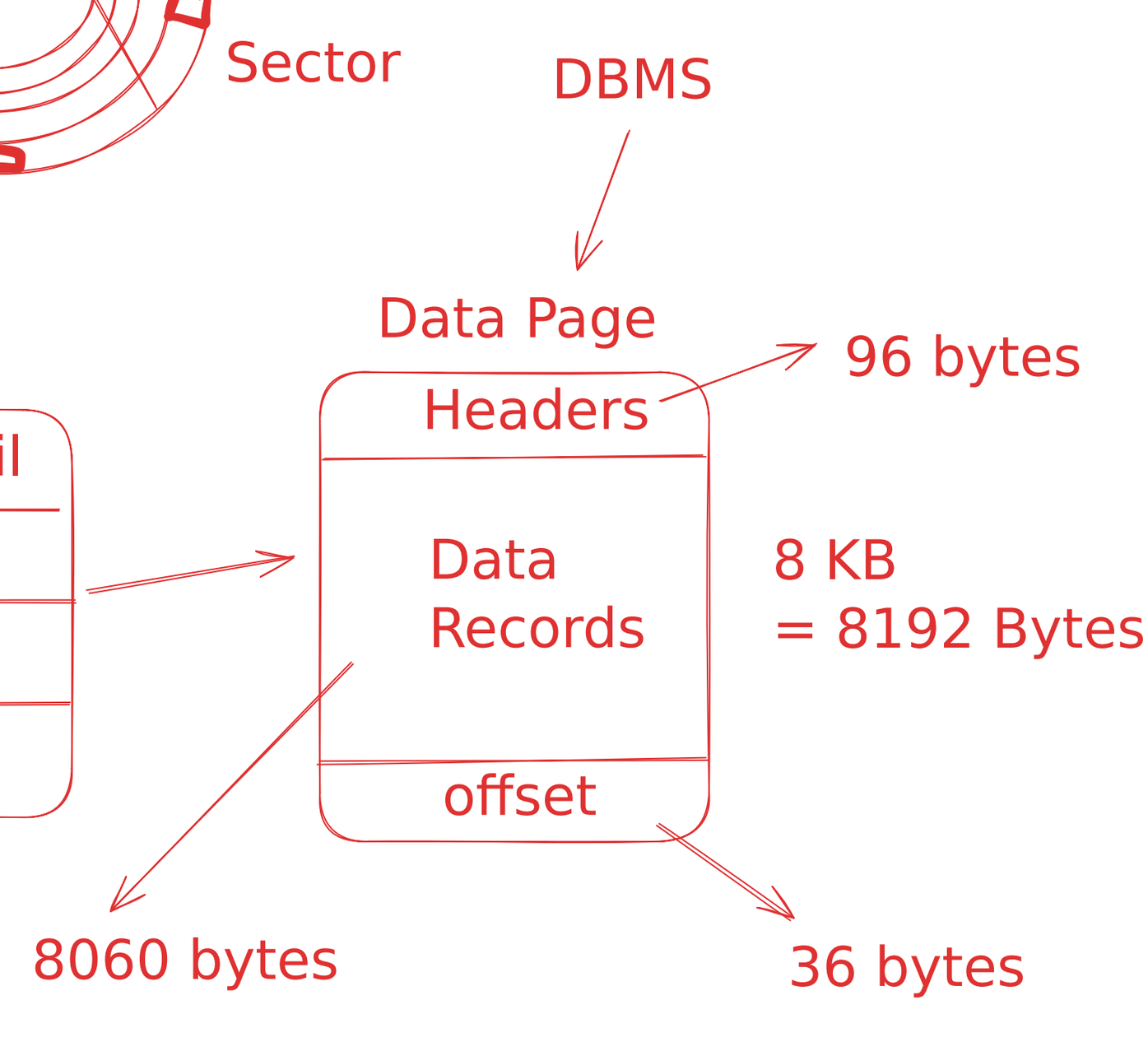
Table

PK

userId	name	email
101	Aditya	xyz
102	Rohit	ert
103	Mukesh	wer

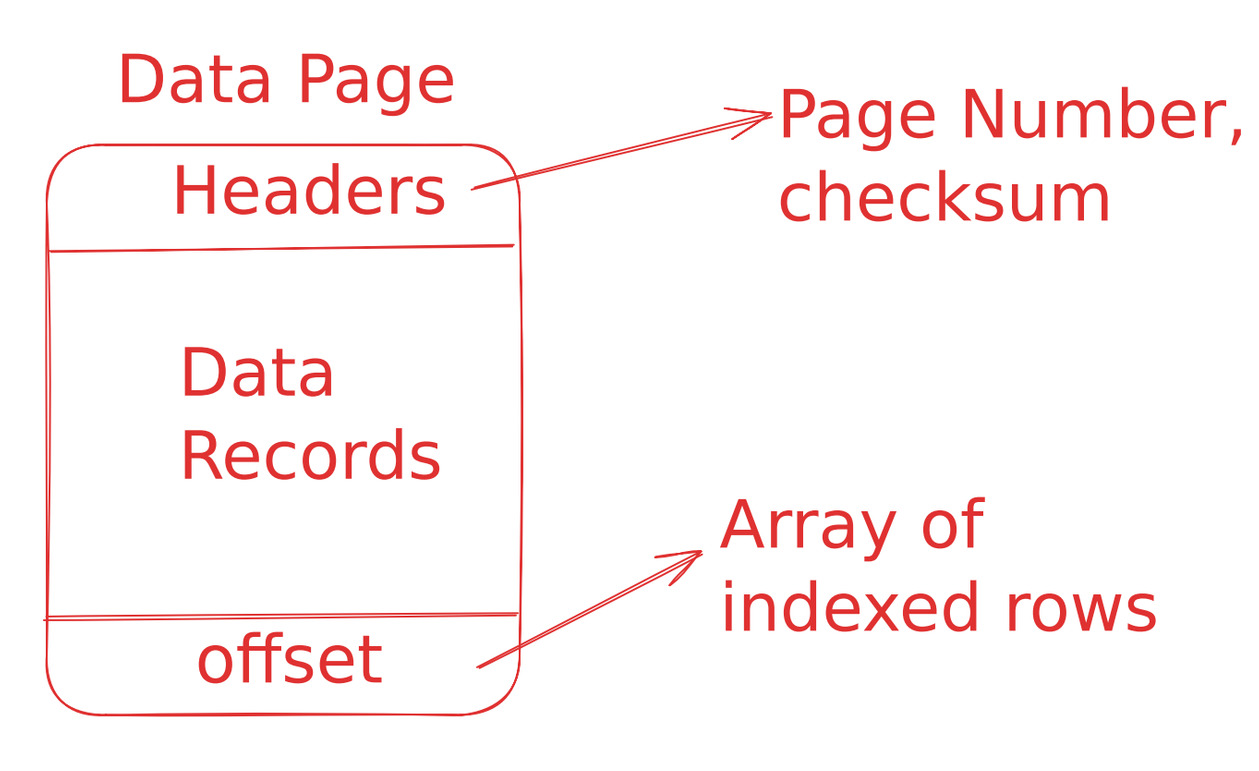
1 Million Rows

row = 32 bytes



1 Data Page = ~251 rows.

Total page. = 3984 pages



Data pages are managed by DBMS.
But practically, All data pages will be stored in memory.

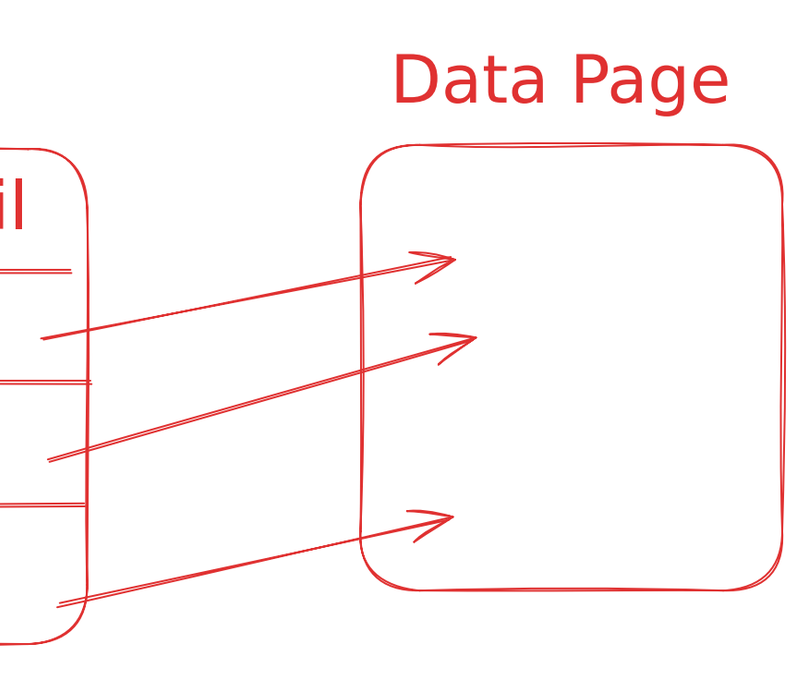
1 Data Block --> 8 KB - 64 KB

If 1 Data block = 8KB
1 Data block = 1 Data Page

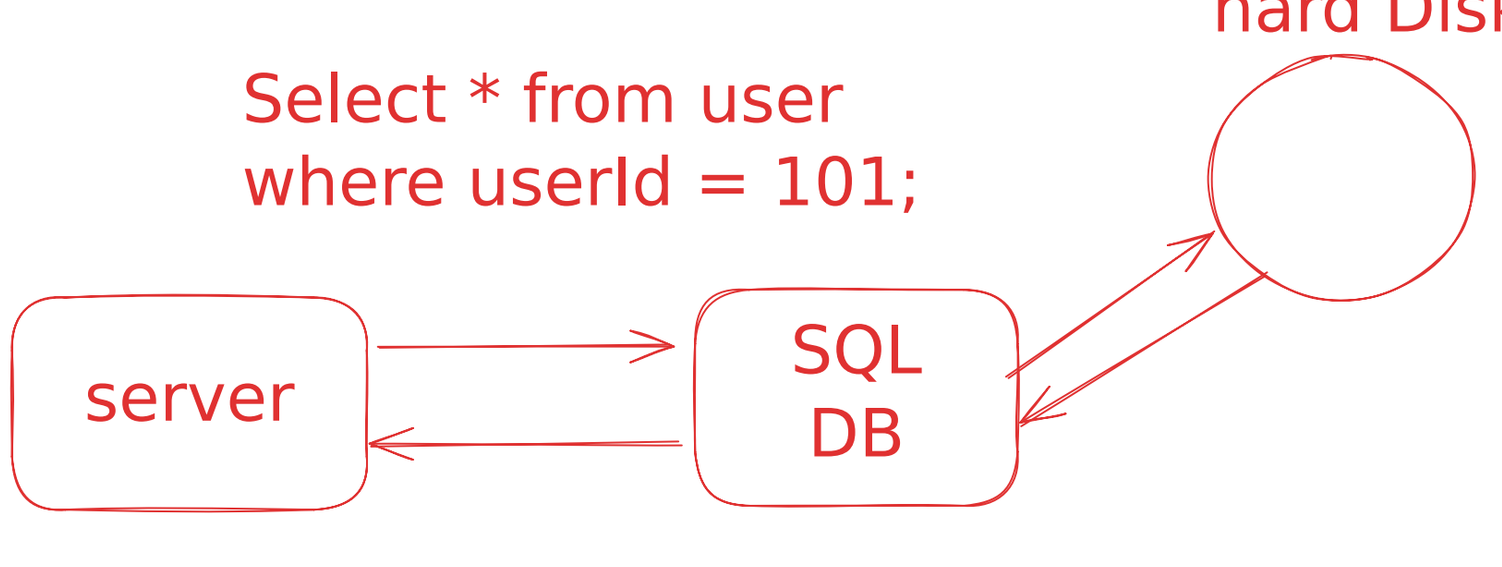
Data Block :
It is the smallest possible value we can expect from an IO operation.

user

userId	name	email
101	Aditya	xyz
103	Rohit	ert
102	Mukesh	wer



Select * from user
where userId = 101;



Data Block 1 - page 1, Page 2
Data Block 2 - page 3, page 4
Data Block 3

O(N) time complexity --> 10 million

Data Blocks are managed by Hard disk.
DBMS manages Data Pages.

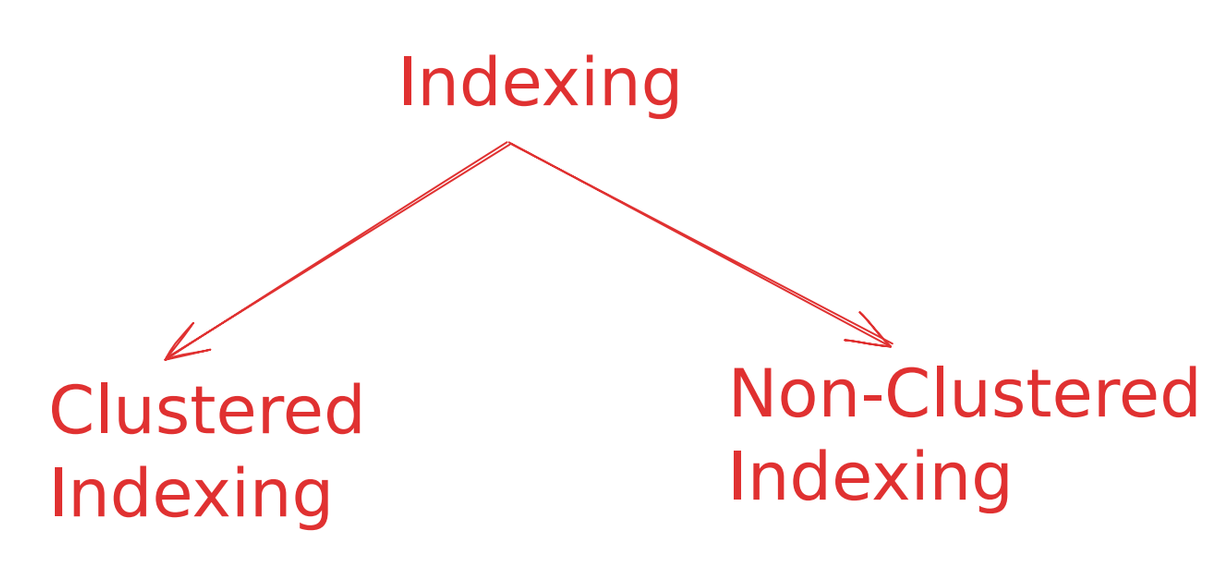
DBMS has a mapping table (User)
✓ Data page 1 ---> Data block 1
✓ Data page 2 --> Data block 1
✓ Data page 3 --> Data block2
....
Data page 3984 --> Data block 1992

Indexing :
used to increase the performance of our Db Queries.
--> Making Queries run faster.

O(N) --> Reduce my search result.

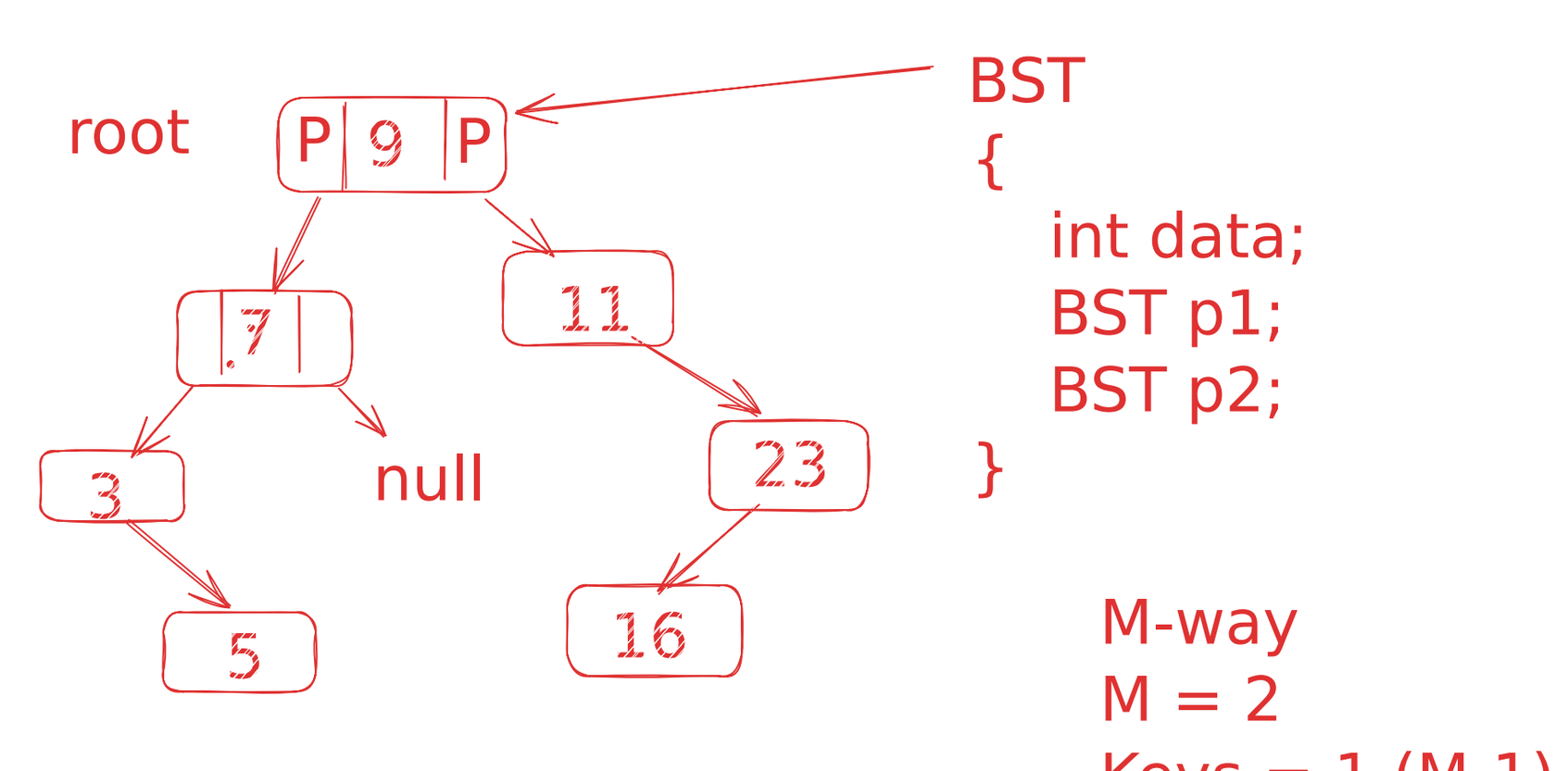
DSA --> Binary Search /Binary Search Tree (BST)
B+ Trees are used for DB indexing.

O(N) --> O(logN)

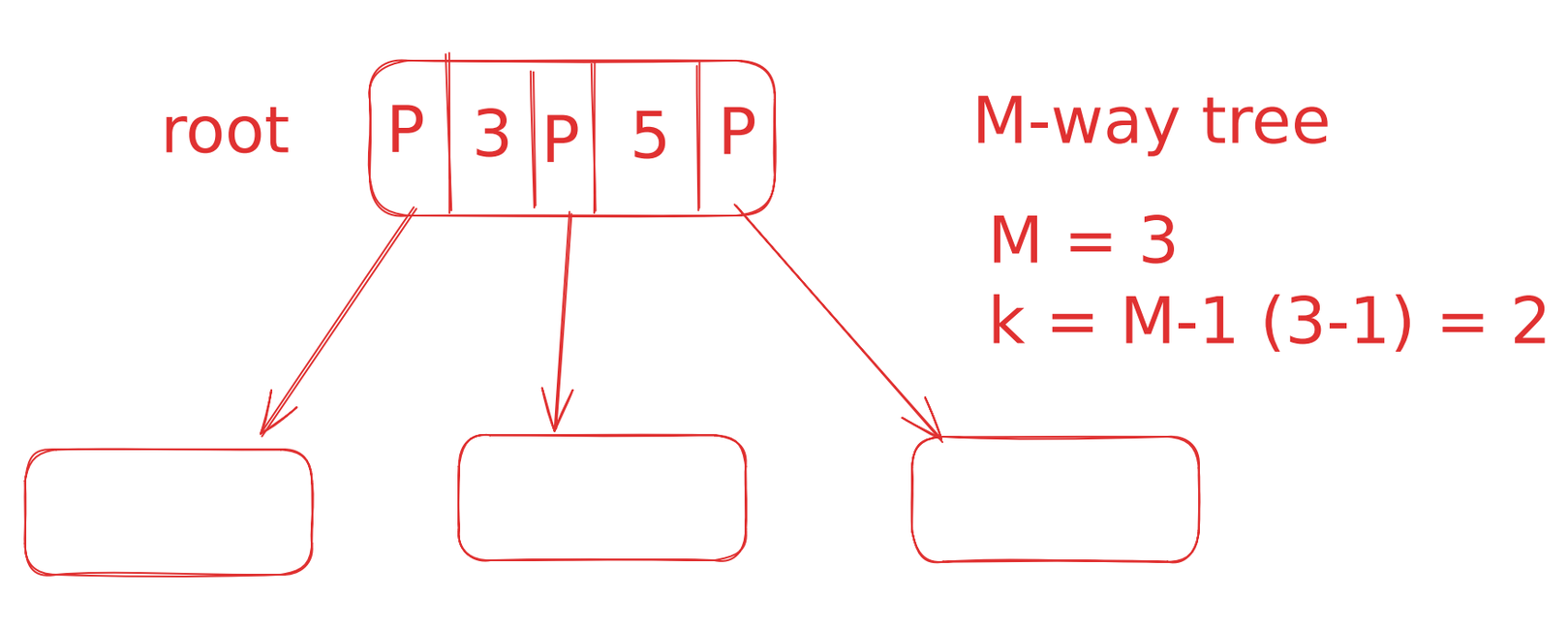


B Trees

9, 11, 7, 3, 23, 5, 16

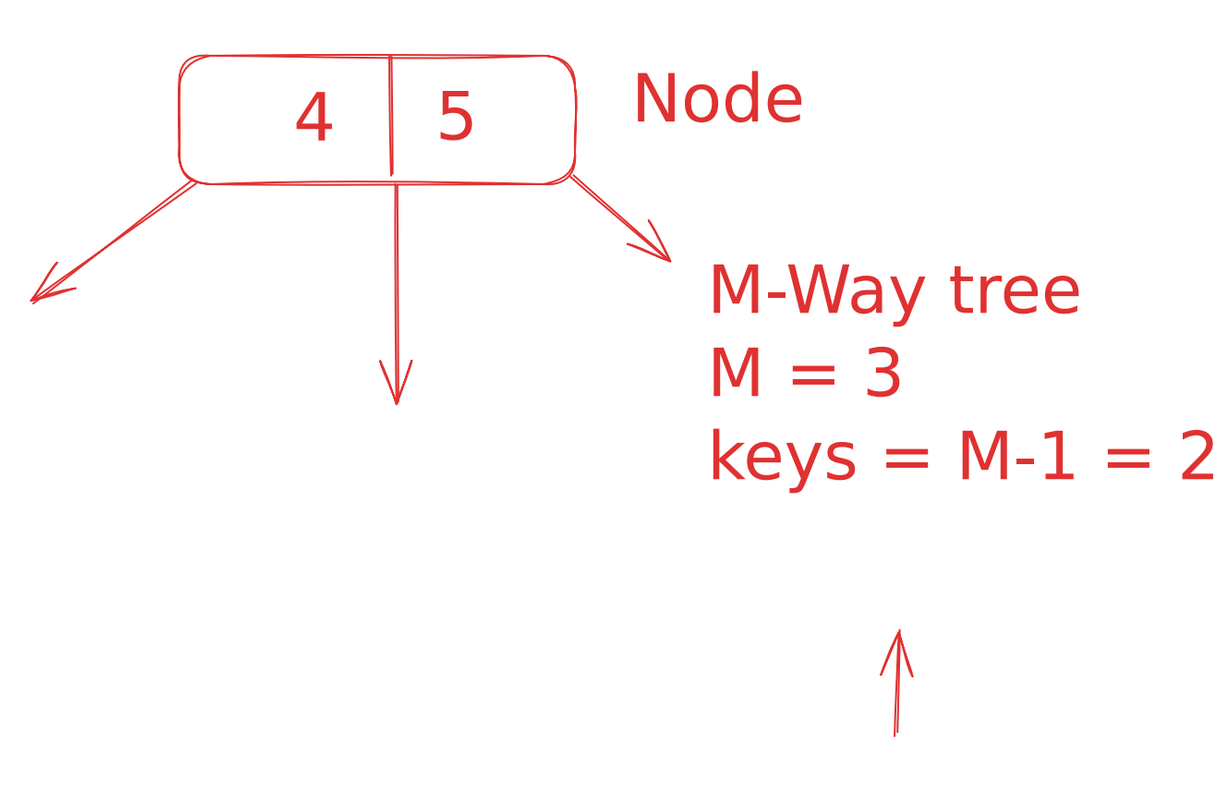


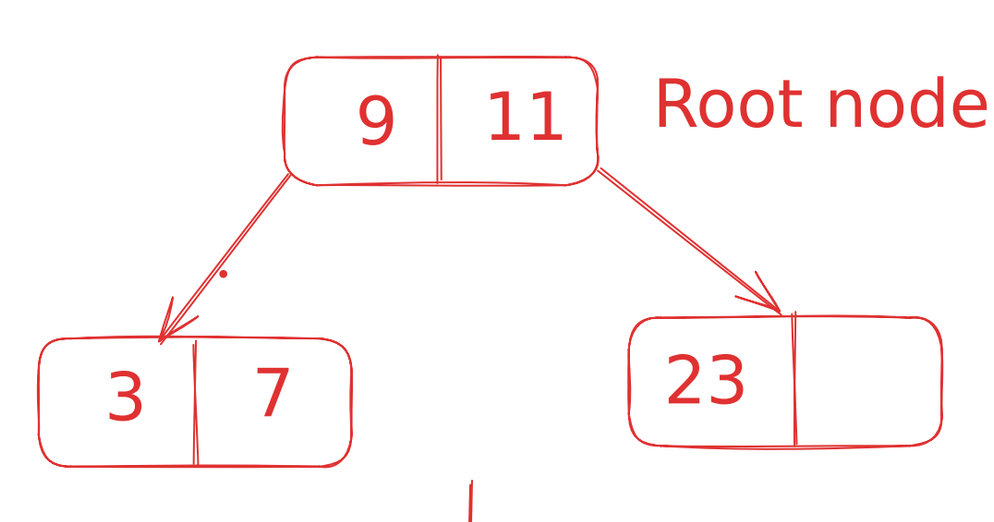
BST --> Binary Seach Tree (2)
TST --> Ternary Search Tree (3)
M- way Trees --> M children (M)




```
{
    int data1;
    int data2;
    Btree p1;
    Btree p2;
    btree p3;
}
```

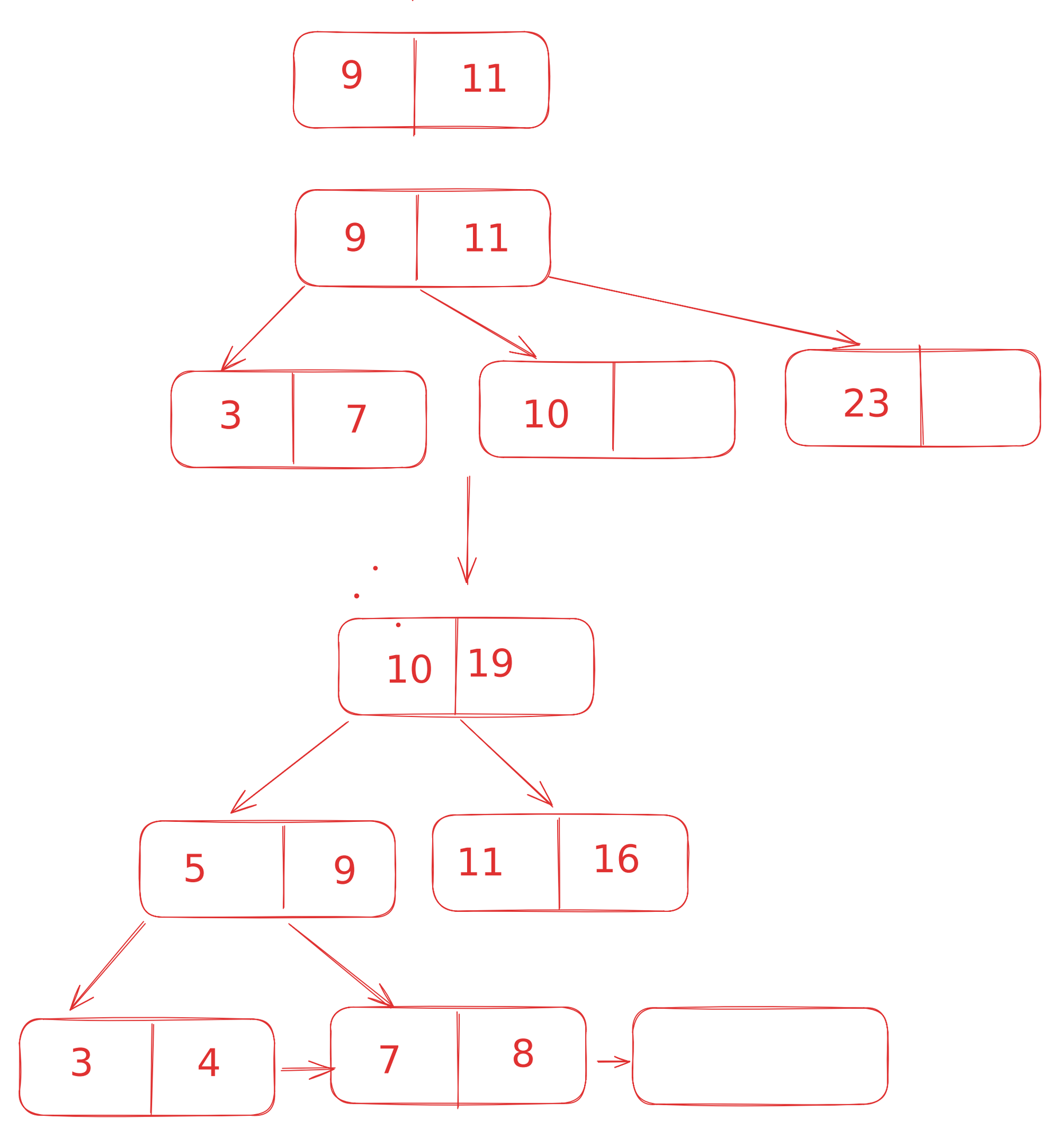
B Trees



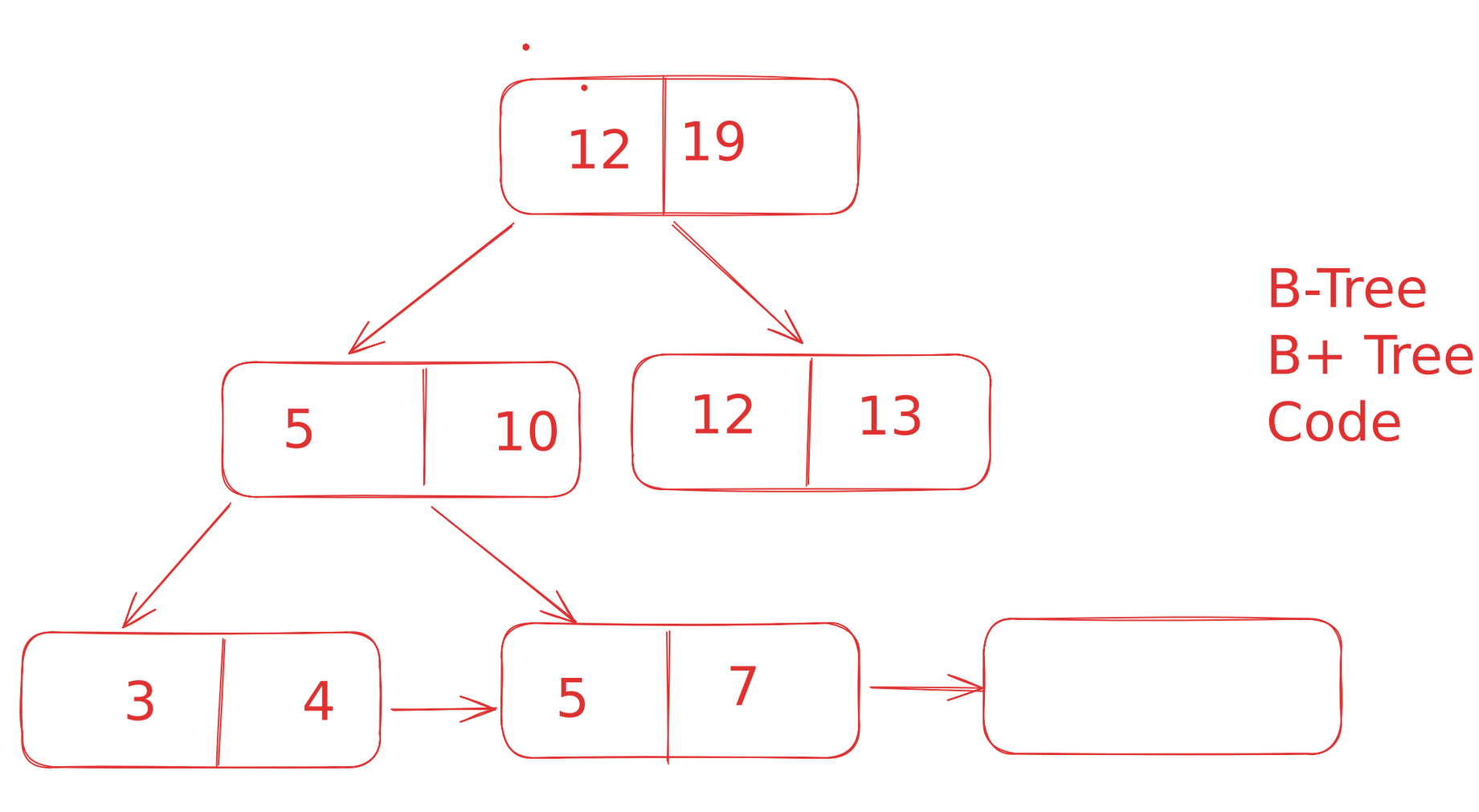


B-Tree Split up

9, 11, 7, 3, 10, 23, 5, 16, (12, 34, 1, 89)



B+ Trees
- leaf nodes connected
- Intermediate nodes also store parent left value

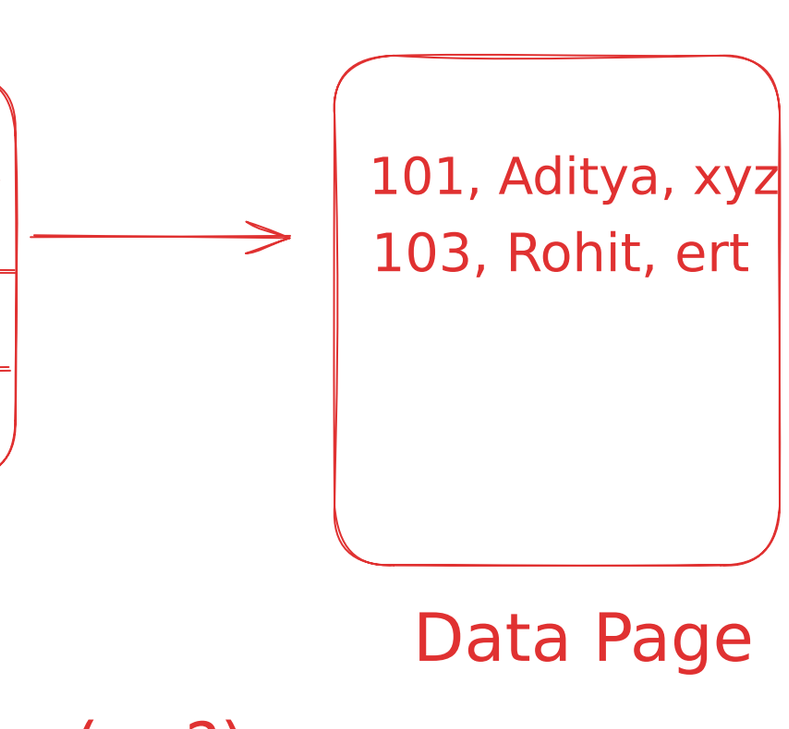


Indexing
↓
B+ Trees

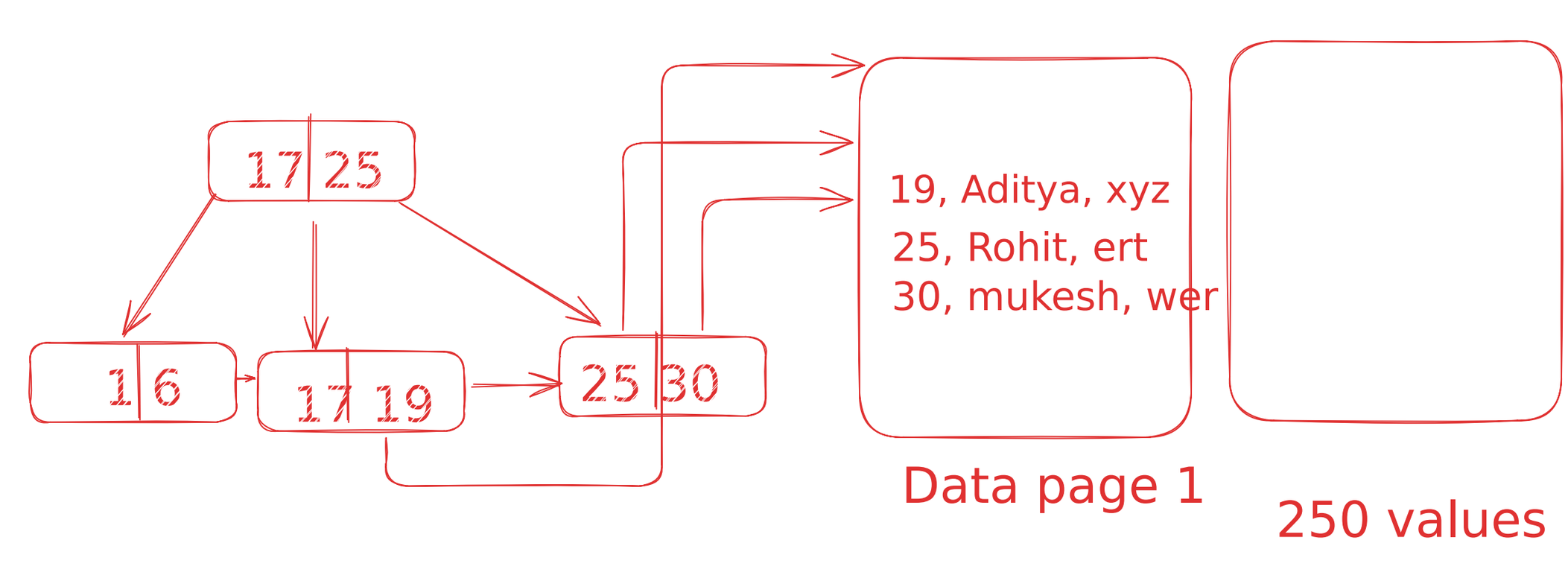
PK user

userId	name	email
19	Aditya	xyz
25	Rohit	ert
30	Mukesh	wer

1 Million records



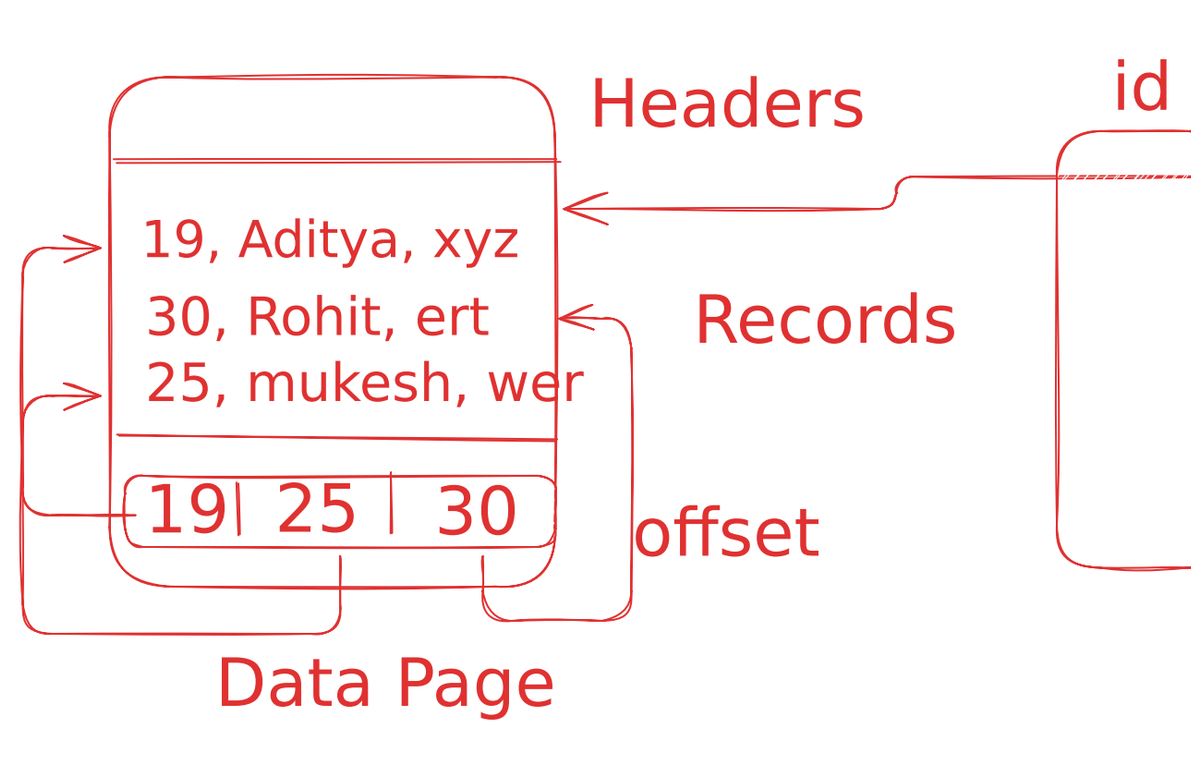
M_way (m ?)



select * from user
where userId = 19;

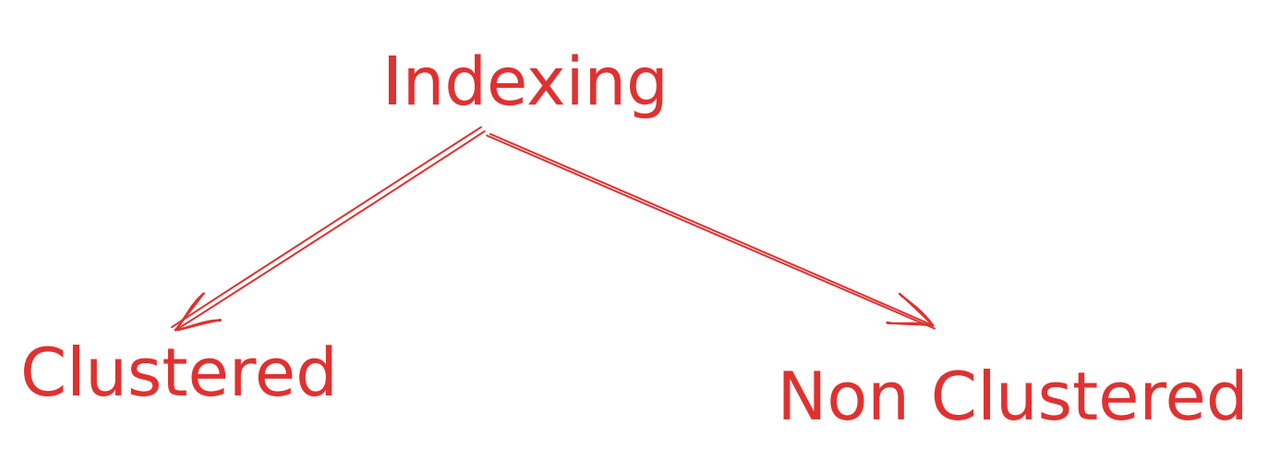
DBMS
data page --> Data block

B+ Tree
Range Query --> 101 - 205



userId	name	email
19	Aditya	xyz
30	Rohit	ert
19	Mukesh	wer

1 Million records



Clustered: Leaf nodes of B+ Trees store complete row information

Non Clustered: leaf nodes of B+ trees store only id information (page no.)

Indexing :
Clustered vs Non-Clustred
--> There can be only 1 clustured index per Table.
--> There can be multiple non clustred index per table.

--> What is the issue to index each column??
--> It will require a lot of space.

NO-SQL DB
Mongo Db --> B+ Trees
Cassandra DB --> LSM Tree