COMPUTER SCIENCE

Database Management System

File Org. & Indexing

Lecture_6

Vijay Agarwal sir


TOPICS TO BE COVERED
01
Multi level Indexing
02
B & B+ Tree

B Tree

ORDER : 5

9, 1, 7, 3, 5, 6, 11

File org & Indexing

File org.

- Blocking factor
- SPANNED & UNSPANNED ORG.
- Ordered file $\lceil \log_2 B \rceil$
- Unordered File (Heap) $B/2$ or R (Worst case)

Indexing

One Index Record size = key + Pointer

Index Block Access $= \lceil \log_2 B_i \rceil$

To Access a Record using Index $= \lceil \log_2 B_i \rceil + 1$

Dense (# Records)

SPARSE (# DB Blocks)

(# Entries)

PI (key + ordered File)

CI (Non key + ordered File)

SI (Non key + unordered)
(key + unordered)

MultiLevel Indexing

- Apply Primary Index on Base Index file.
- till All Index entries fit into 1 Block

Cost = n+1 (n: # of Level)

B Tree Definition

Dynamic Multi Level Indexing.

- B Tree
- B+ Tree

B Tree Definition

Dynamic Multi Level Indexing.

↳ B Tree

Minimum

ORDER : P

$B_p = \lceil P/2 \rceil$

keys $= \lceil P/2 \rceil - 1$

maximum

ORDER : P

$B_p : P$

key : (P-1)

Root

min: 2 B_p to max: P B_p

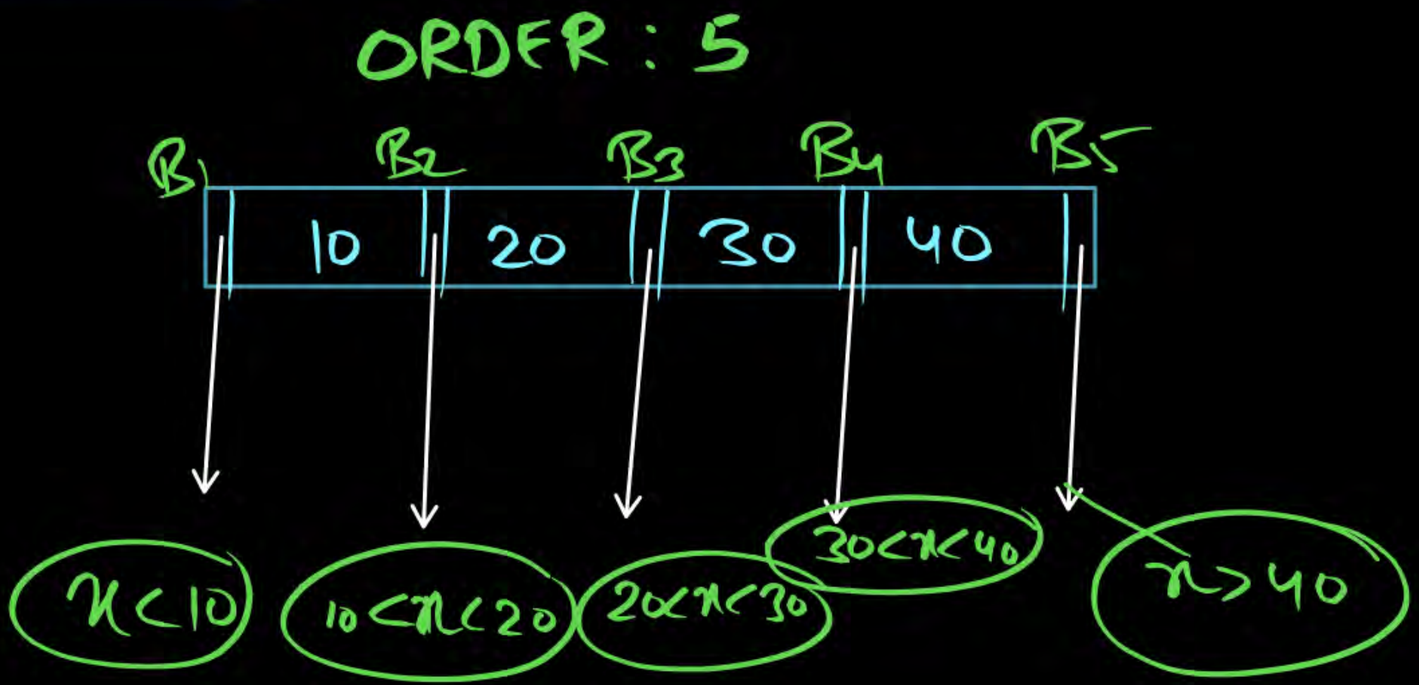
1 key to (P-1) keys.

B Tree Definition

ORDER P: Maximum P Block Pointer.

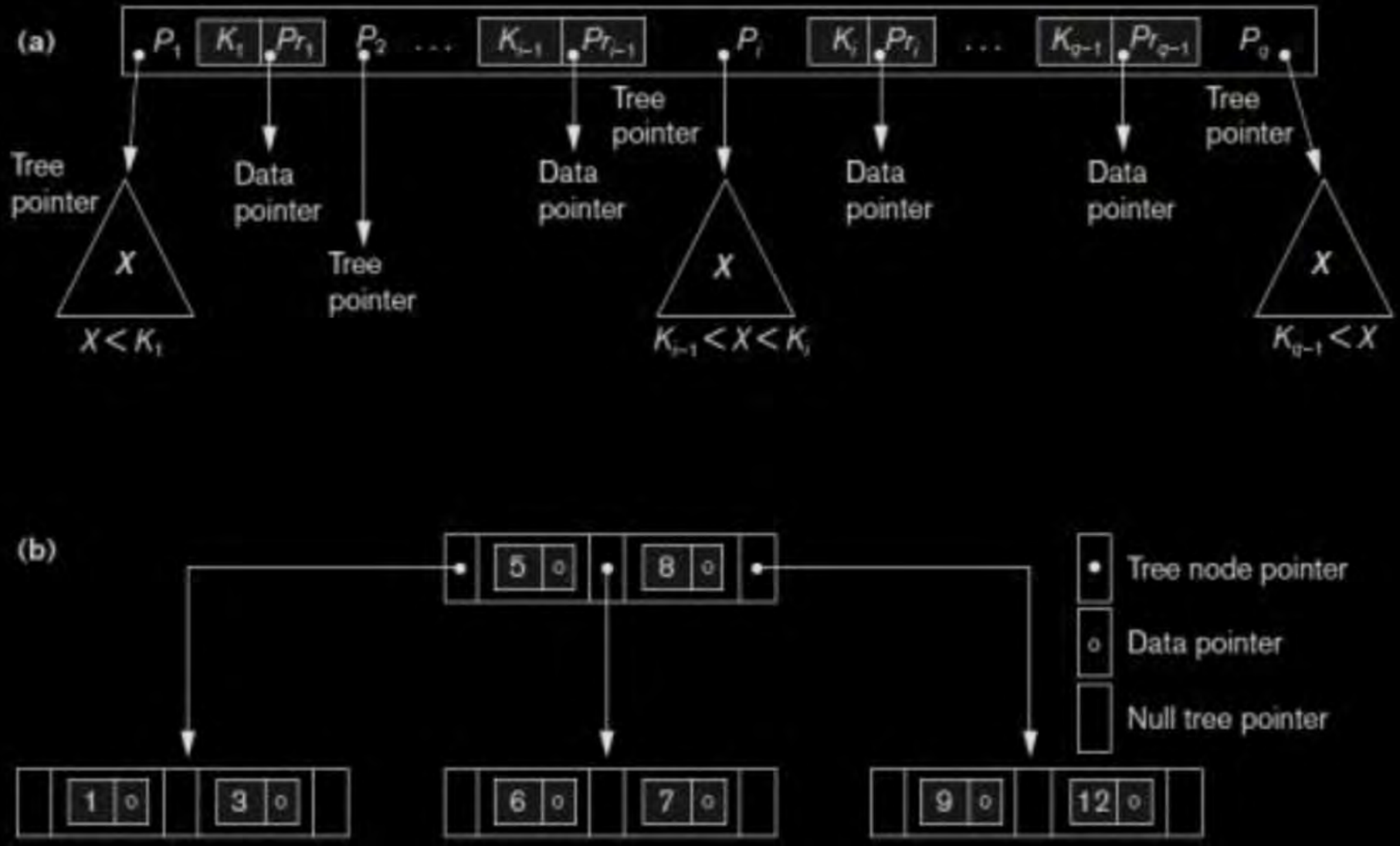
$$P * B_p + (P-1)(key + R_p) \leq \text{Block Size}.$$

B Tree Definition



B Tree Structure

B-tree structures (a) A node in a B-tree with q−1 search values (b) A B-tree of order p=3. The values were inserted in the order 8, 5, 1, 7, 3, 12, 9, 6

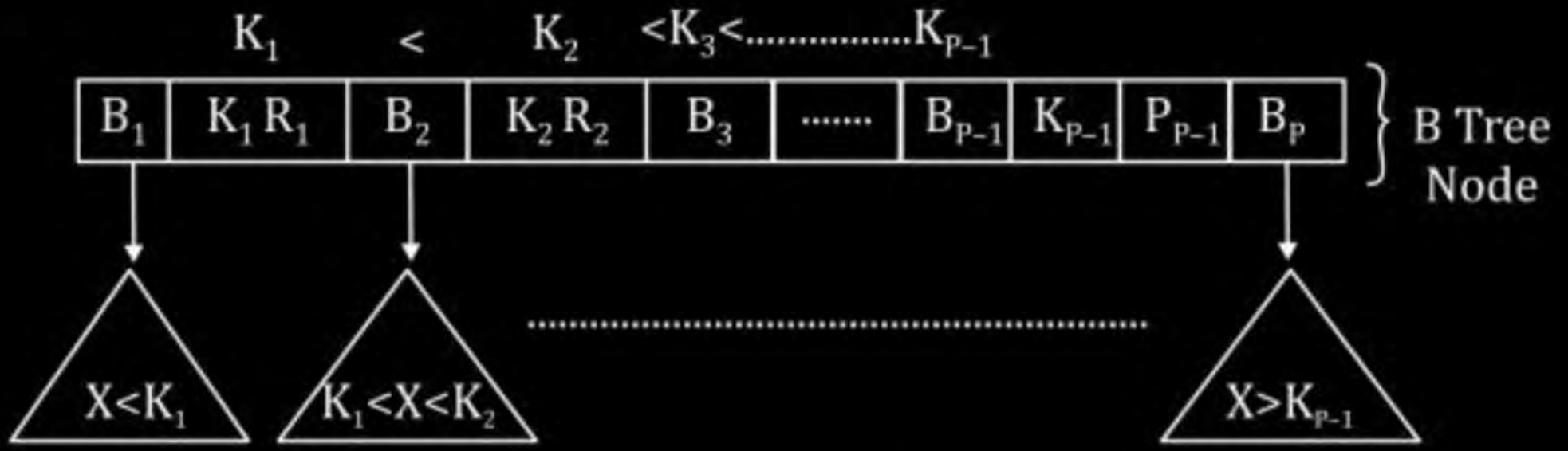


B Tree Definition

Order P: Max possible child pointers

[degree] can store in B Tree node.

(1) Node structure:

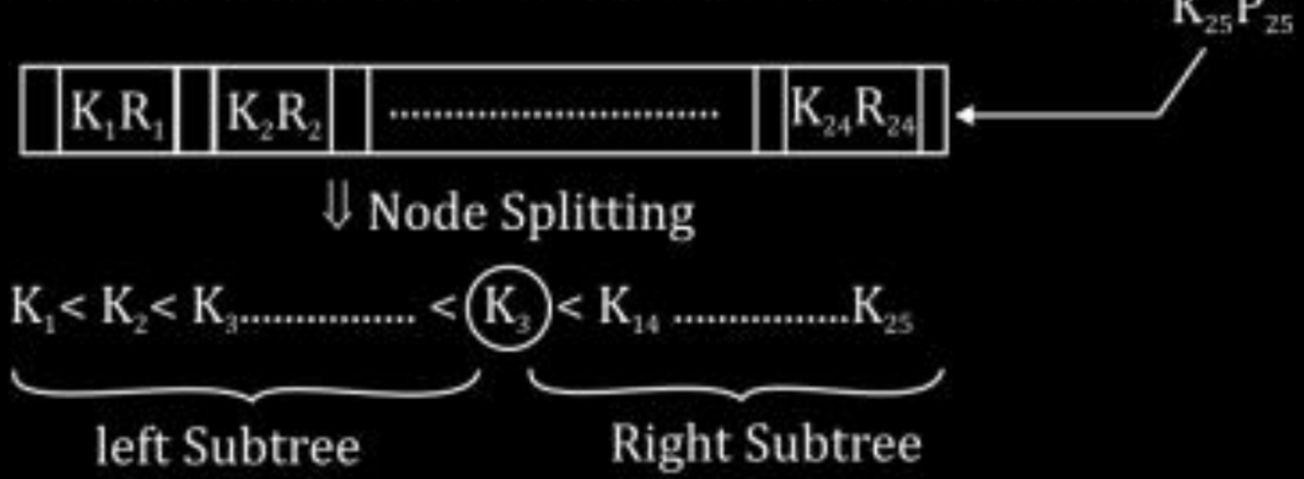


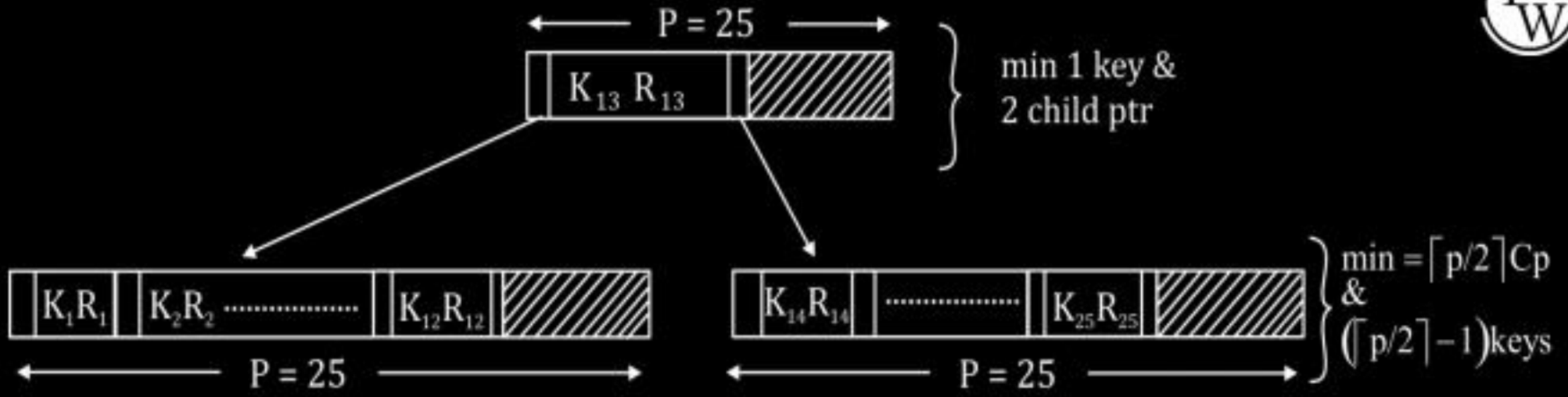
<P.Cp, (p-1)keys, (p-1)Rp>

(2) Every internal node except root must be at least $\lceil P/2 \rceil$ child pointer with $(\lceil P/2 \rceil - 1)$ key's and at most P children pointer with (P - 1) key's must.

(3) Root node can be at least 2 children with 1 key, at most P children pointer and (P - 1) key's.

(4) Every leaf node must be at same level and keys with in node should be sequence order (Increasing order) Order P: 25 [Max Cp in node]





except Root Every
index node
atleast 50% occupied
in B/B* Tree index

Block B tree node
[Free]

Q.3 Consider a table T in a relational database with a key field K. A B-tree of order p is used as an access structure on K, where p denotes the maximum number of tree pointers in B-tree index node. Assume that K is 10 bytes long; disk block size is 512 bytes; each data pointer P_D is 8 bytes long and each block pointer P_B is 5 bytes long. In order for each B-tree node to fit in a single disk block, the maximum value of p is [GATE-2004 : 2 Marks]

A 20

B 22

C 23

D 32

Advantage:

1) B Tree index best suitable for Random access of some record.

```
select *
FROM R
WHERE A = 24 ;
```

One record access

I/O cost: K + 1 blocks

$\approx [\log_p n] + 1 = \theta(\log_p n)$

Disadvantage:

1) B Tree index not best suitable for sequency access of range of records.

```
select *
FROM R
WHERE A ≥ 30 and A ≤ 85;
```

Range of record Access required

← X blocks of DB →

I/O cost: $x[\log_p n + 1]$ + cost of unsuccessful

More Access cost

[Unordered File DB]

Till Now B Tree Done.

B+ Tree Definition

Difference b/w B & B+ Tree

In B+ Tree

1. ALL keys are available at Leaf Node.
2. Internal Node: No Read Pointer.
3. Leaf Node Contain key & Rp + 1 (one) Block Pointer.

Difference b/w B Tree & B+ Tree.

① B Tree Internal Node

ORDER : P (Maximum # Block Pointers)

$B_p = P$

keys = P-1

$R_p = P-1$

$$P * B_p + (P-1)[key + R_p] \leq Block\ size$$

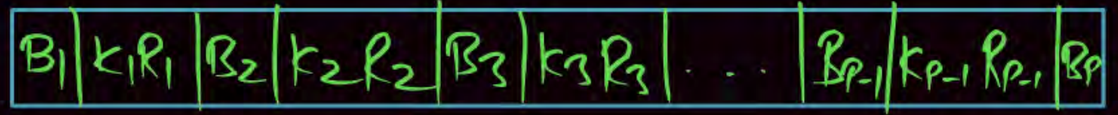
① B+ Tree Internal Node Structure.

ORDER : P ⇒ (Maximum # Pointers)

$$P * B_p + (P-1)\,key \leq Block\ size.$$

Difference b/w B Tree & B+ Tree.

(2) B Tree (Leaf) Node



ORDER : P (Maximum # Block Pointers)

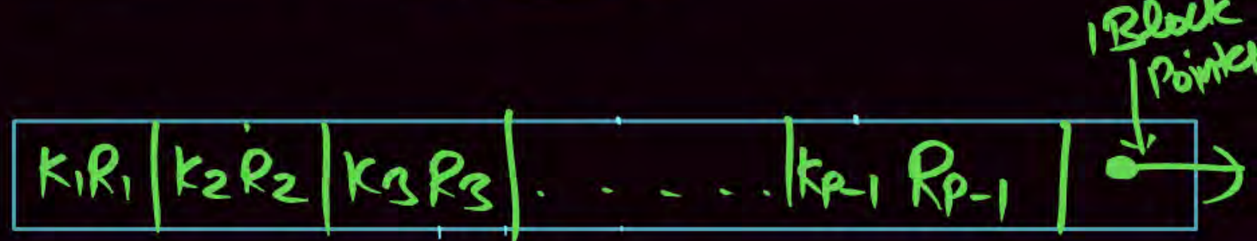
$B_p = P$

keys $= P-1$

$R_p = P-1$

$$P * B_p + (P-1)[key + R_p] \leq Block\ size$$

B+ Tree (Leaf.) Node Structure.



ORDER : P ⇒ (Maximum # Pointers)

$$(P-1)[keys + R_p] + 1B_p \leq Block\ Size$$

NOTE In a B Tree Internal Node & Leaf Node have Same Structure. (order of Internal Node is Same has Order of Leaf Node in B Tree)

NOTE In a B+ Tree Internal Node & Leaf Node have Different Structure. (order of Internal Node is NOT SAME as order of Leaf Node.)

Eg (1) Block size = 1024 Byte, Key = 6 Byte B_p = 8B R_p = 10 Byte

Order of Internal Node & Leaf Node in a B Tree ?

Soln

B Tree

$$P * B_p + (P-1)(key + R_p) \leq Block\ Size.$$

$$P*8 + (P-1)[6+10] \leq 1024$$

$$8P + 16P - 16 \leq 1024$$

$$24P \leq 1040$$

$$P = \left\lfloor \frac{1040}{24} \right\rfloor = \lfloor 43.3 \rfloor = 43 \text{ Ans}$$

ORDER

Internal Node : 43 Ans

Leaf Node : 43 Ans

Ans

(39) Consider.
Block Size = 1024 Byte, Ikey = 6 Byte B_p = 8B R_p = 10 Byte

Order of Internal Node & Leaf Node in a B+ Tree.

B+ Tree ORDER OF Internal Node.

$$P * B_p + (P-1)key \leq Block\ Size$$

$P*8 + (P-1)6 \leq 1024$

$8P + 6P - 6 \leq 1024$

$14P \leq 1030$

$P = \left\lfloor \frac{1030}{14} \right\rfloor = \lfloor 73.57 \rfloor = 73$ Ans

ORDER of Leaf Node in B+ Tree.

$$(P-1)[keys + R_p] + 1B_p \leq Block\ Size.$$

$(P-1)[6+10] + 8 \leq 1024$

$16P - 16 + 8 \leq 1024$

$16P \leq 1032$

$P = \left\lfloor \frac{1032}{16} \right\rfloor = \lfloor 64.5 \rfloor = 64$ Ans

B+ Tree

Non Leaf / Internal Node Order = 73

Leaf Node order = 64

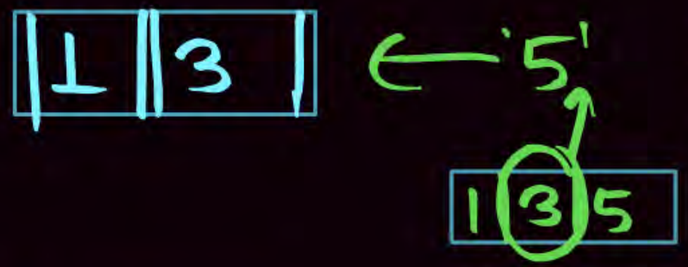
} Ans

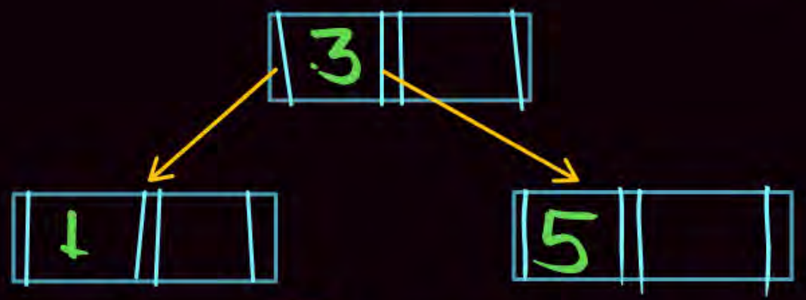
Difference b/w B Tree & B+ Tree.

key: 1, 3, 5

B Tree

ORDER : 3 Max key = 2





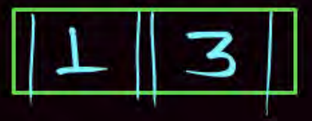
B Tree

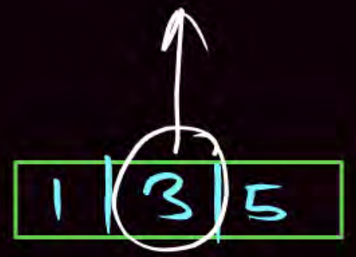
ORDER : 3

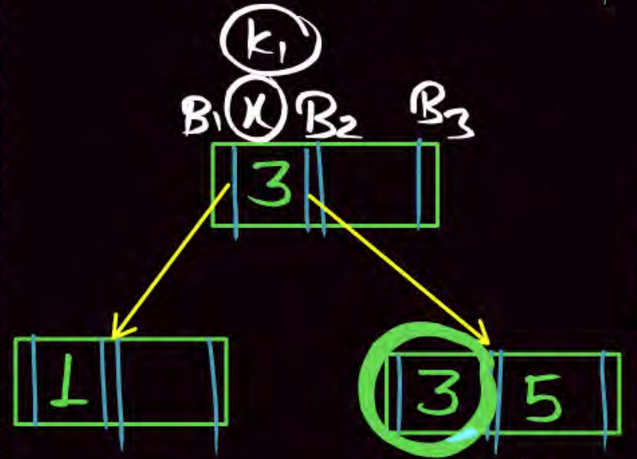
ORDER : 3
1, 3, 5.

B+ Tree

Max keys = 2

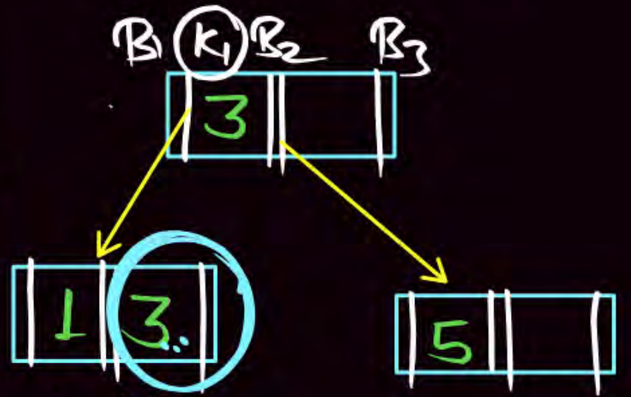






Right Biasing

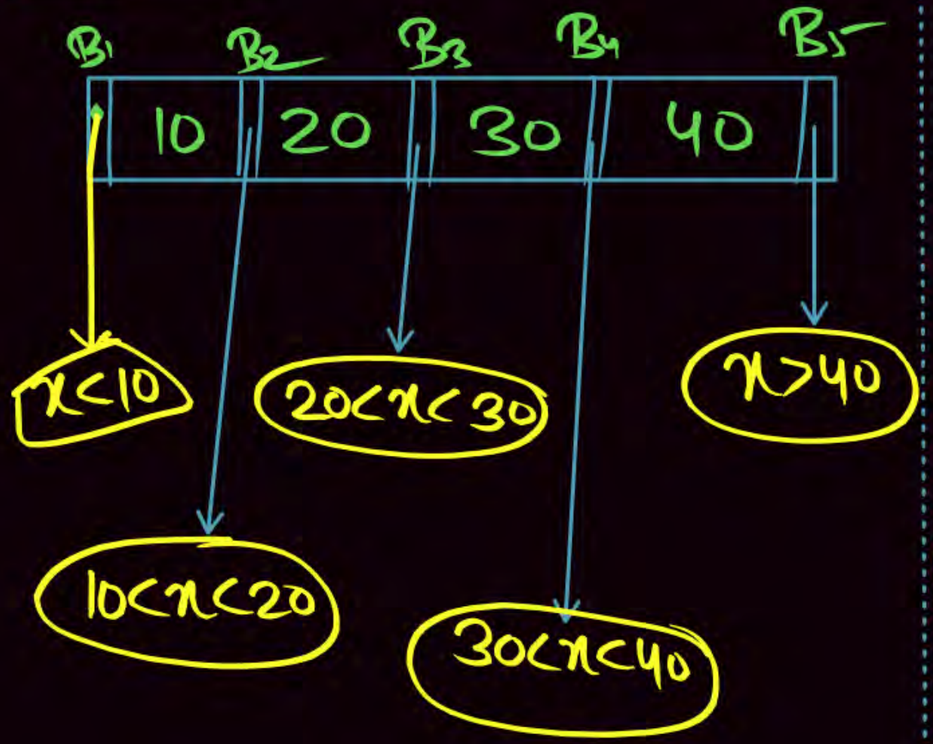
OR



Left Biasing.

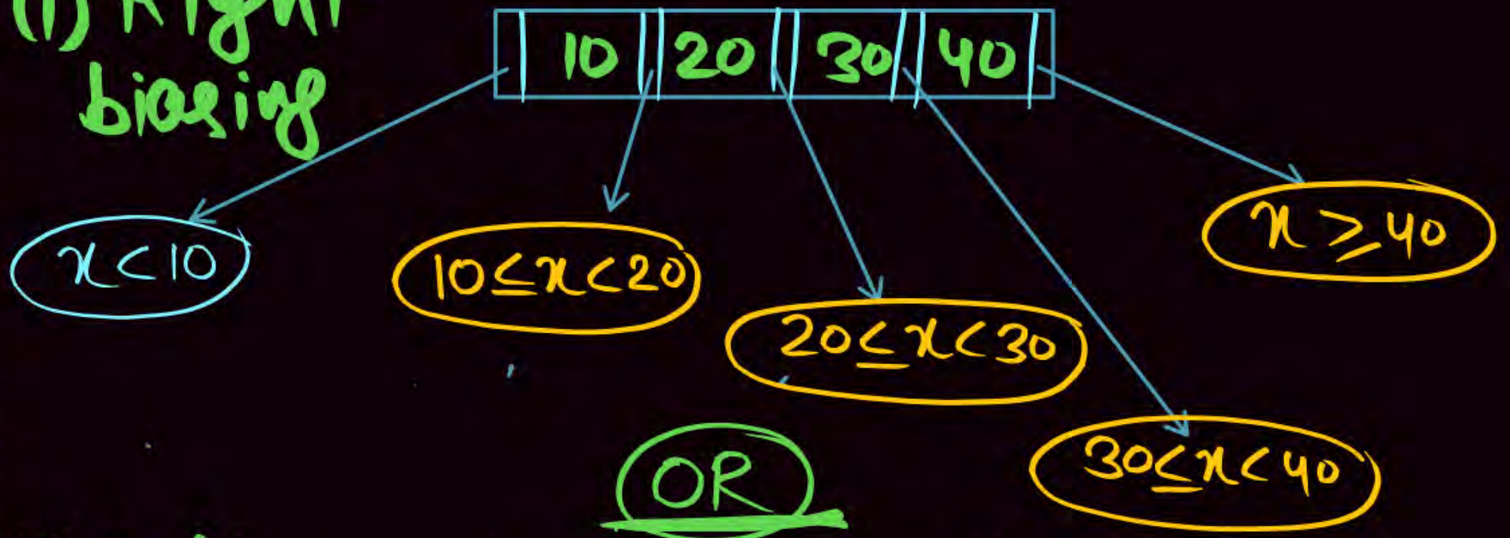
Difference b/w B Tree & B+ Tree.

B Tree ORDER: '5'



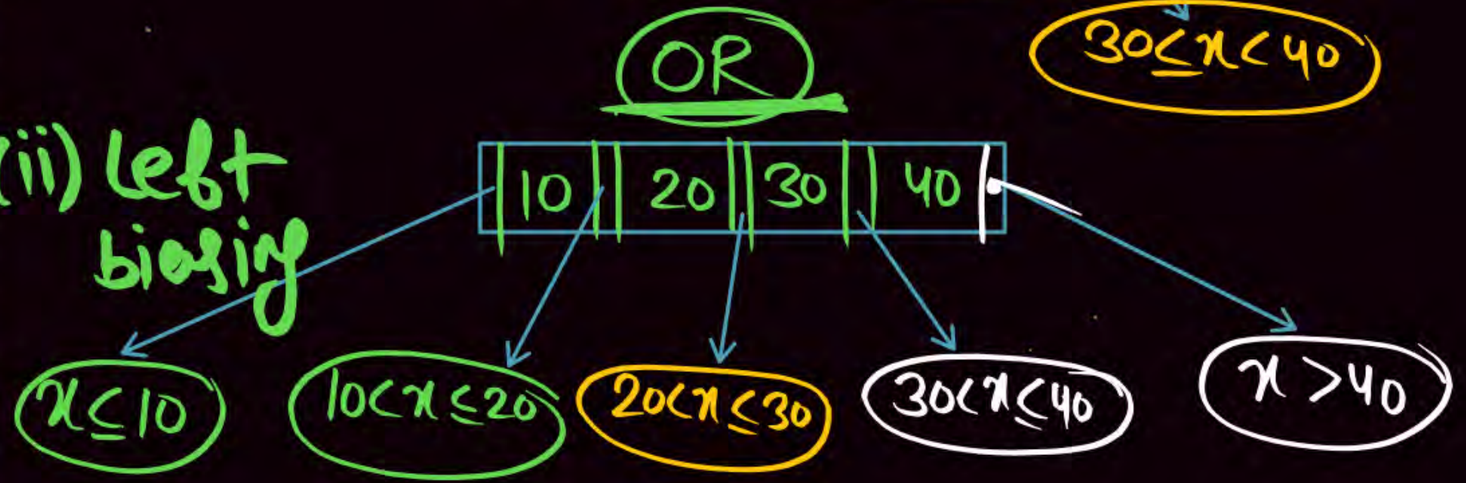
B+ Tree. ORDER: 5.

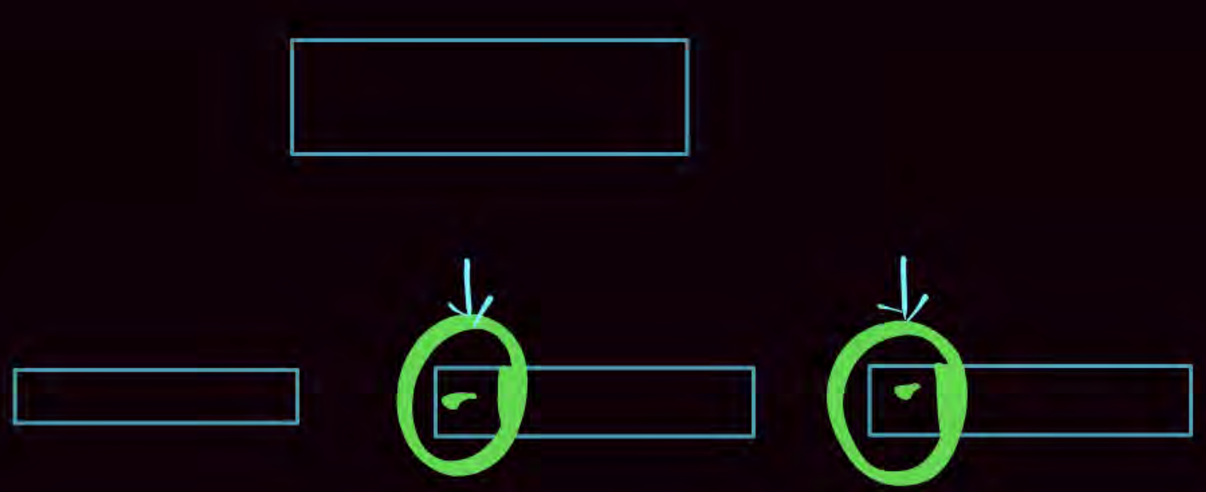
(i) Right biasing



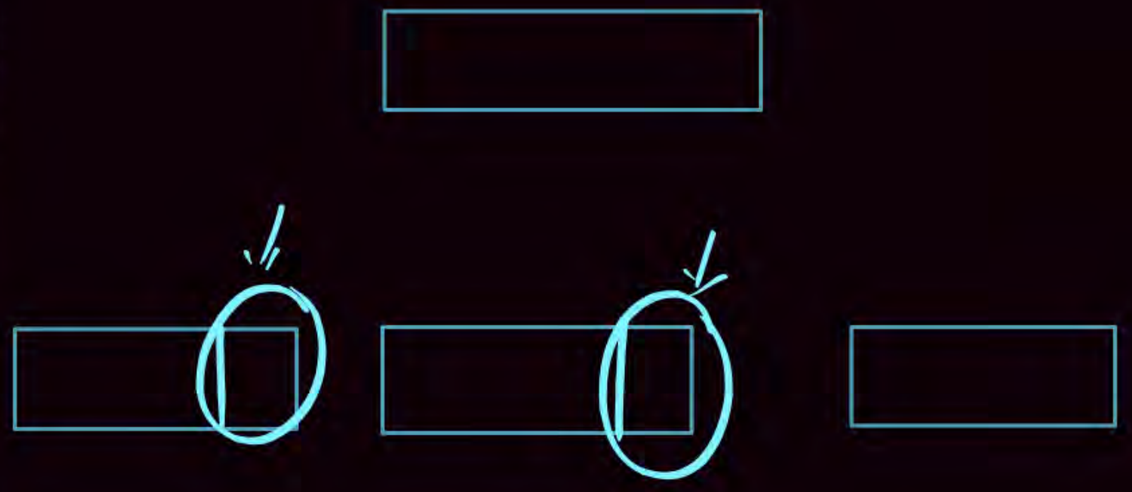
OR

(ii) Left biasing





Right Biasing



Left Biasing.

B+ Tree Definition

① Structure of Internal Node.

Left biasing

OR

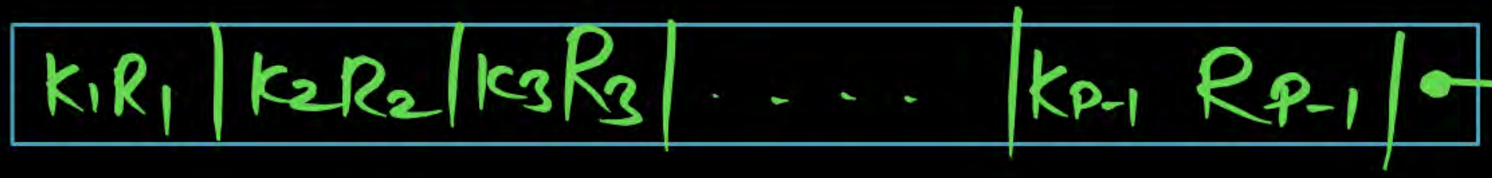
Right biasing

$P * B_p + (P-1)\,key \leq Block\ Size$

B^+ Tree Definition

② Structure of Leaf Node :

$$(P-1)[key + R_P] + LBP \leq \text{Block Size}.$$



③ Internal Node → Internal Node (Min & Max)

④ Same as Root → Root Node (Min & Max)

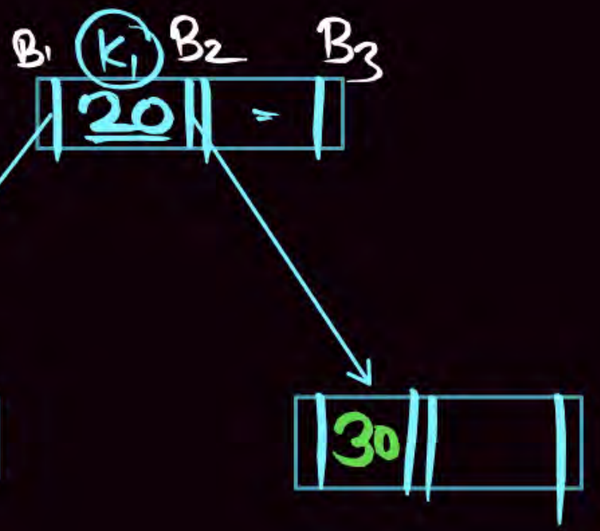
⑤ B Tree. → keys with in the Node Assending.

⑥ → All Leaf Node Same Level

B+ Tree : Left & Right biasing

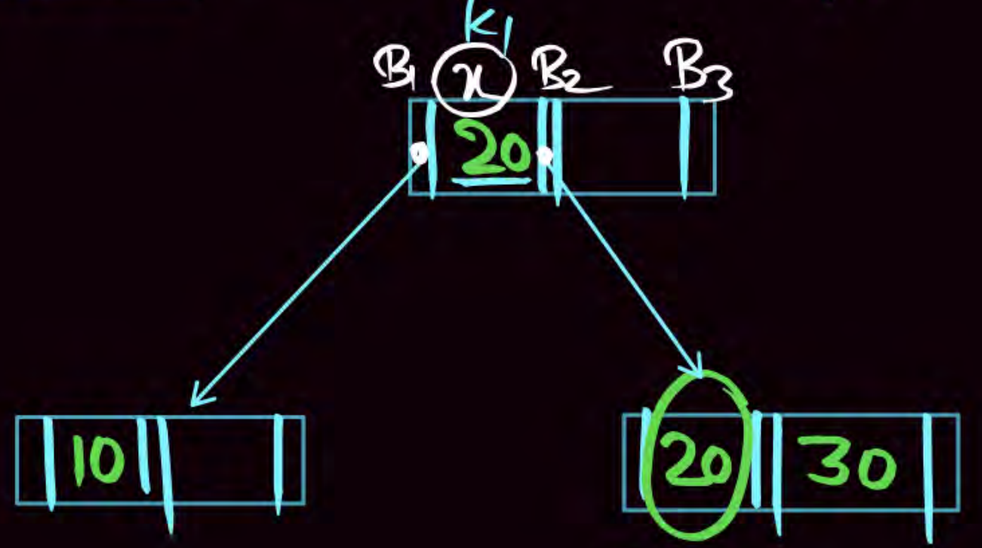
ORDER : 3 :: 10, 20, 30

Max. key = 2.



Left biasing

'30'



Right Biasing

Eg (1) Block size = 1024 Byte, Key = 6 Byte B_p = 8B R_p = 10 Byte

Order of Internal Node & Leaf Node in a B Tree ?

Soln

B Tree

$$P * B_p + (P-1)(key + R_p) \leq Block\ Size.$$

$$P*8 + (P-1)[6+10] \leq 1024$$

$$8P + 16P - 16 \leq 1024$$

$$24P \leq 1040$$

$$P = \left\lfloor \frac{1040}{24} \right\rfloor = \lfloor 43.3 \rfloor = 43 \text{ Ans}$$

ORDER

Internal Node : 43 Ans

Leaf Node : 43 Ans

Ans

Consider.

(39) Block Size = 1024 Byte, Key = 6 Byte B_p = 8B R_p = 10 Byte

Order of Internal Node & Leaf Node in a B+ Tree.

B+ Tree ORDER OF Internal Node.

$$P * B_p + (P-1)\,key \leq Block\ Size$$

$P*8 + (P-1)6 \leq 1024$

$8P + 6P - 6 \leq 1024$

$14P \leq 1030$

$P = \left\lfloor \frac{1030}{14} \right\rfloor = \lfloor 73.57 \rfloor = 73$ Ans

ORDER of Leaf Node in B+ Tree.

$$(P-1)[keys + R_p] + 1B_p \leq Block\ Size.$$

$(P-1)[6+10] + 8 \leq 1024$

$16P - 16 + 8 \leq 1024$

$16P \leq 1032$

$P = \left\lfloor \frac{1032}{16} \right\rfloor = \lfloor 64.5 \rfloor = 64$ Ans

B+ Tree

NonLeaf / Internal Node Order = 73

Leaf Node Order = 64

} Ans

B+ Tree:

FAQ

Q.1 Why Internal Node Not having Rp ?

Q.2 Why Leaf Node has Only key & Rp Only ?

Q.3 Why Leaf Node having Only 1 Block Pointer ?

Q.4 When all keys are available at Leaf Node then Why some keys are available in Non Leaf Node ?

B+ Tree:

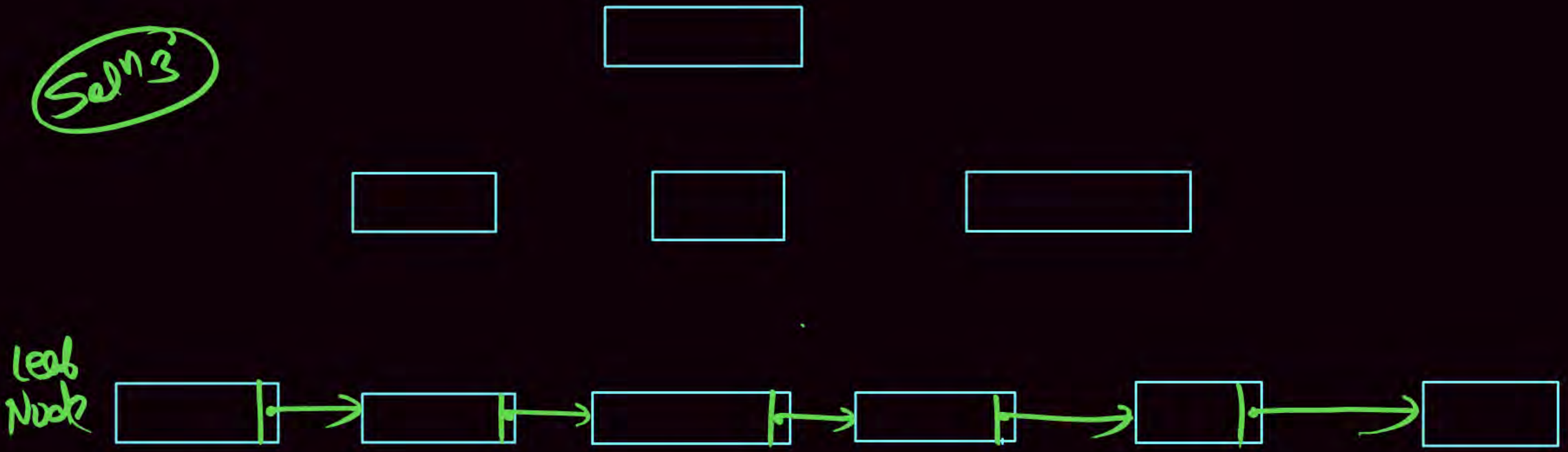
Soln 1 Bcz ALL keys are available at Leaf Node So Read Pointer Not Required in Internal Node. So Rp Wallah Space Utilized to Store More keys.

Soln 2 Bcz all the keys are available in Leaf Node According B+ Defination So To Read that keys (Data) Read | Record | Data Pointer is Required.

Soln 3 B+ Tree is Suitable for Range Query. & In B+ Tree All keys available in the leaf Node & if 1 Block Pointer Connect to Next Node at Leaf (like Link List) So IIo Cost Reduced (Range Query Perform)

B+ Tree.

Soln 3



Range Query Age 30 to 80

B+ Tree.

Sabry

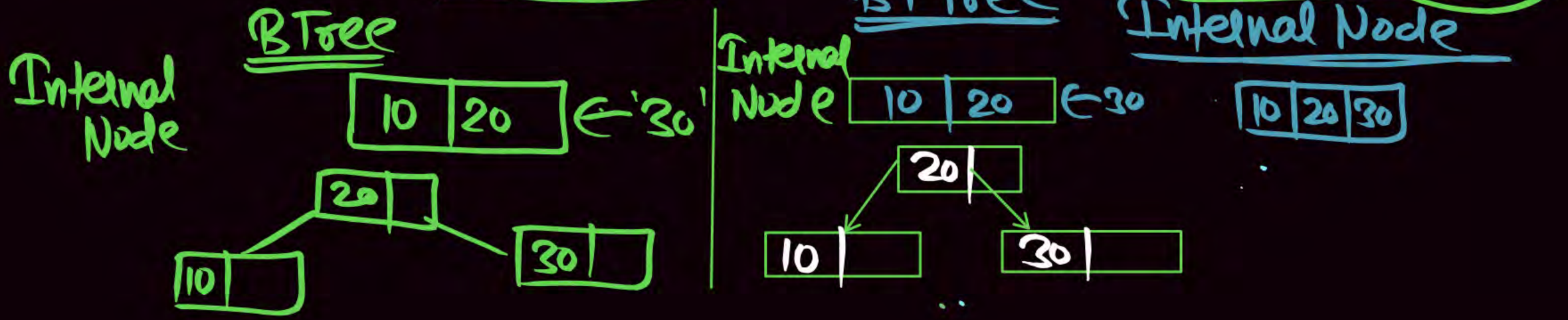
YES Some keys are available on Non Leaf. Node But they are very Useful in Searching Purpose. but that Use only for Searching Purpose. they Don't have Read(Record|Data) Pointer in Non Leaf Node.

B+ Tree. Creation | Insertion in B+ Tree.

keys : 10, 20, 30, 40, 50

① In B+ Tree Insertion start from Leaf Node.

② Split of Internal NODE is same as Split of Internal Node in B+ Tree.

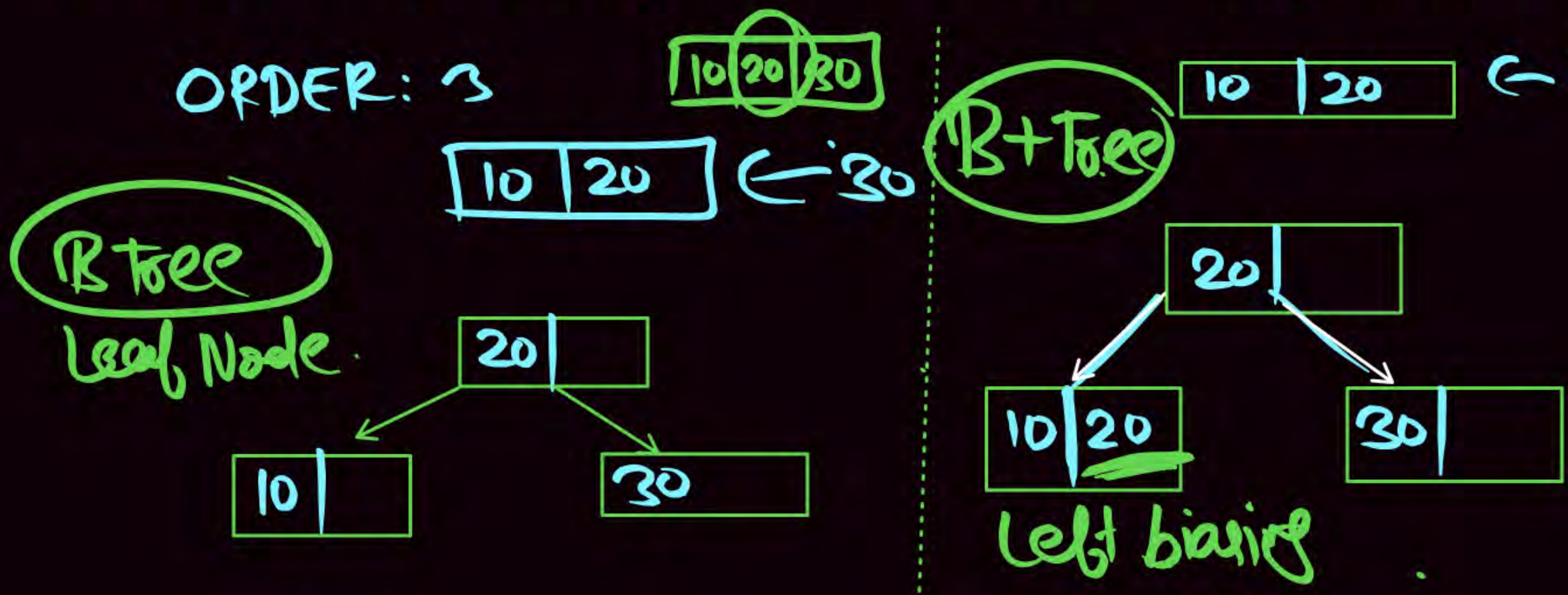


B+ Tree. Creation | Insertion in B+ Tree.

keys : 10, 20, 30, 40, 50.

③ Split of Leaf in B+ Tree Different from B Tree

ORDER: 3



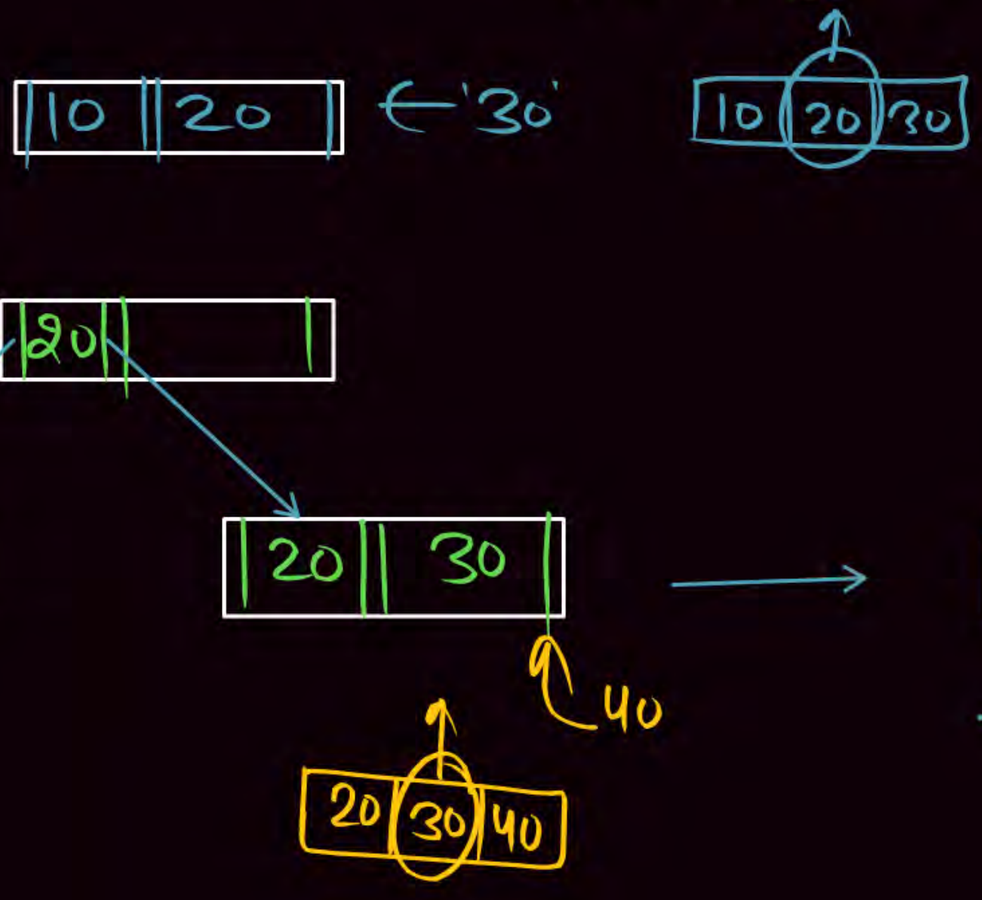
B+ Tree. Creation | Insertion in B+ Tree.

keys : 10, 20, 30, 40, 50

ORDER : 3

Max key = 2

Right biasing.



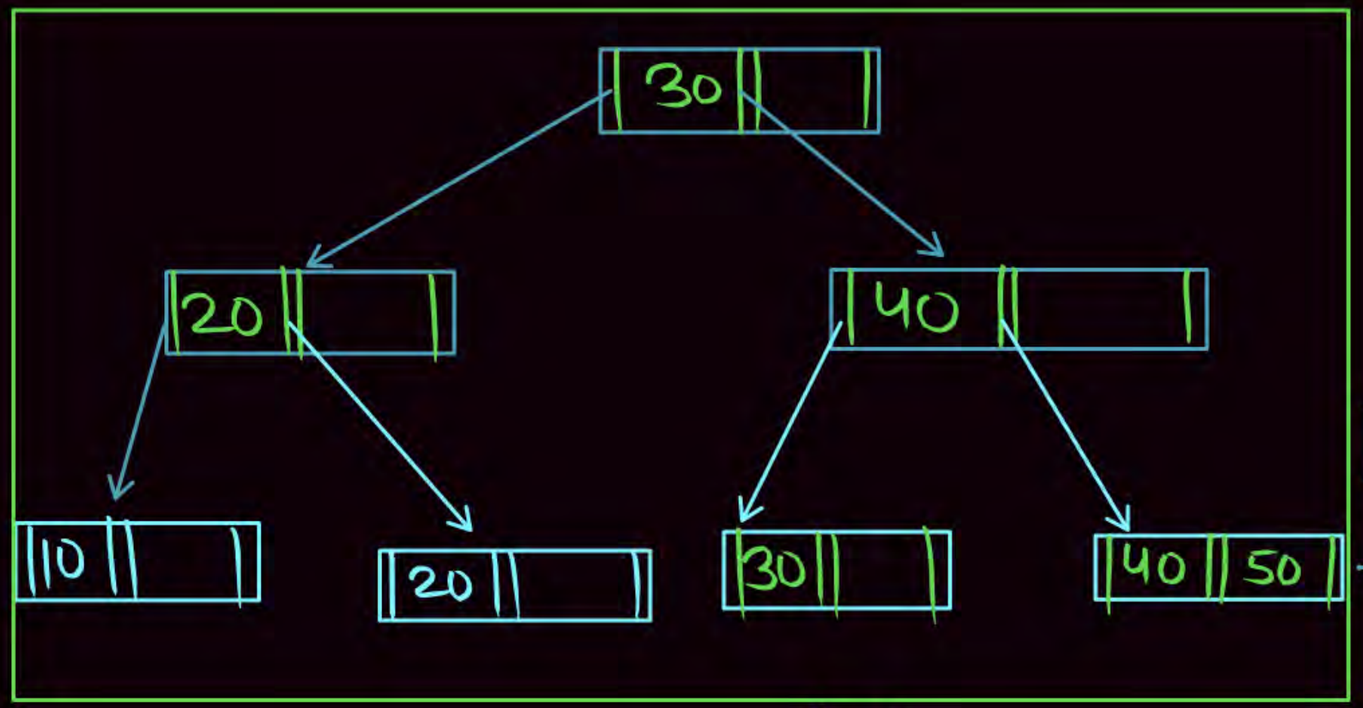
B+ Tree. Creation | Insertion in B+ Tree.

keys : 10, 20, 30, 40, 50

ORDER : 3

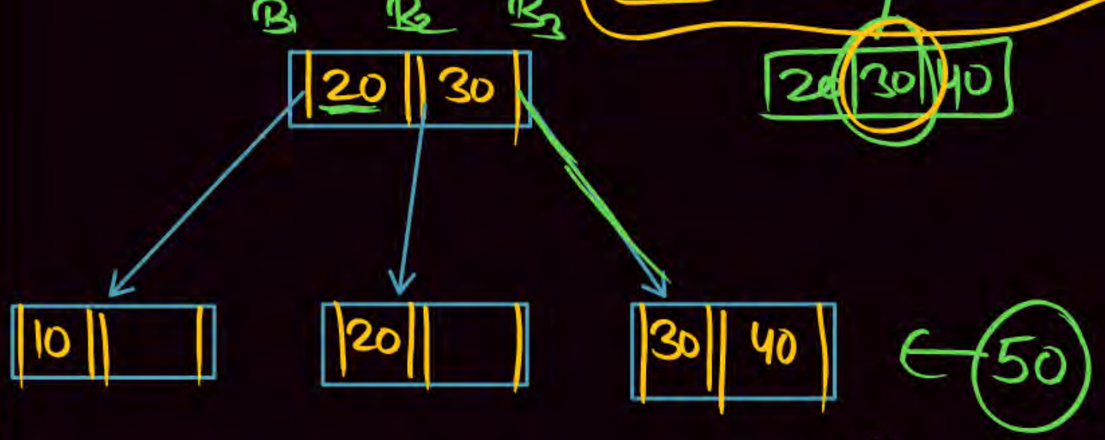
Max key = 2.

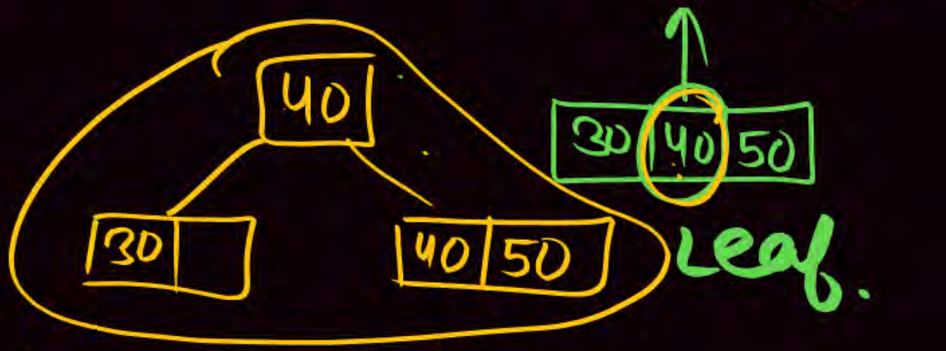
Right biasing.



Ans

Internal





Leaf.

Important Point

(e.g.) 10, 20, 30, 40, 50

ORDER : 3

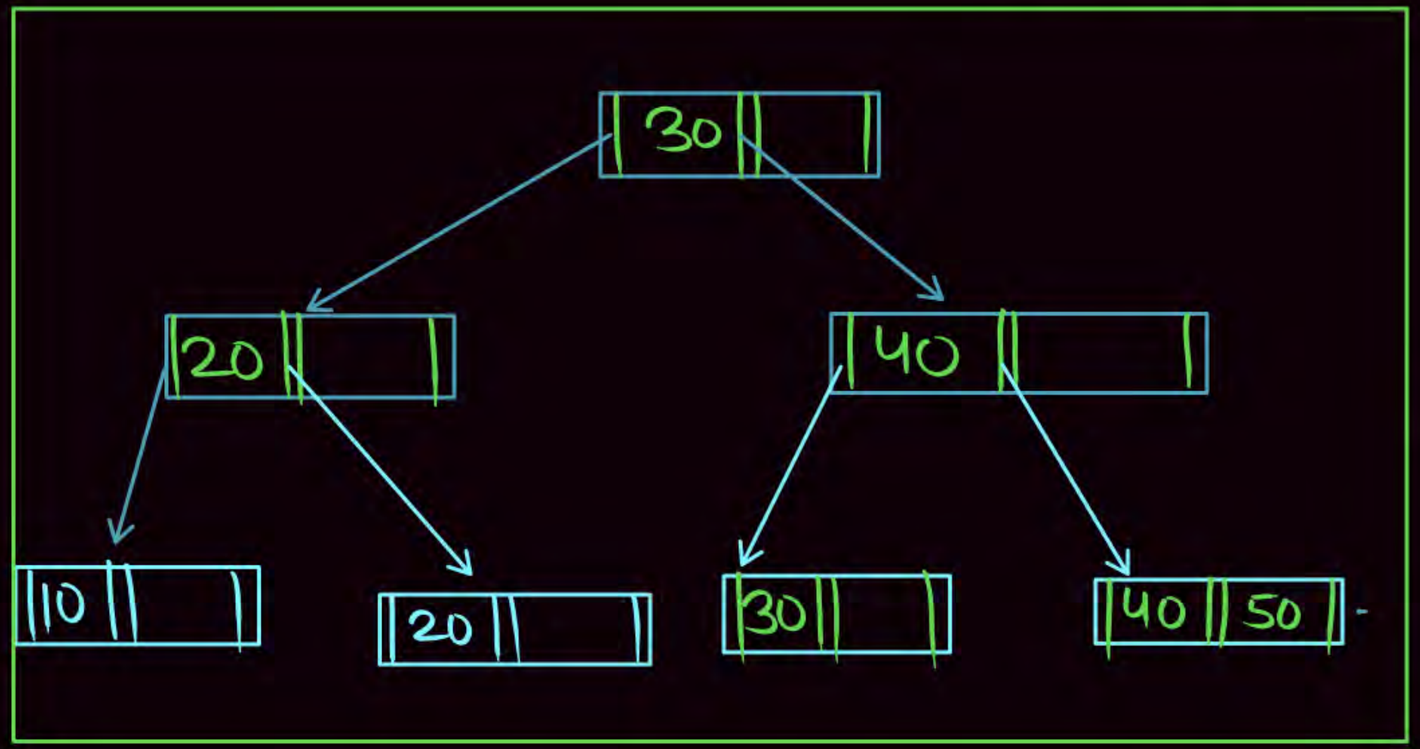
Draw B+ Tree

Points

B+ Tree. Creation | Insertion in B+ Tree.

Keys : 10, 20, 30, 40, 50 ORDER : 3 Right biasing.

Max key = 2.



Ans 10, 20, 30, 40, 50

Important Point

① Every key of Leaf Node is Not Present in Non Leaf Node.

@ keys 10, 20, 30, 40, 50 in Leaf Node But all keys are Not Present in Non Leaf Node (eg 10, 50 Not Present in Non Leaf in Last Example)

Important Point

② Some key of Leaf Node is present in Non Leaf Node.

③ But Every key of Non Leaf Node Must be Present in Leaf Node. (ie in B+ Tree All keys are available at the Leaf Node).

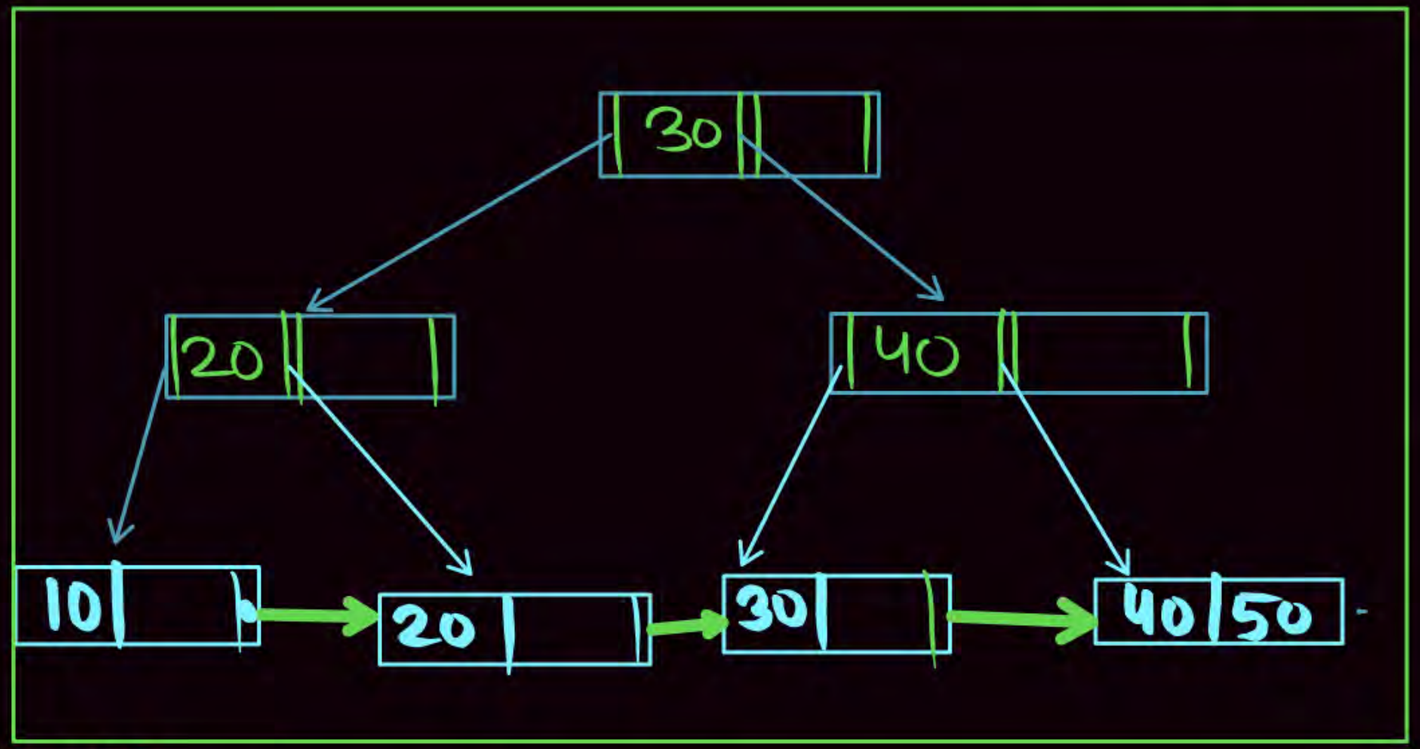
B+ Tree. Creation | Insertion in B+ Tree.

Keys : 10, 20, 30, 40, 50

ORDER : 3

Max key = 2.

Right biasing.



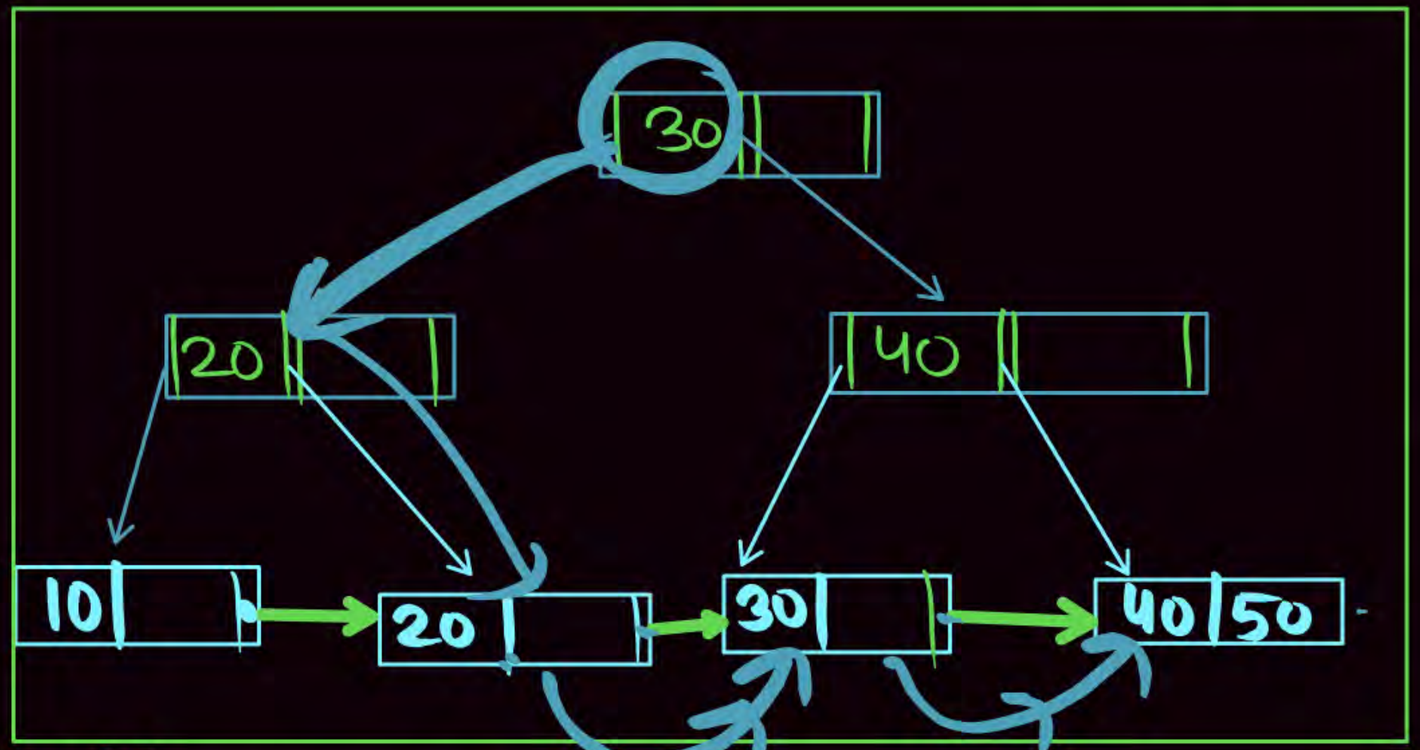
Ans 10, 20, 30, 40, 50

Range Query 20 to 40.

B+ Tree. Creation | Insertion in B+ Tree.

Keys : 10, 20, 30, 40, 50 ORDER : 3 Right biasing.

Max key = 2.



Range Query 20 to 40.

Avg 10, 20, 30, 40, 50

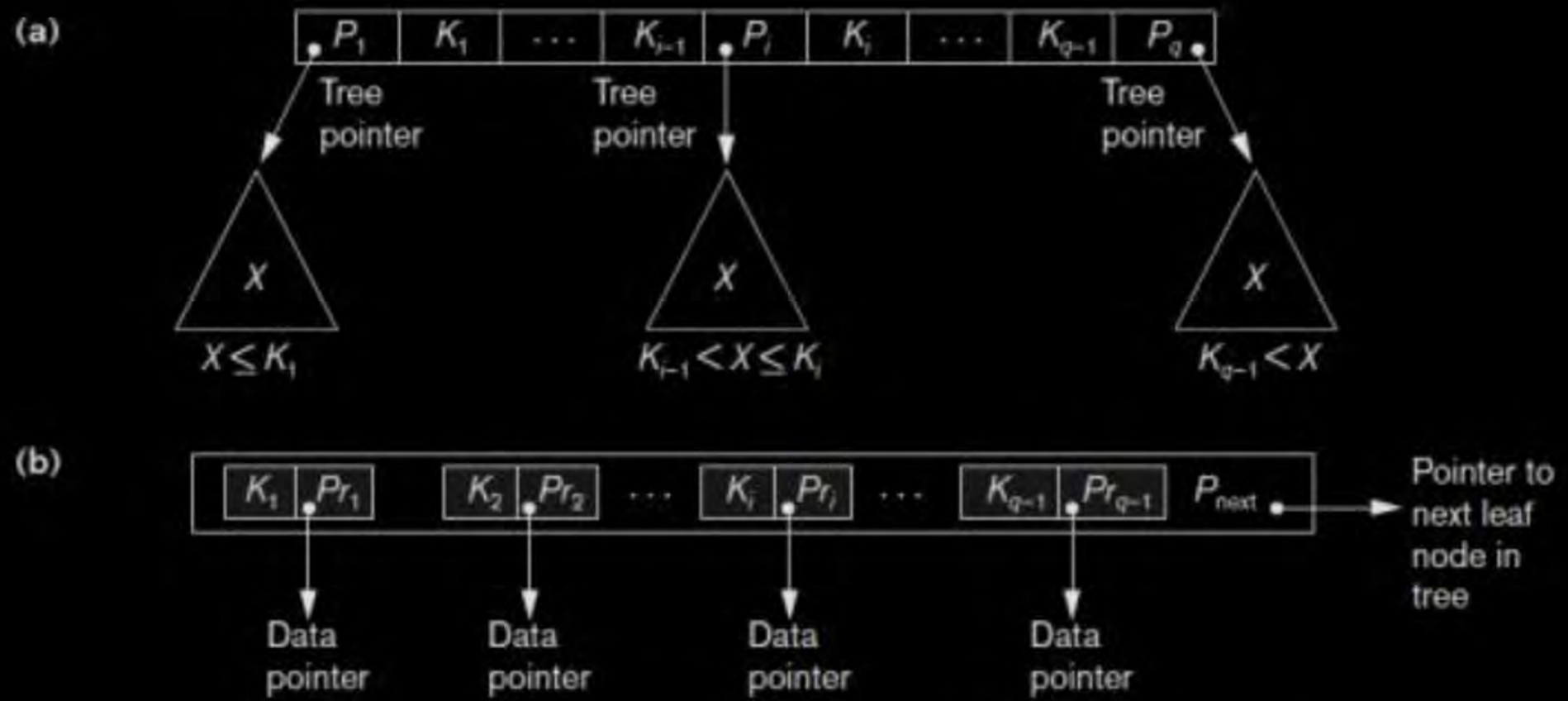
Important Point

Leaf NODE : Key Order

B+ Tree :

$$p(\text{key} + R_p) + \perp B_p \leq \text{Block Size}$$

B+ Tree



The nodes of a B+-tree (a) Internal node of a B+-tree with q−1 search values (b) Leaf node of a B+-tree with q−1 search values and q−1 data pointers

B+ Tree Definition

Order P: max possible pointers [degree] can store in B^+ Tree node.

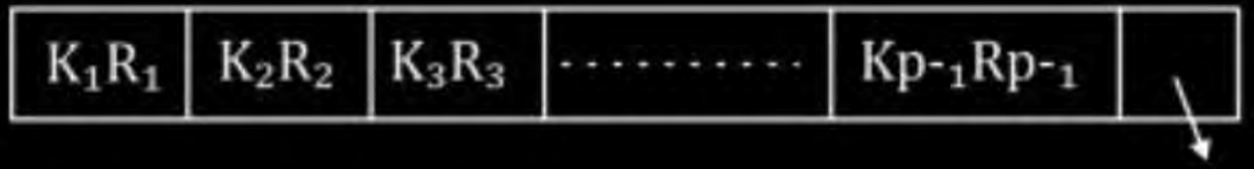
(1) Node Structure:

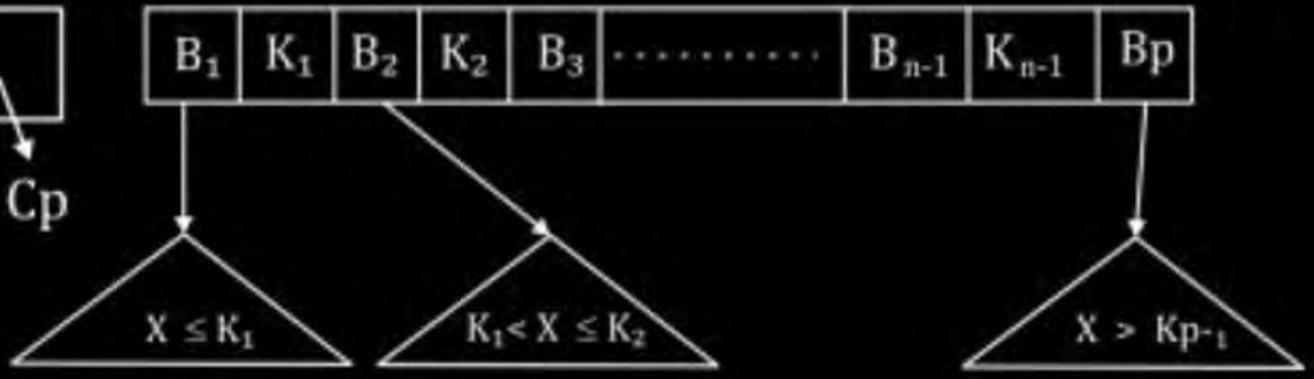
(a) Leaf Node

[set of (key, Rp) pair and only one block pointer pointed to next Leaf node]

(b) Internal Node

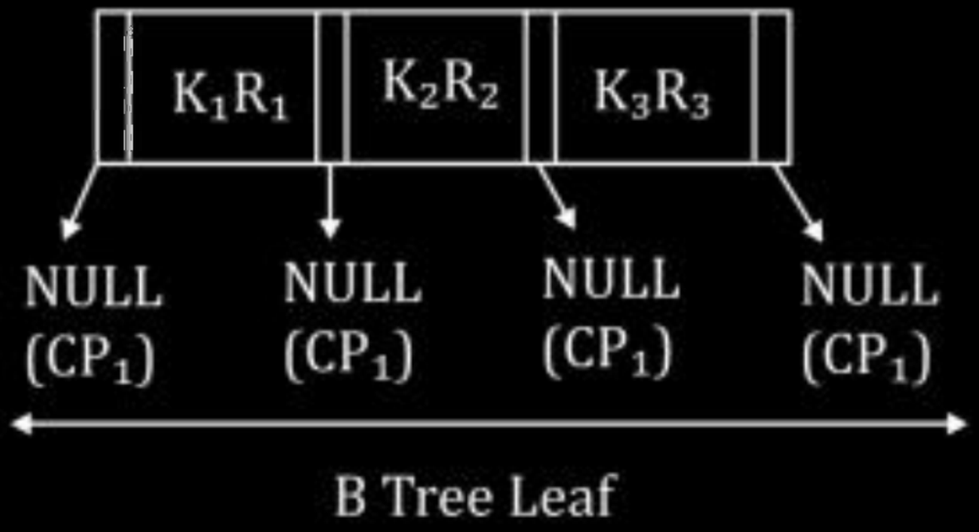
[key's and child pointer]
∴ No record pointer (Rp).

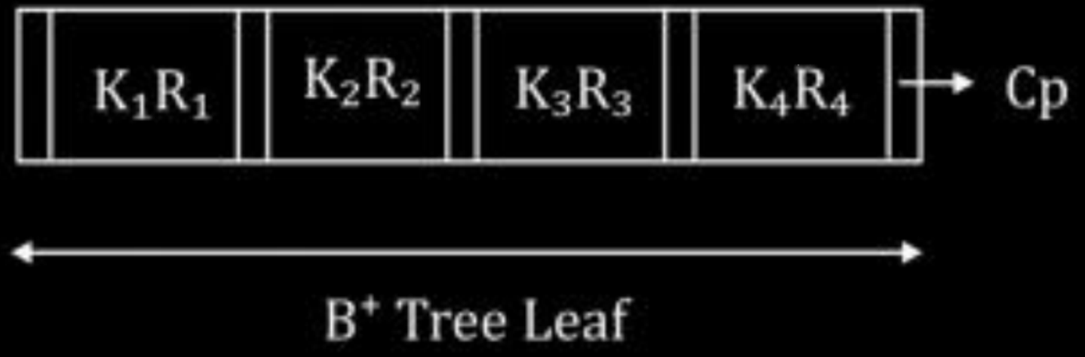




⇒ Left Biasing condition $k_1 < x \leq k_2$ .....

Right Biasing condition $k_1 \leq x < k_2$ ..... $x \geq k_{p-1}$





⇒ Remaining Balancing Condition same as B Tree

Advantage:

1) B^+ Tree index best suitable for sequence access of range of records.

   [Range Quires runs faster using B^+ Tree index]

   Select *

   FROM R

   WHERE A ≥ 30 and A ≤ 85;

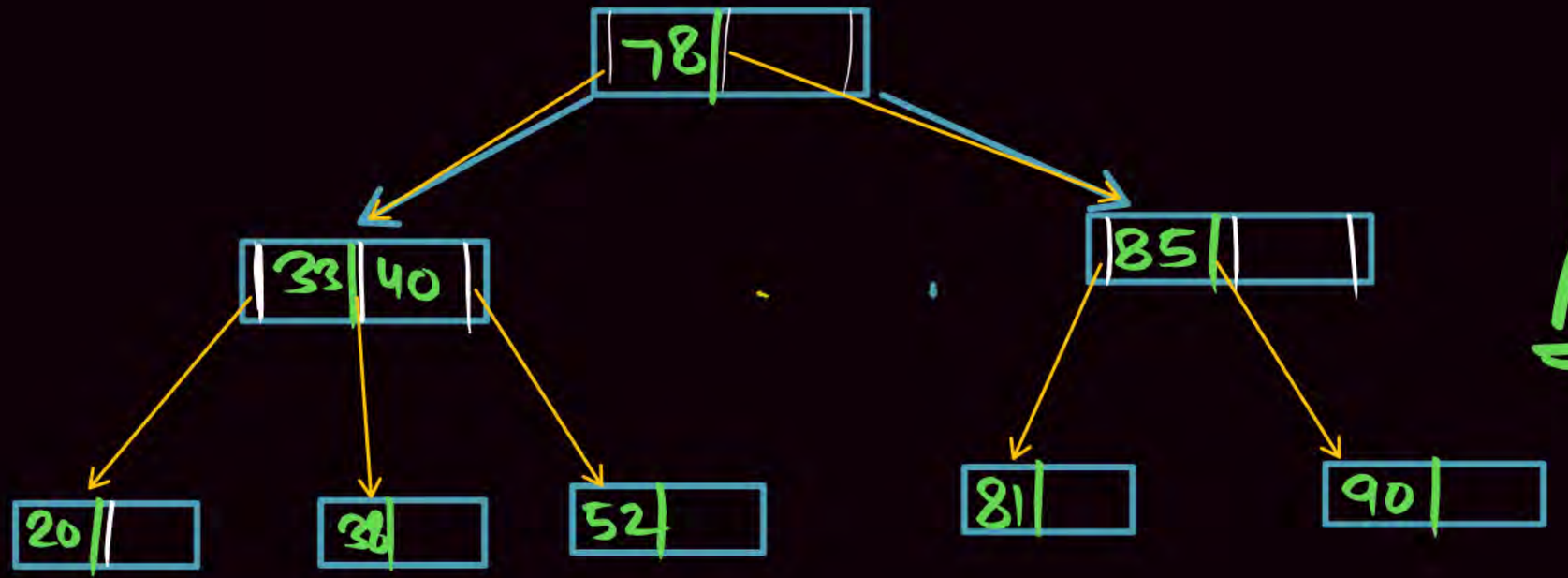
   I/O cost: $[\log_p n + x + x]$

   For Index

   For Database File

78, 52, 81, 40, 33, 90, 85, 20, 38.

B Tree : ORDER : 3.



Ans

Home Work Question.

Q.4

The order of an internal node in a B^+tree index is the maximum number of children it can have. Suppose that a child pointer takes 6 bytes, the search field value takes 14 bytes, and the block size is 512 bytes. What is the order of the internal node?

A 24

B 25

C 26

D 27

MCQ 2

Consider a B^+ tree in which the maximum number of keys in a node is 5 .
What is the minimum number of keys in any non-root node?

[GATE-2010-CS: 1M]

A 1

B 2

C 3

D 4

MCQ

3

A file is organized so that the ordering of data records is the same as or close to the ordering of data entries in some index. Then that index is called.

[GATE-2015-CS: 1M]

(A) Dense

(B) Sparse

(C) Clustered

(D) Unclustered

MCQ 4

Consider a file of 16384 records. Each record is 32 bytes long and its key field is of size 6 bytes. The file is ordered on a non-key field, and the file organization is unspanned. The file is stored in a file system with block size 1024 bytes, and the size of a block pointer is 10 bytes. If the secondary index is built on the key field of the file, and a multilevel index scheme is used to store the secondary index, the number of first-level and second-level blocks in the multilevel index are respectively-

[GATE-2008-CS: 2M]

A 8 and 0

B 128 and 6

C 256 and 4

D 512 and 5

MCQ 5

A clustering index is defined on the fields which are of type

[GATE-2008-CS: 1M]

A non-key and ordering

B non-key and non-ordering

C key and ordering

D key and non-ordering

NAT

6.

In a B+ tree, if the search-key value is 8 bytes long, the block size is 512 bytes and the block pointer size is 2 bytes, then the maximum order of the (Max. Child pointer or internal Node) B+ tree is _____

[GATE-2017-CS: 2M]

NAT

7.

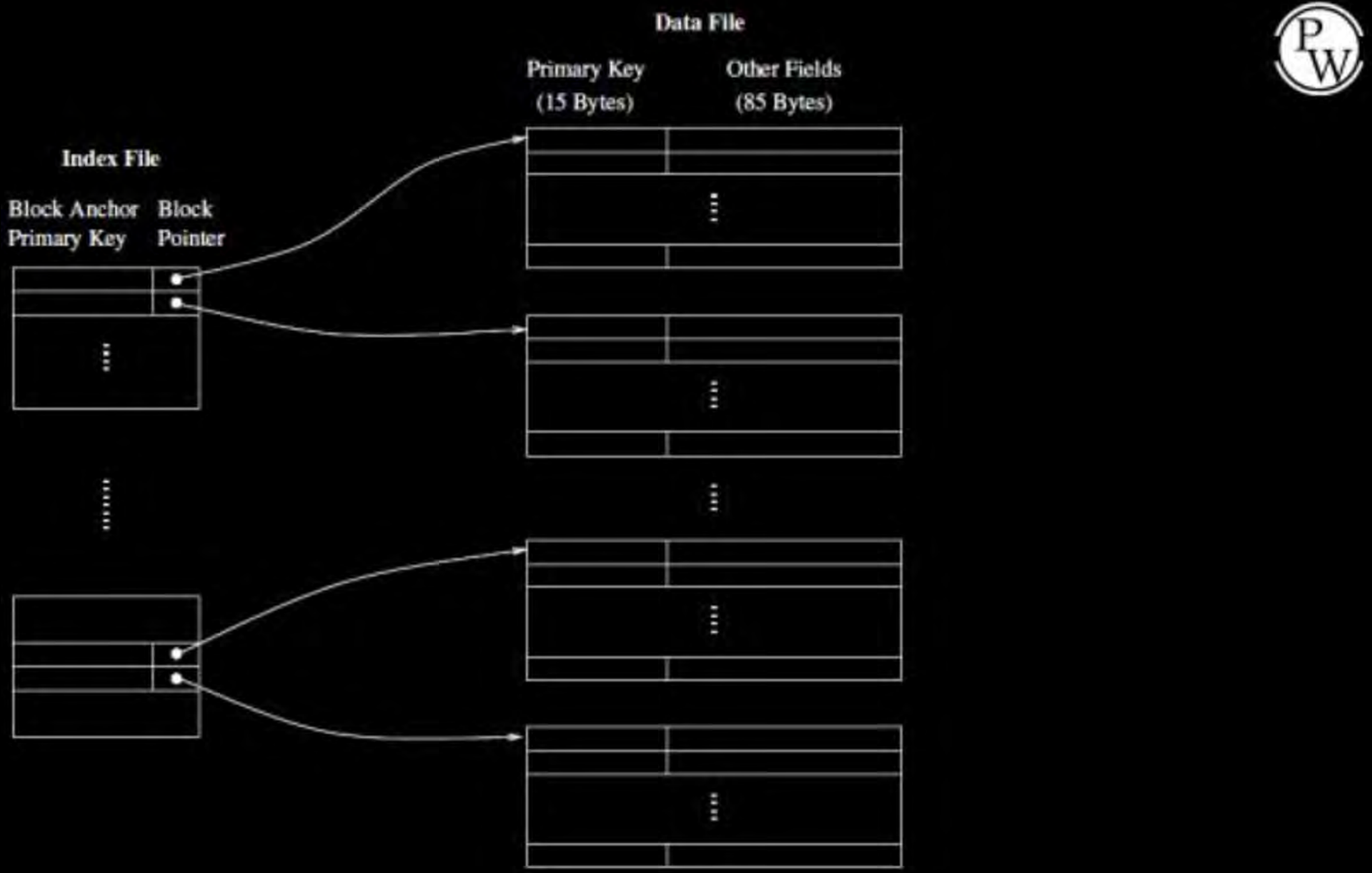
Consider a B^+ tree in which the search key is 12 bytes long, block size is 1024 bytes, record pointer is 10 bytes long and block pointer is 8 bytes long. The maximum number of keys that can be accommodated in each non-leaf node of the tree is _____

[GATE-2015-CS: 2M]

MCQ 8

Consider a database of fixed-length records, stored as an ordered file. The database has 25,000 records, with each record being 100 bytes, of which the primary key occupies 15 bytes. The data file is block-aligned in that each data record is fully contained within a block. The database is indexed by a primary index file, which is also stored as a block-aligned ordered file. The figure below depicts this indexing scheme.

[GATE-2023-CS: 2M]


Data File
Primary Key
(15 Bytes)
Other Fields
(85 Bytes)
Index File
Block Anchor
Primary Key
Block
Pointer

Suppose the block size of the filo system is 1024 bytes, and a pointer to a block occupies 5 bytes. The system uses binary search on the index file to search for a record with a given key. You may assume that a binary search on an index file of b blocks takes $[\log_2 b]$ block accesses in the worst case.
Given a key, the number of block accesses required to identify the block in the data file that may contain a record with the key. in the worst case, is_____.

MCQ 9.

The following key values are inserted into a B+ -tree in which order of the internal nodes is 3, and that of the leaf nodes is 2, in the sequence given below. The order of internal nodes is the maximum number of tree pointers in each node, and the order of leaf nodes is the maximum number of data items that can be stored in it. The B+ -tree is initially empty.

10, 3, 6, 8, 4, 2, 1

The maximum number of times leaf nodes would get split up as a result of these insertions is

[GATE-2009-CS: 2M]

A 2

B 3

C 4

D 5

MCQ

10 1,2,3,4.

A B-tree of order 4 is built from scratch by 10 successive insertions. What is the maximum number of node splitting operations that may take place?

[GATE-2008-CS: 2M]

A 3

B 4

C 5

D 6

MCQ 4

Which one of the options given below refers to the degree (or arity) of a relation in relational database systems?

[GATE-2023-CS: 1M]

A. Number of attributes of its relation schema.

B. Number of tuples stored in the relation.

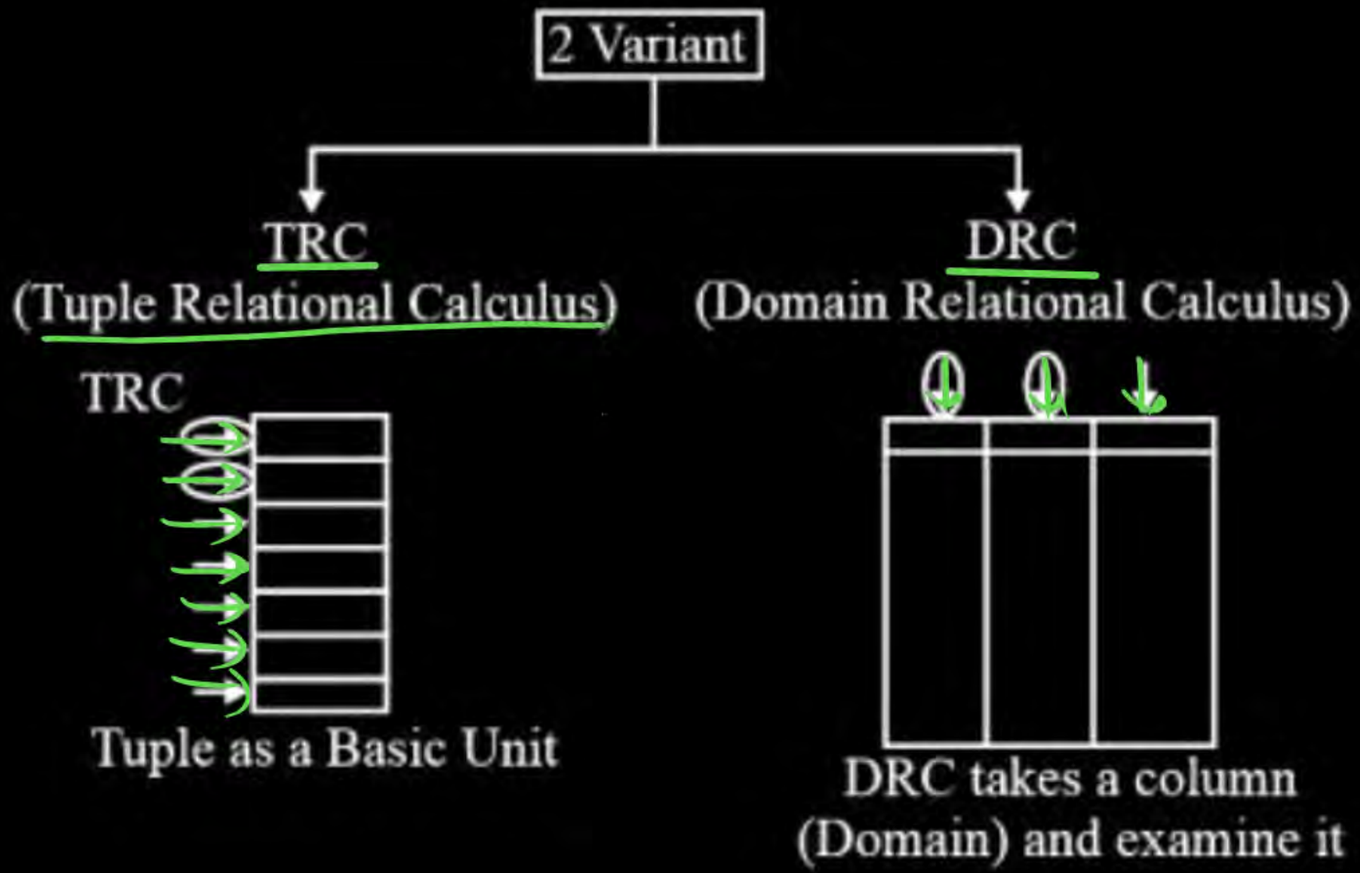
C. Number of entries in the relation.

D. Number of distinct domains of its relation schema.

TRC :

Relational Calculus (Based on predicate calculus)

Non procedural query language and has



TRC

- Query describe result in the form of set of Tuples.

Tuple Variable

- It is variable that takes on Tuples of a relation schema as value.

DRC

- Query describe result in the form of set of Column. (Domain).

Domain Variable

- It is variable that Range over the domain of some attribute (column).

Form of Query

$\{T \mid P(T)\}$

Tuple Variable

Formula which Describe Tuple variable

Form of Query

$\{<x_1 x_2 x_3 \ldots\ldots x_n> \mid P<x_1 x_2 x_3 \ldots\ldots x_n>\}$

Domain variable

Formula which describe Domain variable

TRC

{T|P(T)}

T : Tuple variable

P(T): Formula over Tuple (T)

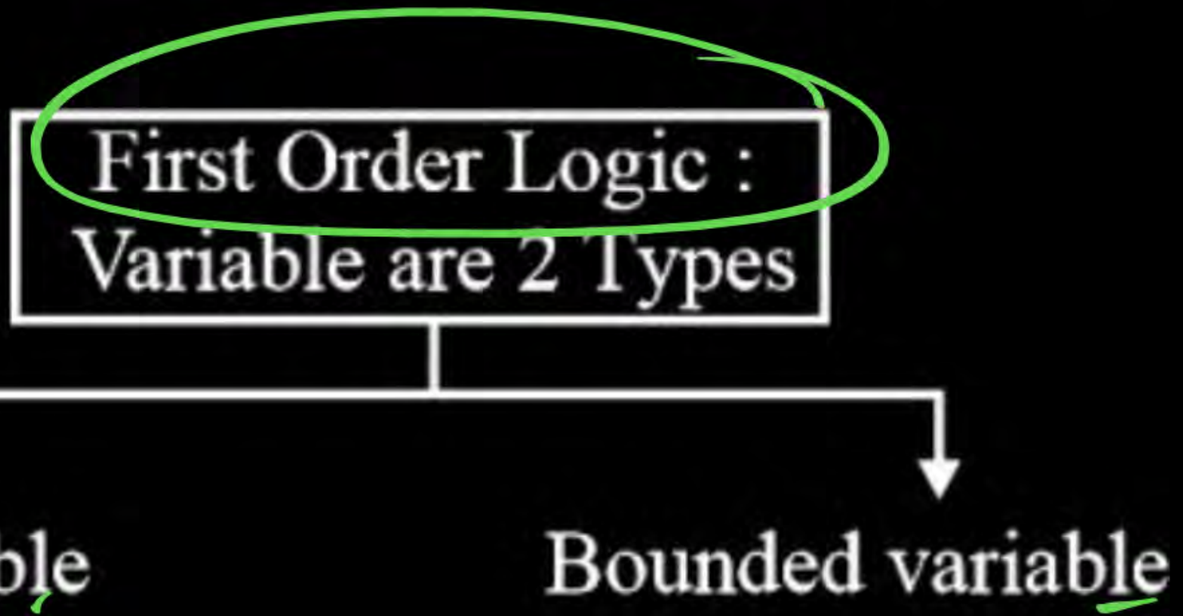
Such that P(T) is satisfied

Form of Query

T{Selected Sid

P(T){ FROM Student
WHERE Condition

NOTE: Formula of TRC is expressed using first order logic **(Predicate logic).**



Quantifiers

1. ∀ : For all (ALL)
2. ∃ : There exists (Any)

Bounded Variable: If tuple variable is preceded by the quantifier then it is Bounded variable.

Belong

1. $\forall \in$ Supplier

2. $\exists \in$ Supplier

Free variable: If tuple variable is not bounded by the quantifier then it is free variable.

NOTE: The result of TRC should be free variable.

Bounded Variable: If tuple variable is preceded by the quantifier then it is Bounded variable.

Belong

1. ∀ ∈ Supplier

2. ∃ ∈ Supplier

Free variable: If tuple variable is not bounded by the quantifier then it is free variable.

NOTE: The result of TRC should be free variable.

Ques.

LOAN (Loanno, Branch name, Amount)

Q. Find Loan no of amount above 50000.

TRC: {T | T∈ Loan (Total amount > 50,000)}

DRC: {<Loanno., branch name, amount>| < Loan no, Branch, amount ∈ Loan (amount > 50000) }

Q. Retrieve Sid of the supplier who supplied some Red color parts.

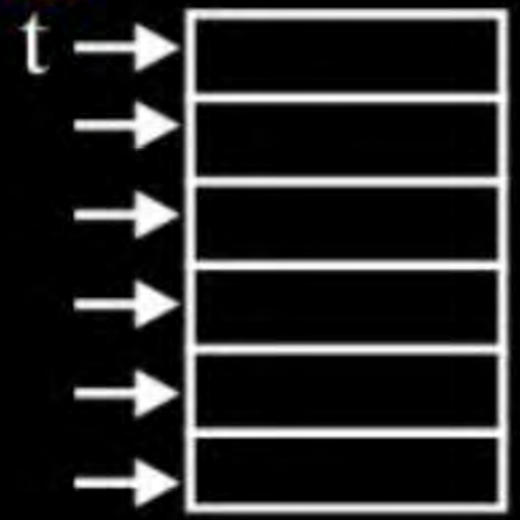
R.A:

$$\pi_{Sid}\left(\sigma_{C.Pid=P.Pid}\left(Catalog \times \sigma_{colour=Red}\left(Parts\right)\right)\right)$$

TRC:

$$\left(T \mid \exists C \in Catalog\left(\exists P \in parts\left(P.color = Red \wedge P.Pid = C.Pid\right) \wedge T = C.Sid\right)\right)$$

TRC:

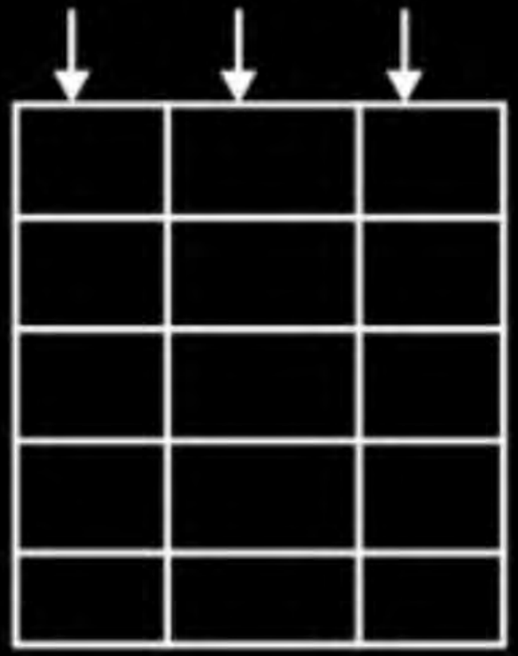


Takes a tuple and examine it

Tuple as a Basic unit

DRC: Takes a column

Range over the column or Domain and examine it



TRC: Unsafe operation occur

Unsafe operation: Student (t)

$\neg$(t): Not belongs to student table (infinite/Universe all tuple)

So, its unsafe operation.

THANK YOU!