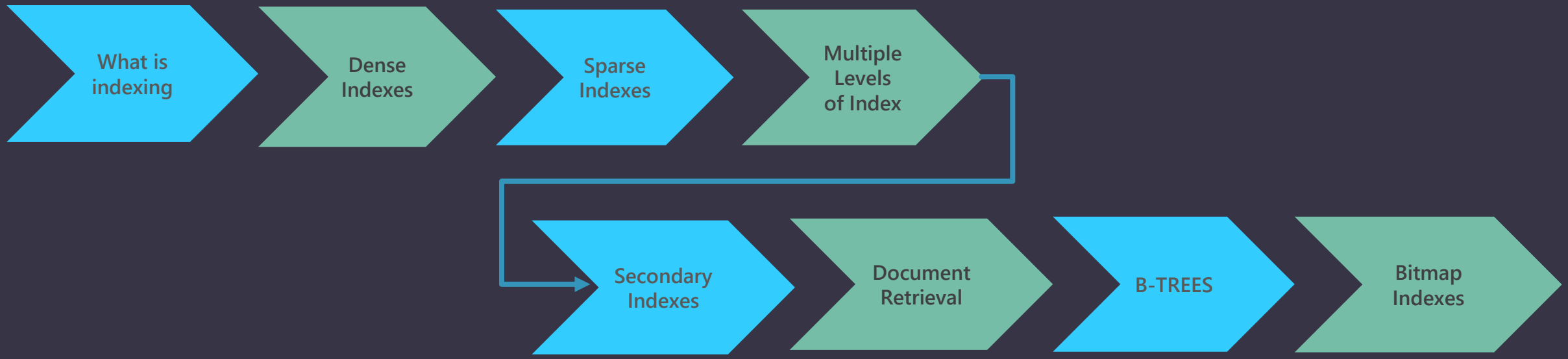
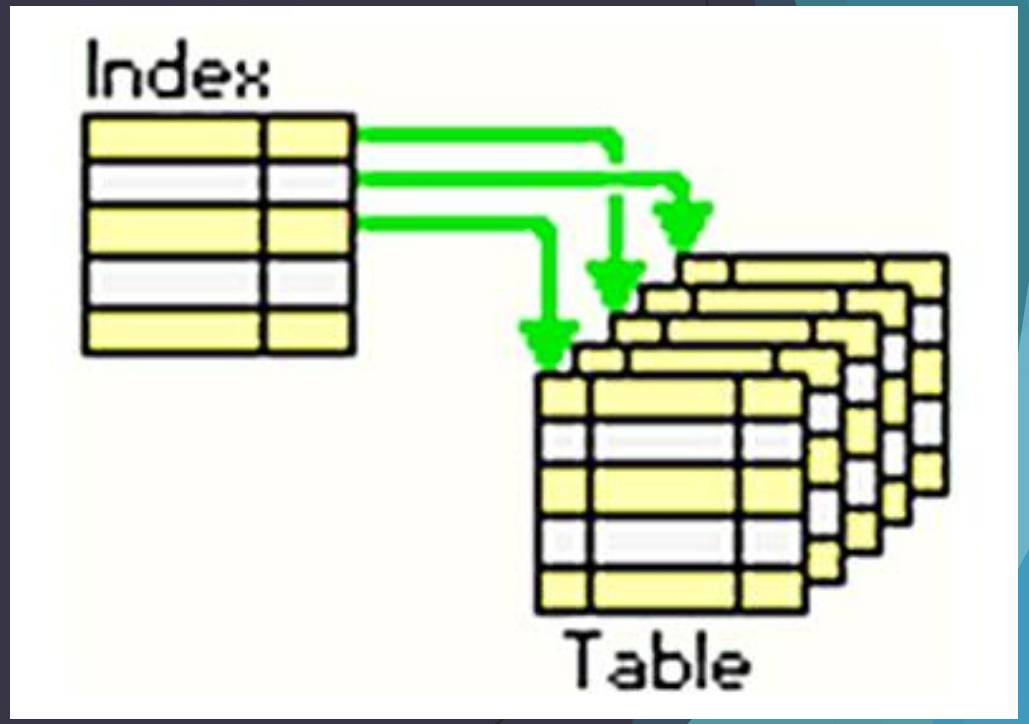
indexing structure

AGENDA

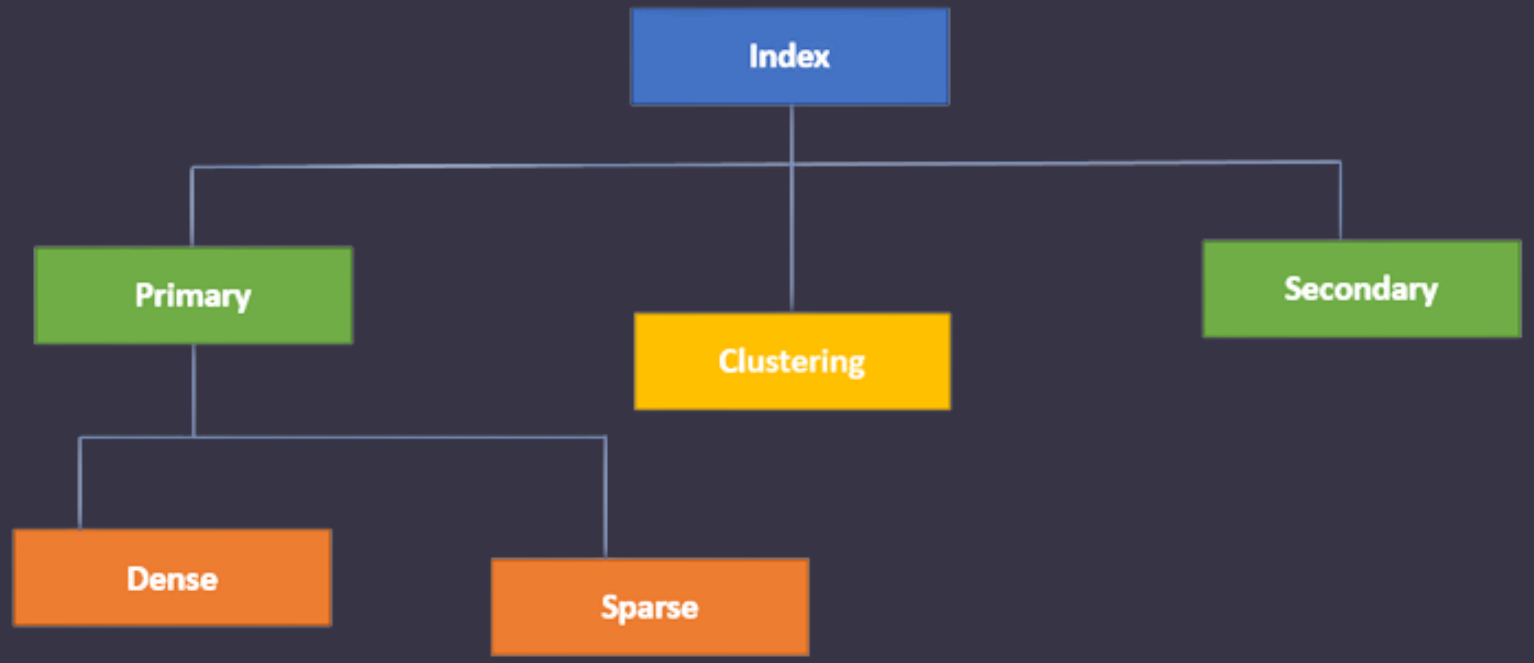

What is indexing
Dense Indexes
Sparse Indexes
Multiple Levels of Index
Secondary Indexes
Document Retrieval
B-TREES
Bitmap Indexes

What is indexing?

- Suppose you have a database with 20 million rows How long does it take to find a specific row?!
- How much space does it take? (RAM)



Types of Indexing:

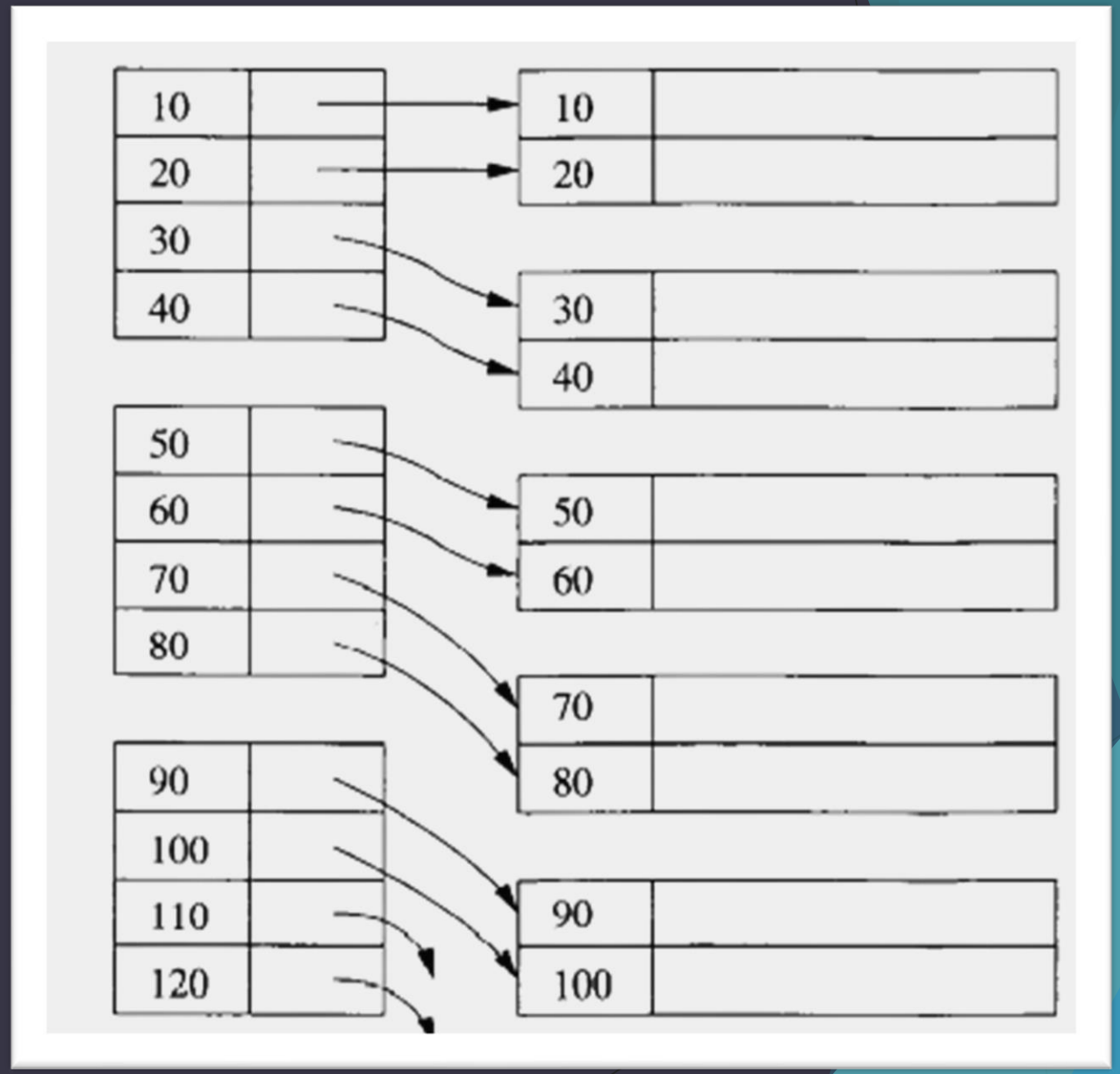


Dense Indexes

- records are sorted.
- needs more space to store index records.

- advantage:
  1. The number of index blocks is usually smaller than data blocks.
  2. Since keys are sorted, we can use binary search.

Example



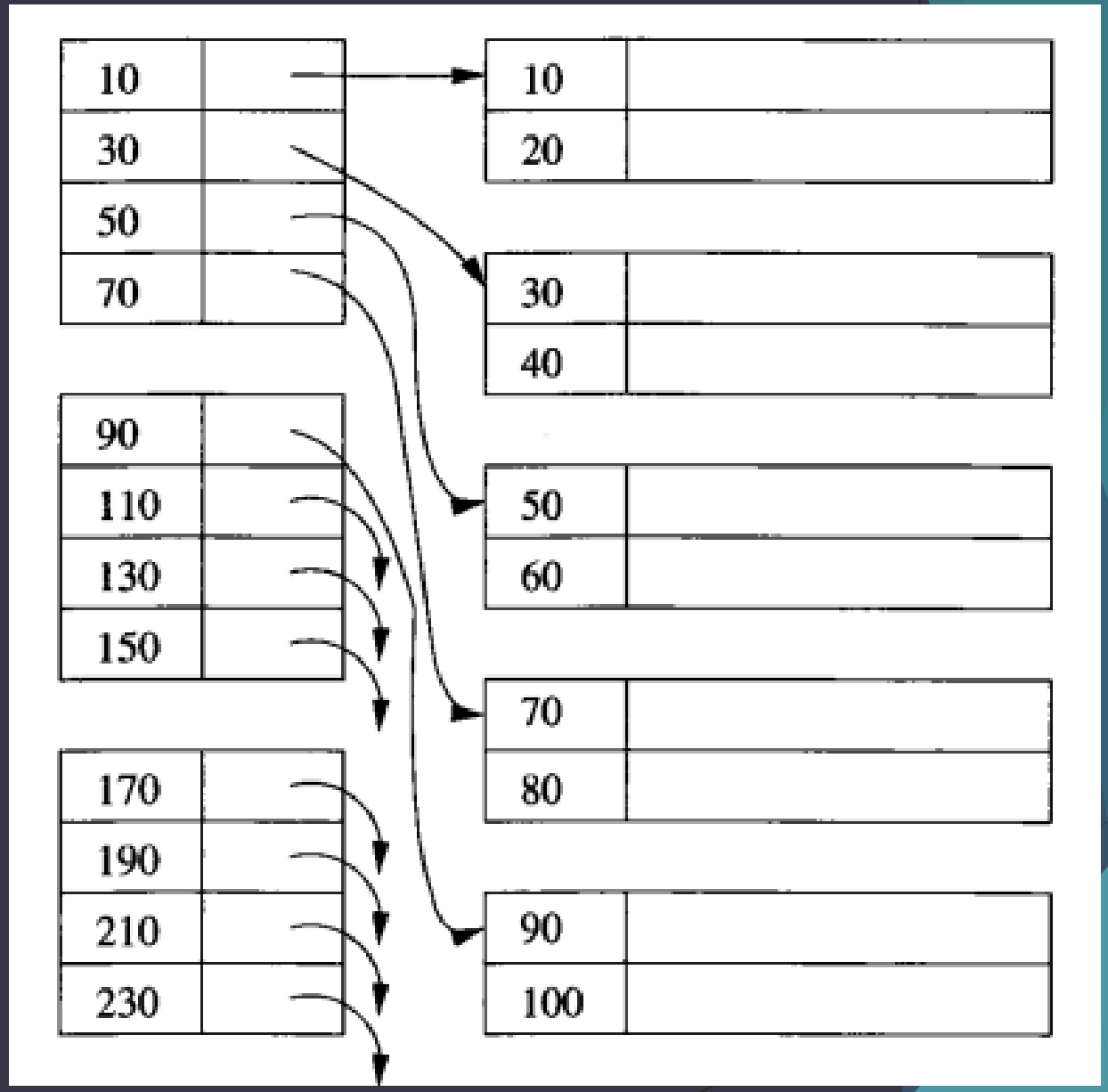
Sparse Indexes

- only use for e data file is sorted by key.
- a range of index columns stores the same data block address.

- advantage:

  less space than a dense index.

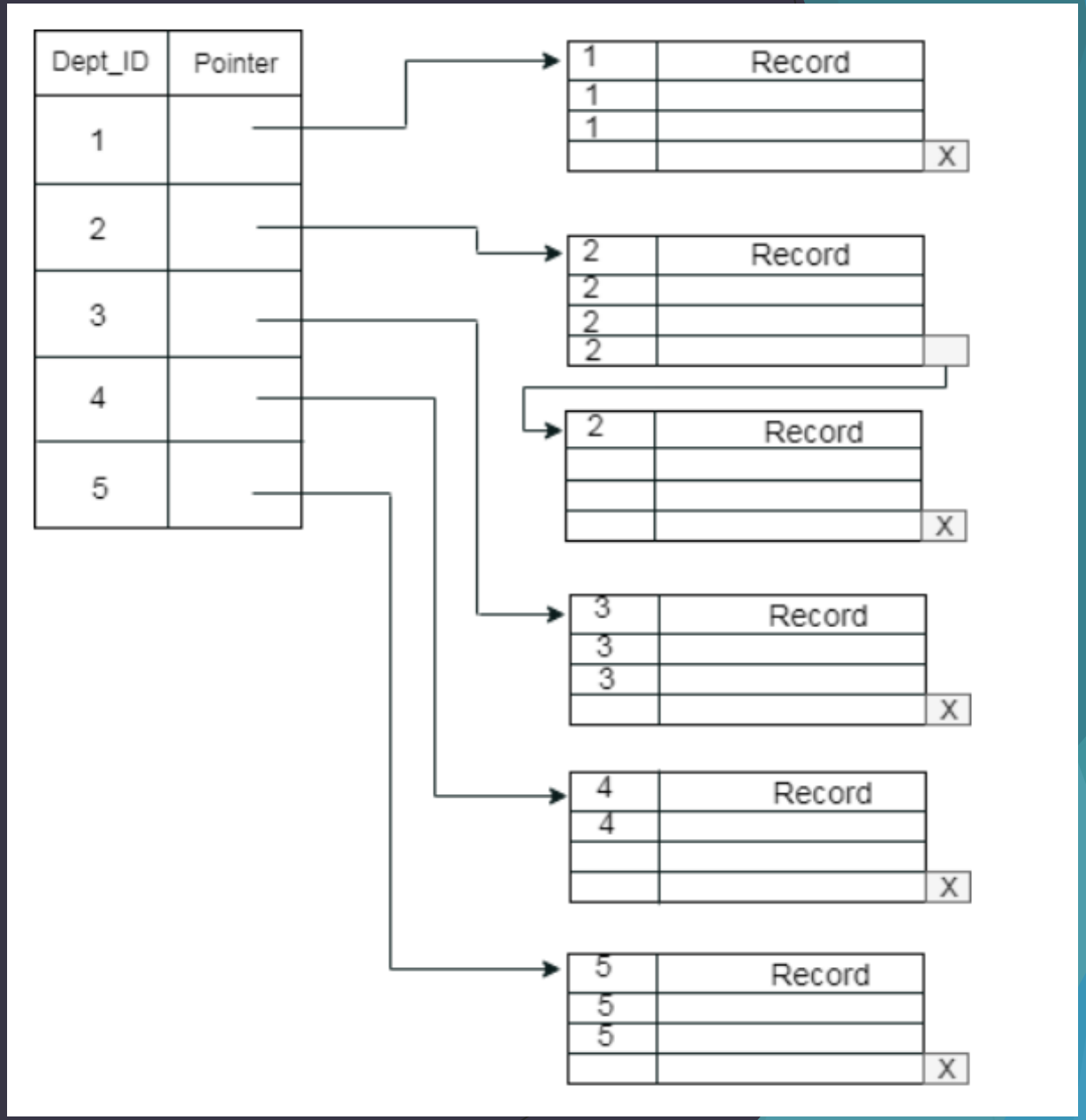
Example


10
30
50
70
90
110
130
150
170
190
210
230
10
20
30
40
50
60
70
80
90
100

Clustering Index

- index is created on non-primary key columns which may not be unique.
- group two or more columns to get the unique value and create index out of them.

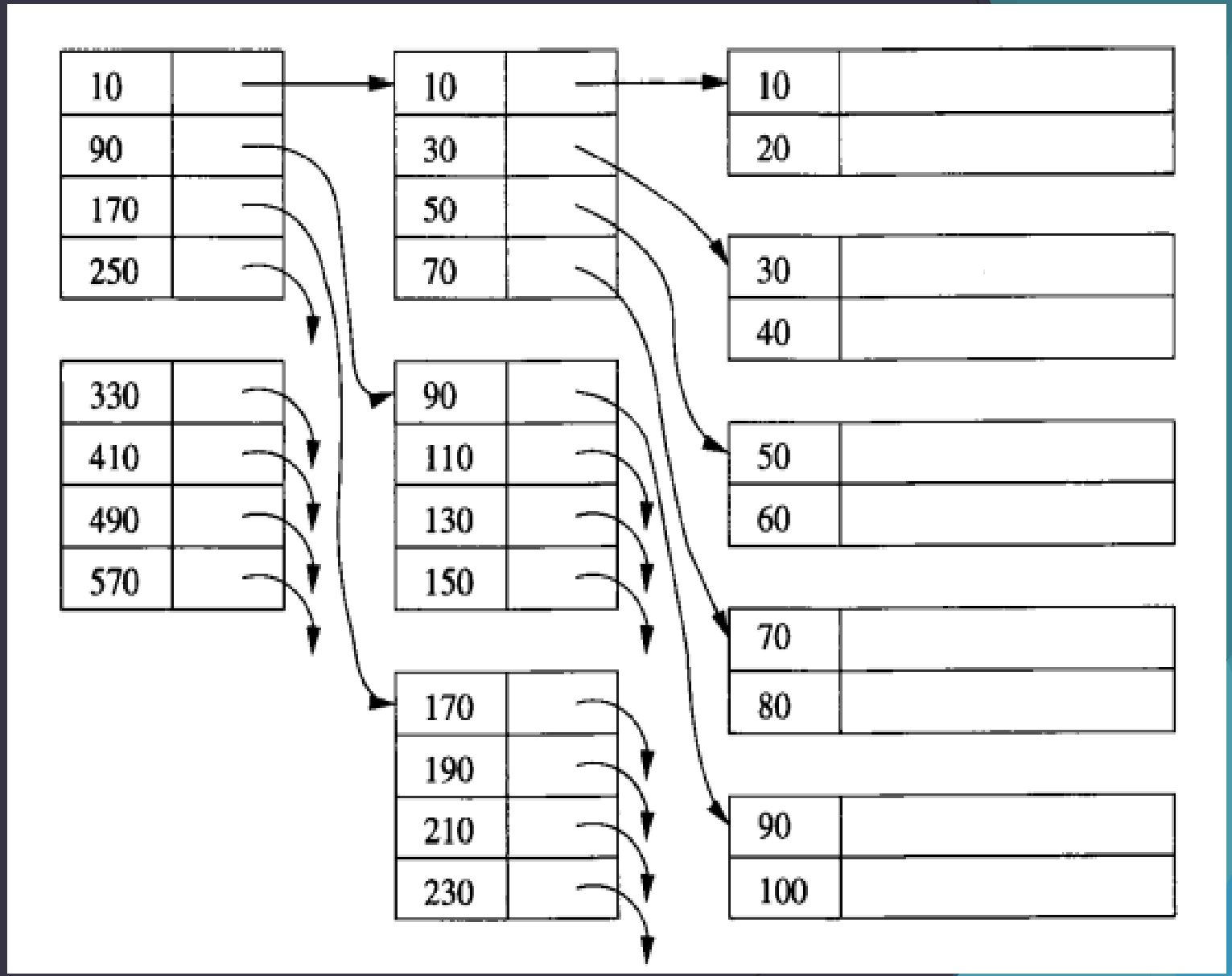
Example



Multiple Levels Indexing

- when a primary index does not fit in memory.
- putting an index on the index, can make the use more efficient.

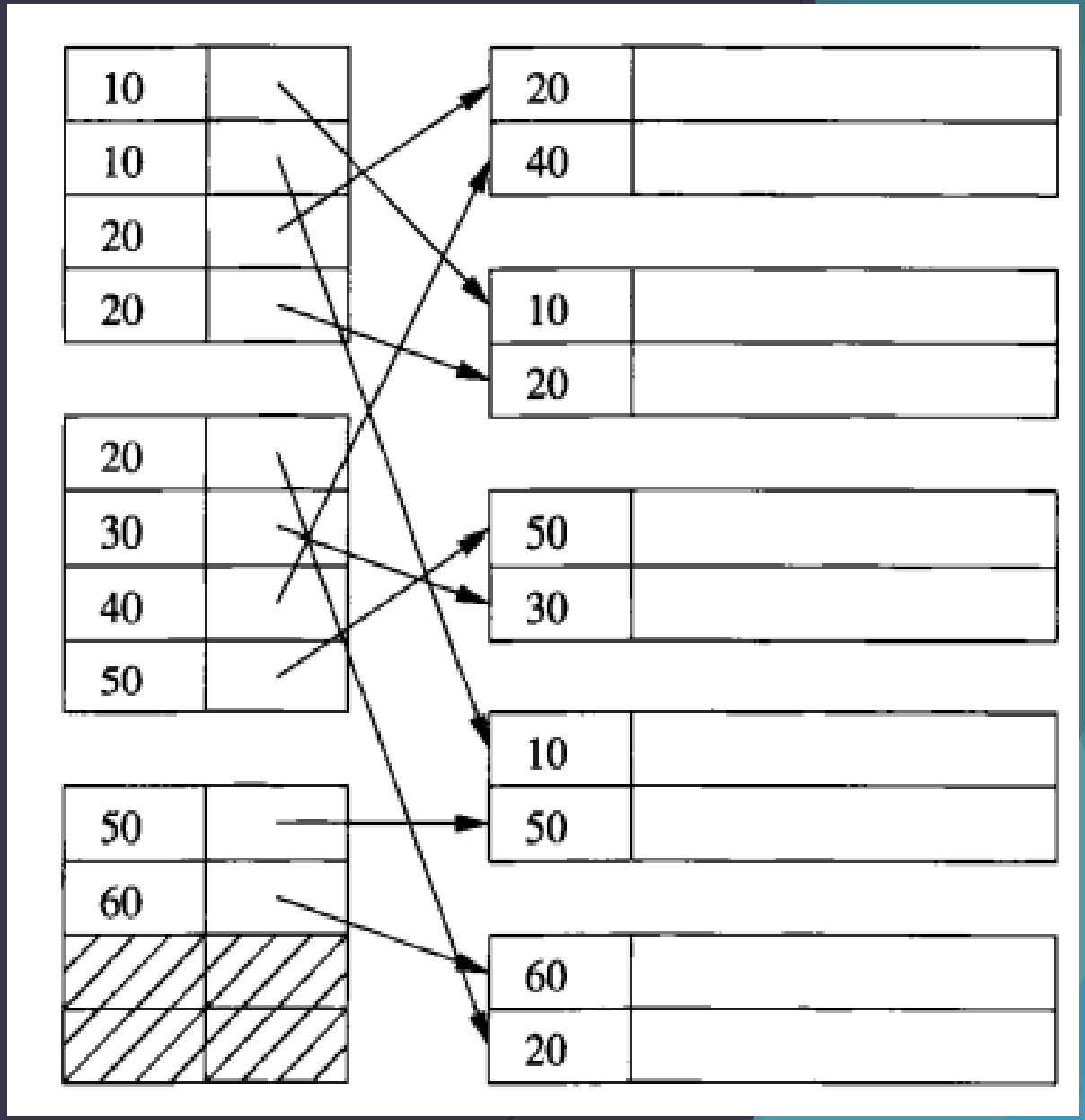
Example


10
90
170
250
330
410
490
570
10
30
50
70
90
110
130
150
170
190
210
230
10
20
30
40
50
60
70
80
90
100

Secondary Indexes

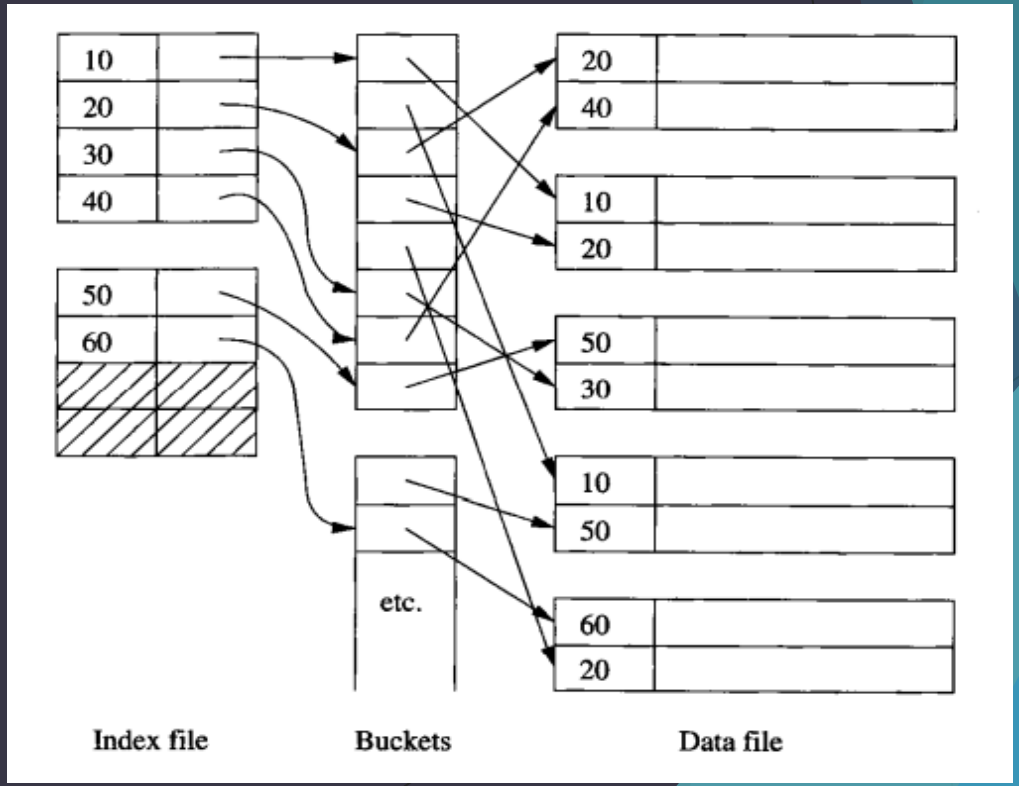
- Secondary indexes are always dense.
- unordered

But there is a problem!!!



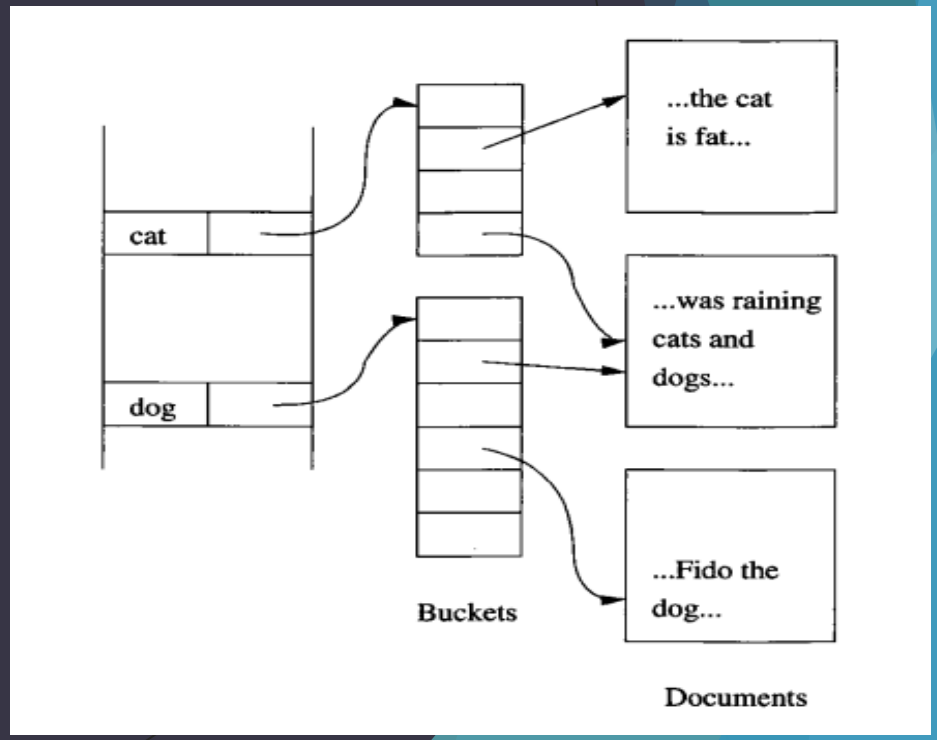
Here is a solution

- **buckets**
  - buckets, between the secondary index file and the data file



Document Retrieval

- the retrieval of documents given keywords
- Each attribute is boolean — either the word is present in the document, or it is not.
- the index has entries only for the search-key value TRUE.



B-Trees

- The B tree is a balanced binary search tree. It follows a multi-level index format.

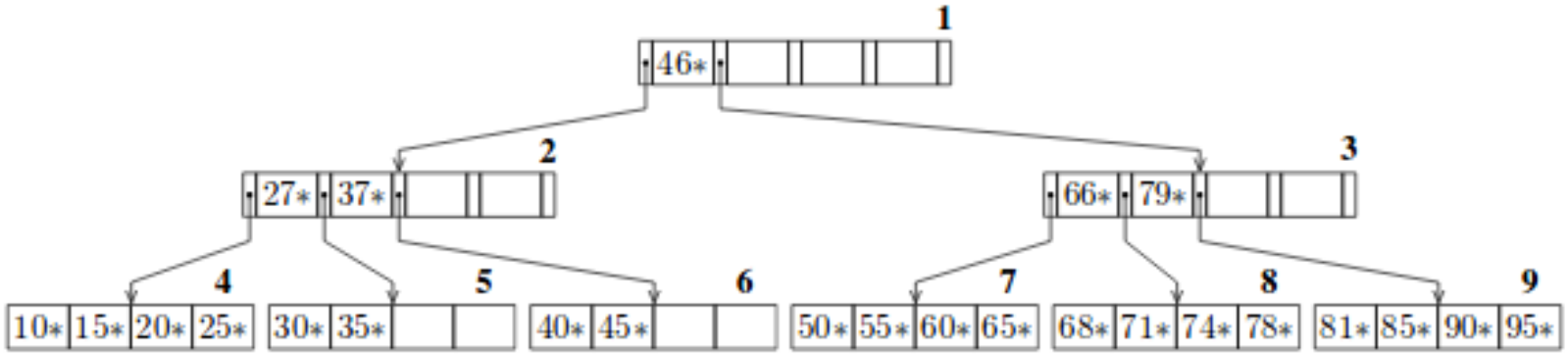
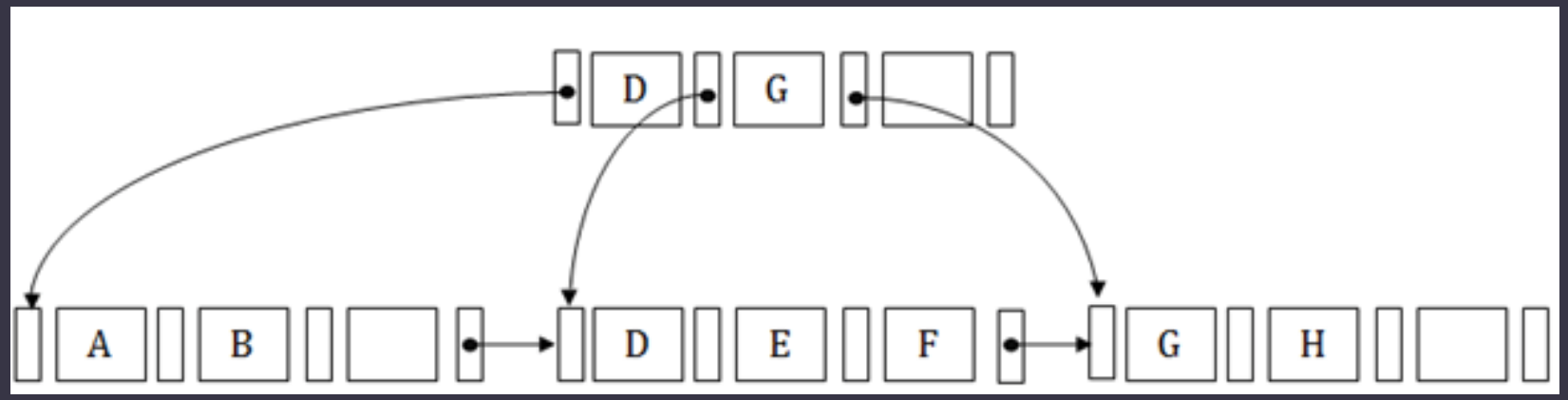


Figure 5.8: Example of a B–tree with $m = 5$

B+ Trees



B-Trees VS B+ Trees

B-Trees	B+ Trees
Data is stored in leaf nodes as well as internal nodes	Data is stored only in leaf nodes
Searching is a bit slower as data is stored in internal as well as leaf nodes	Searching is faster as the data is stored only in the leaf nodes
Leaf nodes cannot be linked together	Leaf nodes are linked together to form a linked list
No redundant search keys are present	Redundant search keys may be present

Bitmap Indexes

- Less memory use
- Use for low cardinality and columns are most frequently

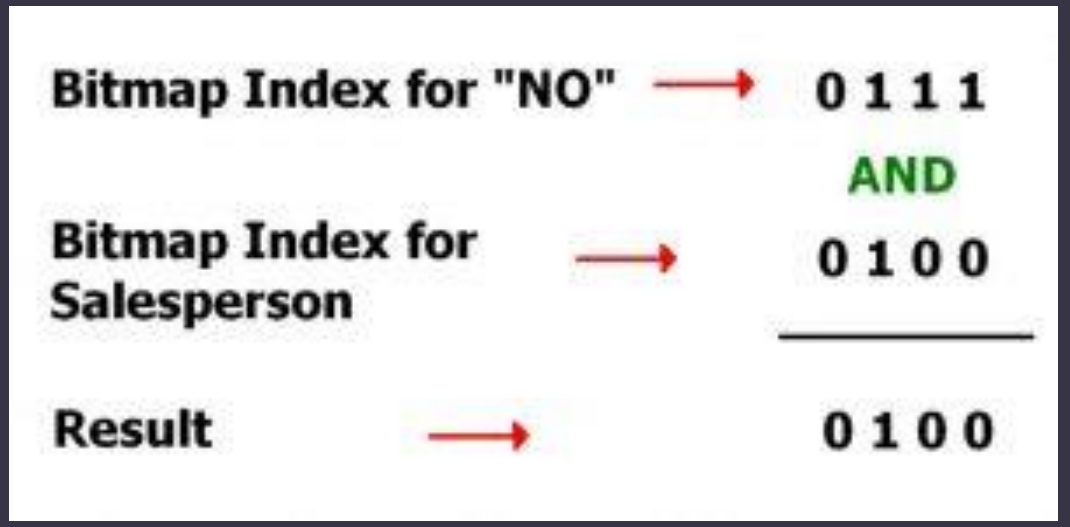
EmpNo	EmpName	Job	New_Emp	Salary
1	Alice	Analyst	Yes	15000
2	Joe	Salesperson	No	10000
3	Katy	Clerk	No	12000
4	Annie	Manager	Yes	25000

New_Emp Values	Bitmap Indices
Yes	1001
No	0110

Job Values	Bitmap Indices
Analyst	1000
Salesperson	0100
Clerk	0010
Manager	0001

- CREATE BITMAP INDEX index_New_Emp ON Employee (New_Emp);
- CREATE BITMAP INDEX index_Job ON Employee (Job);

- SELECT * FROM Employee WHERE New_Emp = "No" AND Job = "Salesperson";



Sources :

- DATABASE SYSTEMS The Complete Book Second Edition
- https://www.guru99.com/indexing-in-database.html
- https://www.javatpoint.com/indexing-in-dbms

THANK YOU

Amir Hasanebrahimi