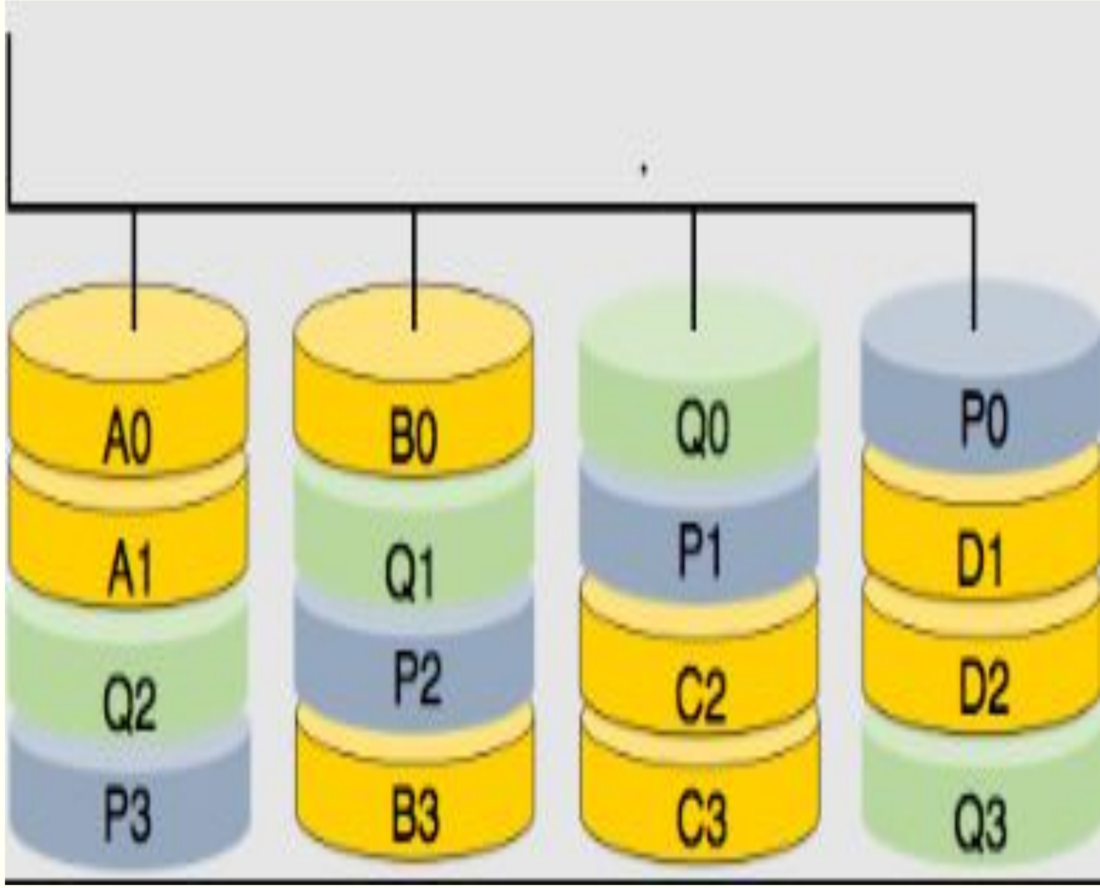
STORAGE STRATEGIES

UNIT-06



What is database Index?

- Indexes are **special lookup tables** that the **database search engine can use to speed up data retrieval**.

- A database index is a **data structure that improves the speed of data retrieval operations on a database table**.

- An index in a database is very similar to an index in the back of a book.

- Indexes are **used to retrieve data from the database very fast**. The users cannot see the indexes, they are just used to speed up searches/queries.

- **Updating a table with indexes takes more time** than updating a table without (because the indexes also need an update). So, only **create indexes on columns that will be frequently searched against**.

Syntax to create and drop an Index

- Syntax to create an index:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

- Example to create an index :

```
CREATE INDEX idx_studentname
ON Student (Studentname);
```

- Syntax to drop an index:

```
DROP INDEX table_name.index_name;
```

- Example to drop an index :

```
DROP INDEX Student.idx_studentname;
```

What is Indexing?

- Indexing is a **way to optimize the performance of a database** by **minimizing the number of disk accesses required** when a query is processed.

- It is a **data structure technique** which is **used to quickly locate and access the data in a database**.

Structure of Index in database

- Indexes are **created using a few database columns**.

search-key	pointer

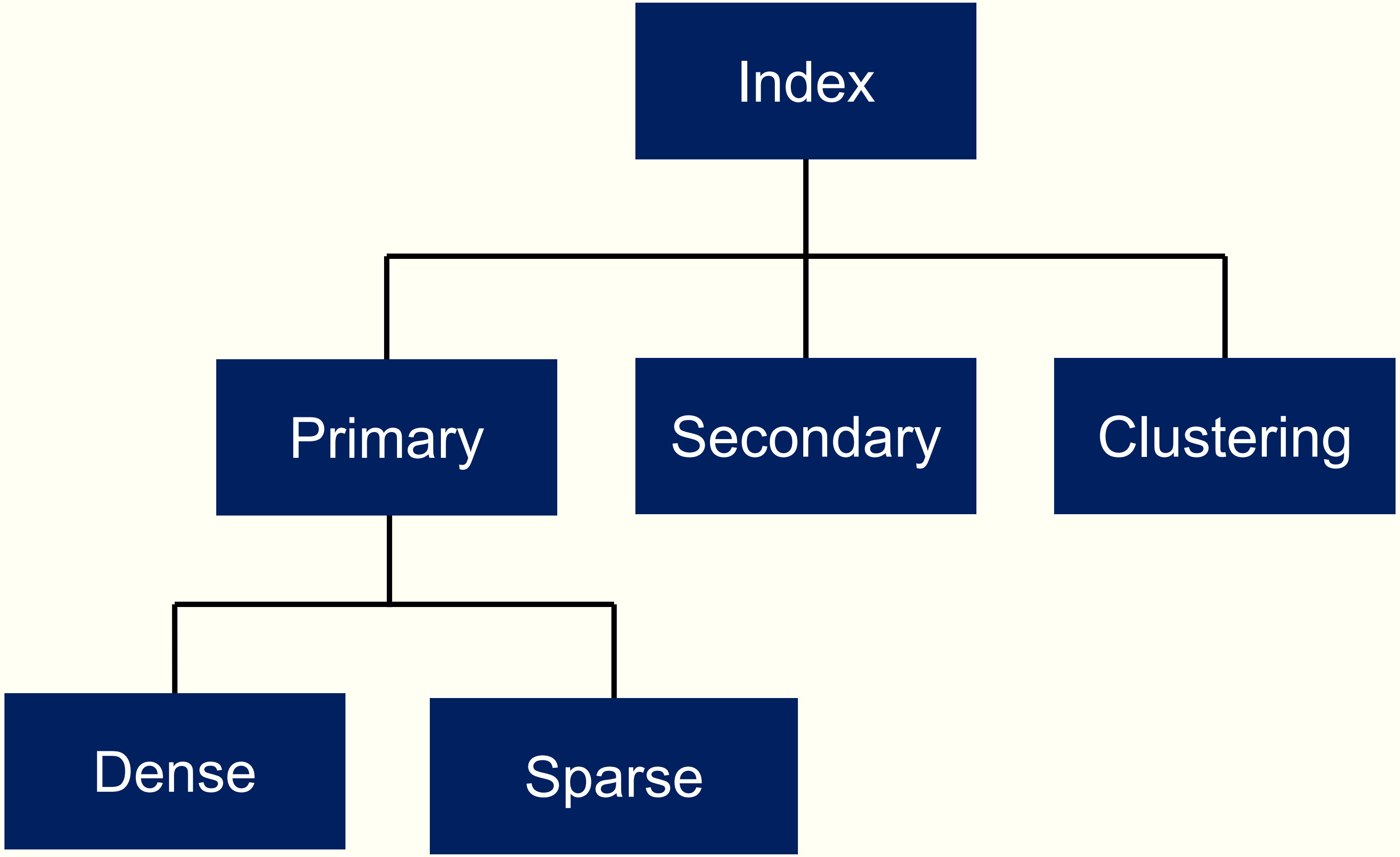
  - The first column is the **search key** that contains a **copy of the primary key** or **candidate key** of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.

  - The second column is the **data reference** or **pointer** which **contains a set of pointers holding the address of the disk block** where that particular key value can be found.

Cont…

- The indexing has various attributes:
  - **Access Types**: This refers to the **type of access** such as **value based search, range access**, etc.
  - **Access Time**: It refers to the **time needed to find particular data element** or set of elements.
  - **Insertion Time**: It refers to the **time taken to find the appropriate space and insert a new data**.
  - **Deletion Time**: **Time taken to find an item and delete it** as well as **update the index structure**.
  - **Space Overhead**: It refers to the **additional space required by the index**.

Indexing Methods (Types)



Primary Index (Ordered Index)

- If the **index is created on the primary key** of the table, then it is known as primary index. These primary keys are unique to each record.

- As primary keys are stored in sorted order, the **performance of the searching operation is quite efficient**.

- Student(RollNo, Name, Address, City, MobileNo)

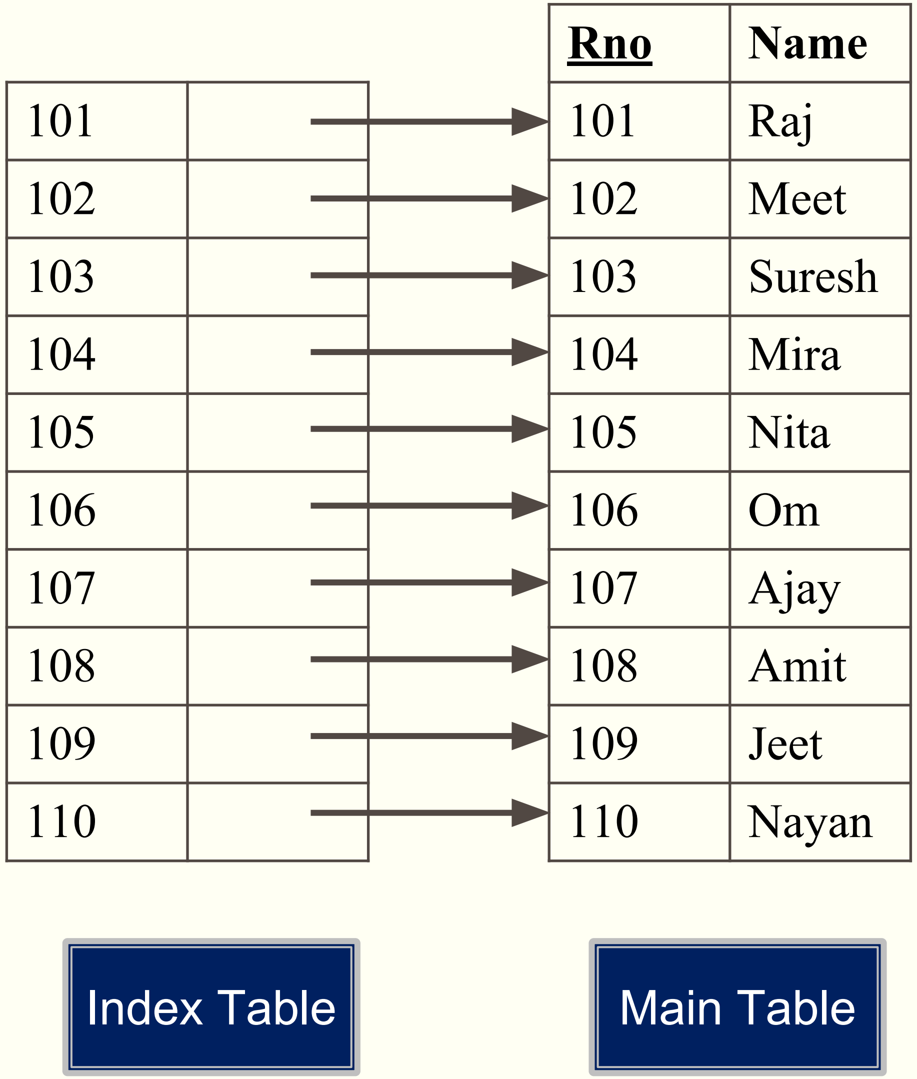
```
CREATE INDEX idx_StudentRno

ON Student (RollNo);
```

Exercise Create an Primary Index for Employee(EID, Name, Address, City).

- The primary index can be classified into two types:
  - Dense index
  - Sparse index

Dense Index

- In dense index, **there is an index record for every search key** value in the database.
- This makes **searching faster** but **requires more space** to store index records.
- In this, the **number of records in the index table is same as the number of records in the main table**.
- **Index records contain search key value and a pointer** to the actual record on the disk.



101	
102	
103	
104	
105	
106	
107	
108	
109	
110	

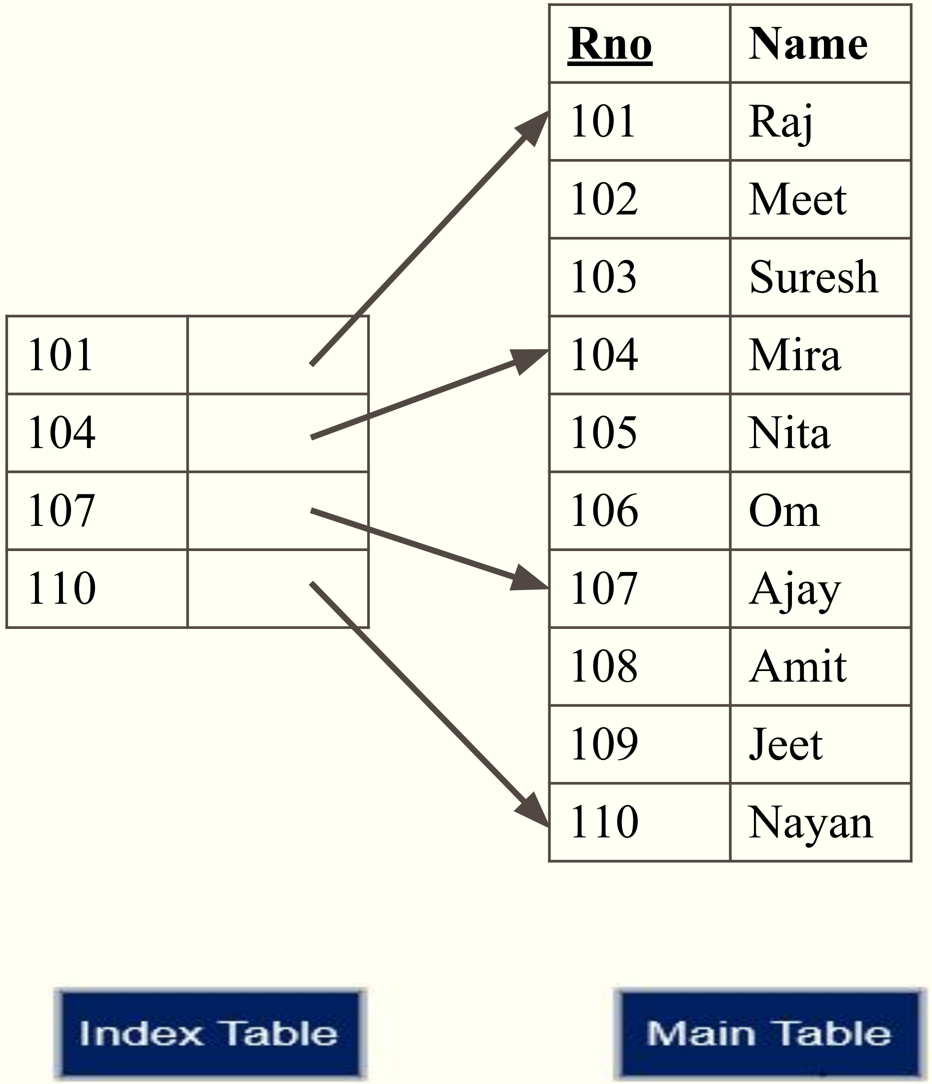
Index Table

Rno	Name
101	Raj
102	Meet
103	Suresh
104	Mira
105	Nita
106	Om
107	Ajay
108	Amit
109	Jeet
110	Nayan

Main Table

Sparse Index

- In sparse index, **index records are not created for every search key**.
- The index record appears only for a few items in the data file.
- It **requires less space**, less maintenance overhead for insertion, and deletions but is **slower** compared to the dense index for locating records.
- To search a record in sparse index we search for a value that is less than or equal to value in index for which we are looking.
- After getting the first record, linear search is performed to retrieve the desired record.
- In the sparse indexing, as the size of the main table grows, the size of index table also grows.



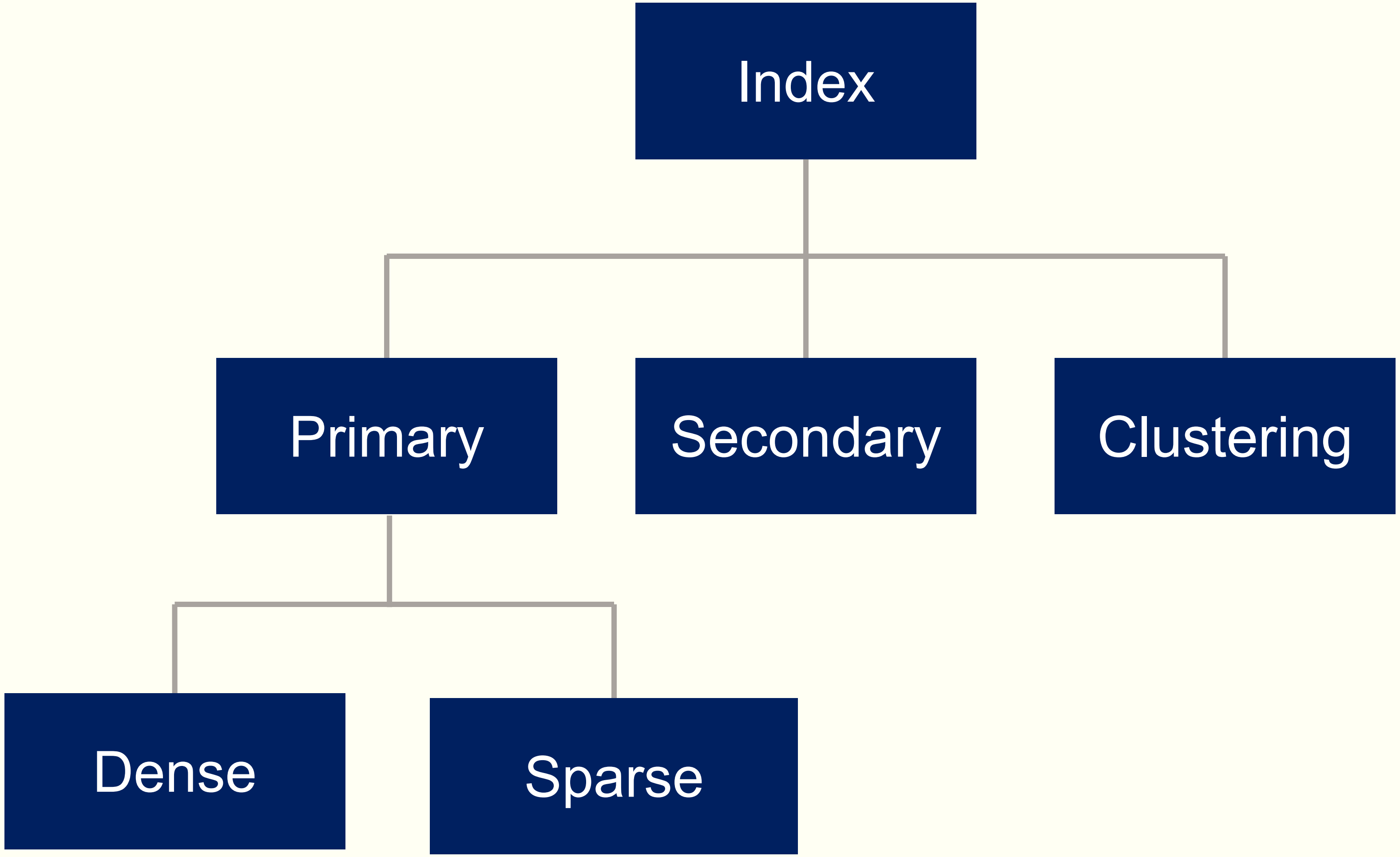
101	
104	
107	
110	

Index Table

Rno	Name
101	Raj
102	Meet
103	Suresh
104	Mira
105	Nita
106	Om
107	Ajay
108	Amit
109	Jeet
110	Nayan

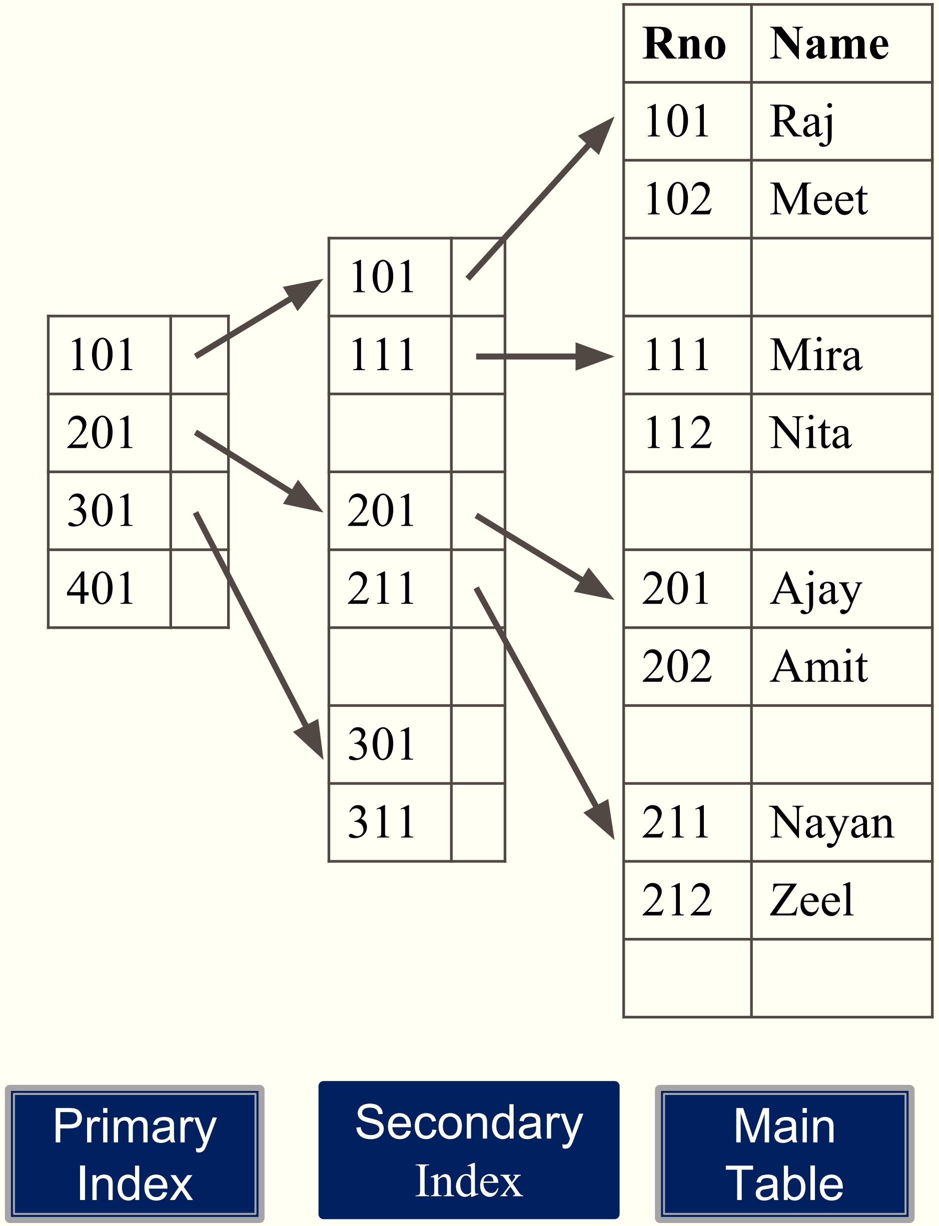
Main Table

Indexing Methods (Types)



Secondary Index (Non-clustering Index) (Multilevel Index)

- In secondary indexing, **to reduce the size of mapping, another level of indexing is introduced**.
- In this method, the **huge range for the columns is selected** initially so that the mapping size of the first level becomes small.
- Then **each range is further divided** into smaller ranges.
- The **mapping of the first level is stored in the primary memory**, so that address fetch is faster.
- The **mapping of the second level and actual data are stored in the secondary memory** (hard disk).



101	
201	
301	
401	

101	
111	
201	
211	
301	
311	

Rno	Name
101	Raj
102	Meet
111	Mira
112	Nita
201	Ajay
202	Amit
211	Nayan
212	Zeel

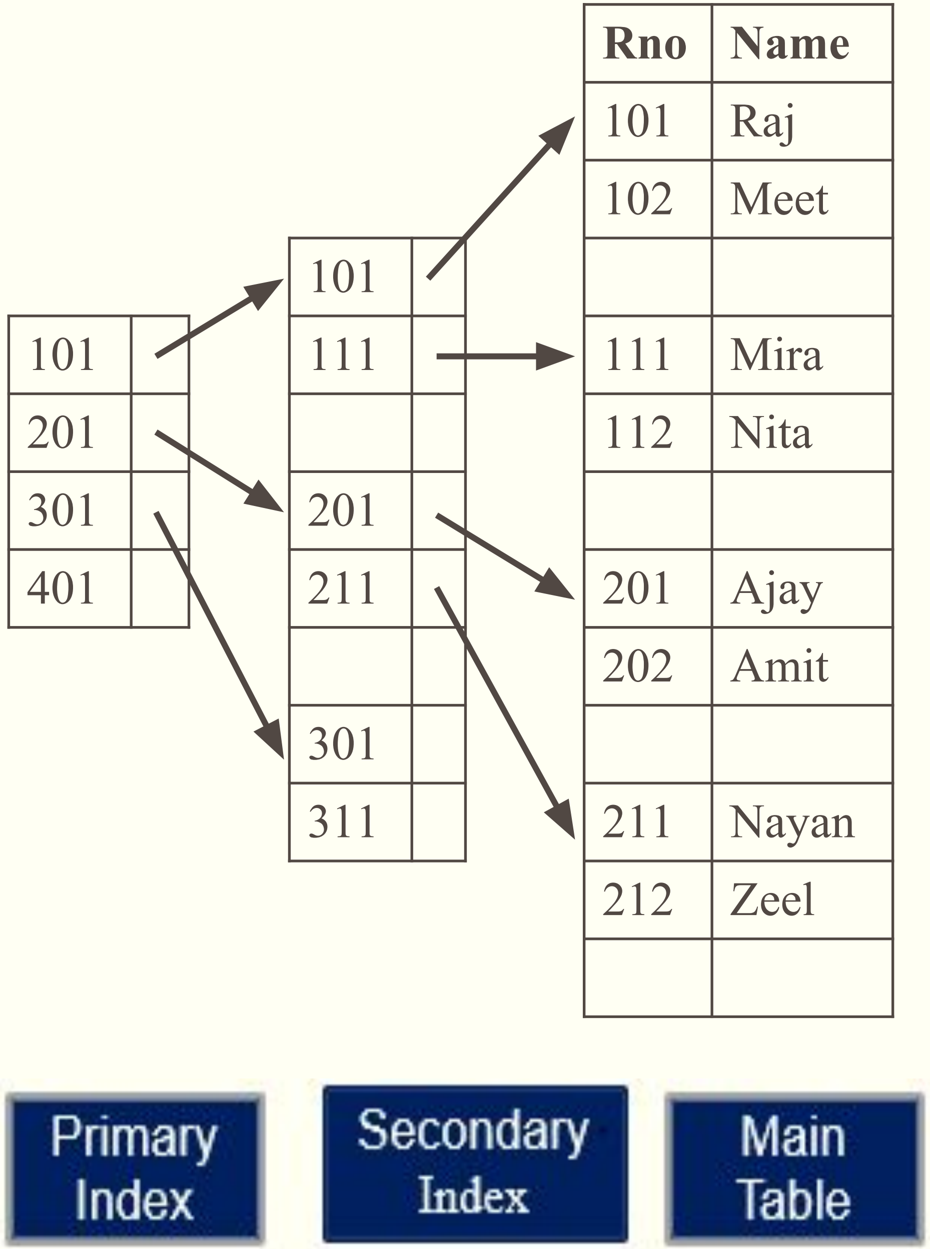
Secondary Index (How to find a particular record?)

- If you want to find the record of roll 112, then it will search the highest entry which is smaller than or equal to 112 in the first level index. It will get 101 at this level.

- Then in the second index level, again it does max (112) <= 112 and gets 111. Now using the address 111, it goes to the data block and starts searching each record till it gets 112.

- This is how a search is performed in this method.

- Inserting, updating or deleting is also done in the same manner.

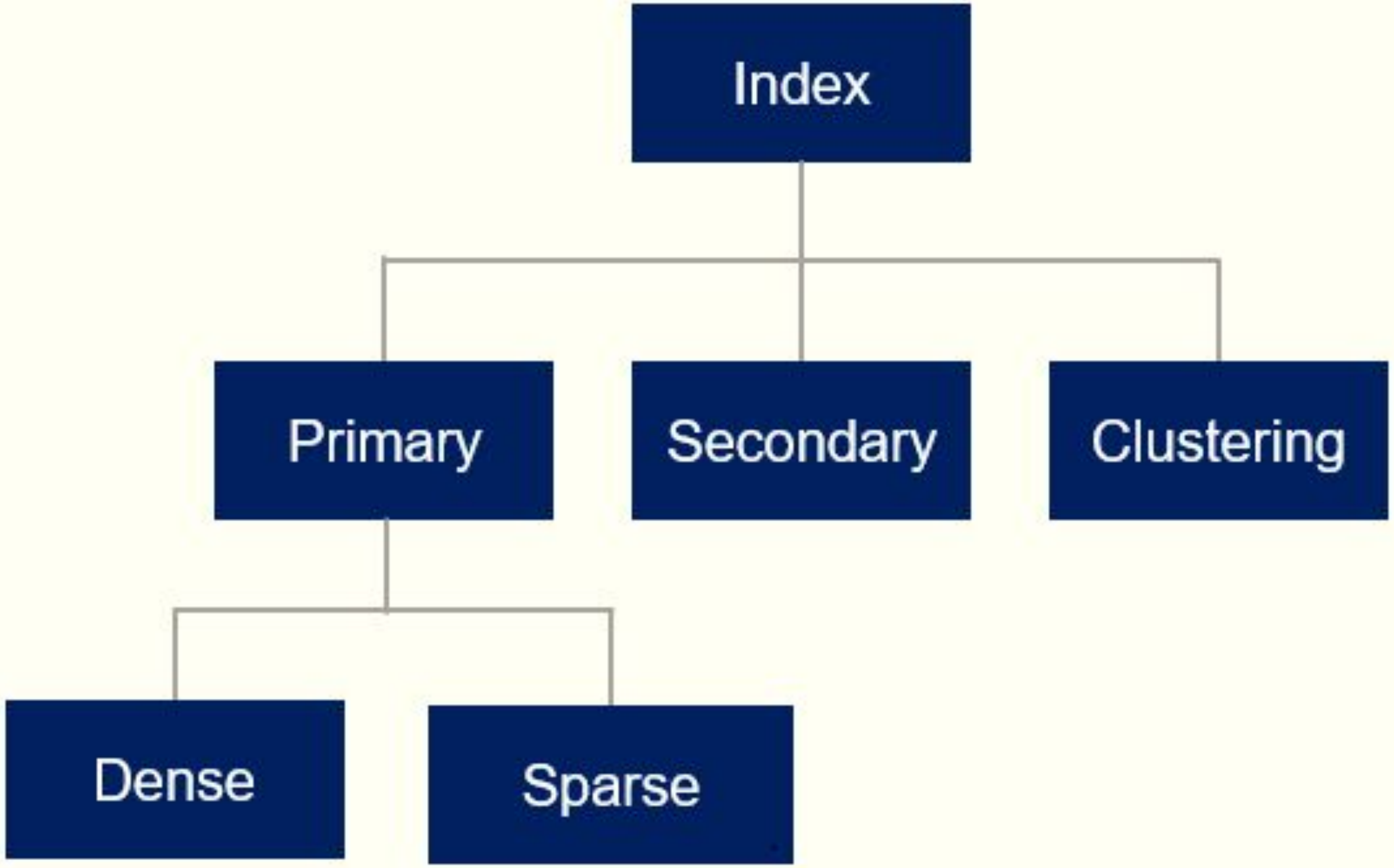


101	
201	
301	
401	

101	
111	
201	
211	
301	
311	

Rno	Name
101	Raj
102	Meet
111	Mira
112	Nita
201	Ajay
202	Amit
211	Nayan
212	Zeel

Indexing Methods (Types)

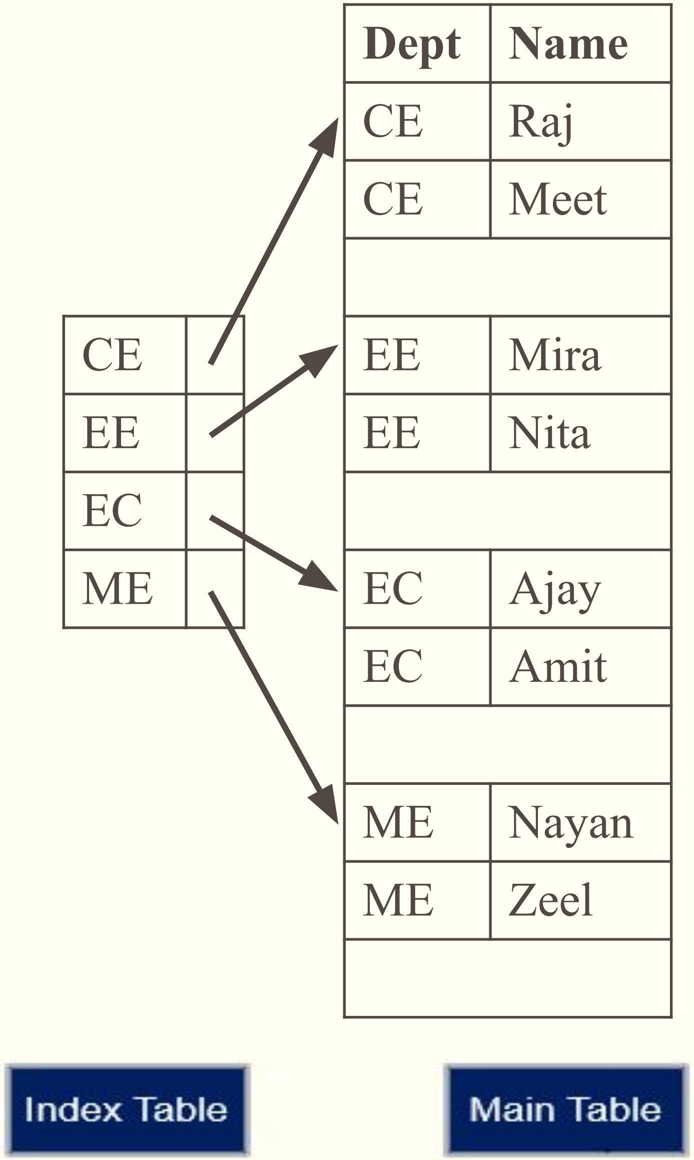


Clustering Index

- Sometimes the **index is created on non-primary key columns** which may not be unique for each record.

- In this case, **to identify the record faster, we will group two or more columns to get the unique value and create index out of them**. This method is called a clustering index.

- The records which have similar characteristics are grouped, and indexes are created for these group.



CE	
EE	
EC	
ME	

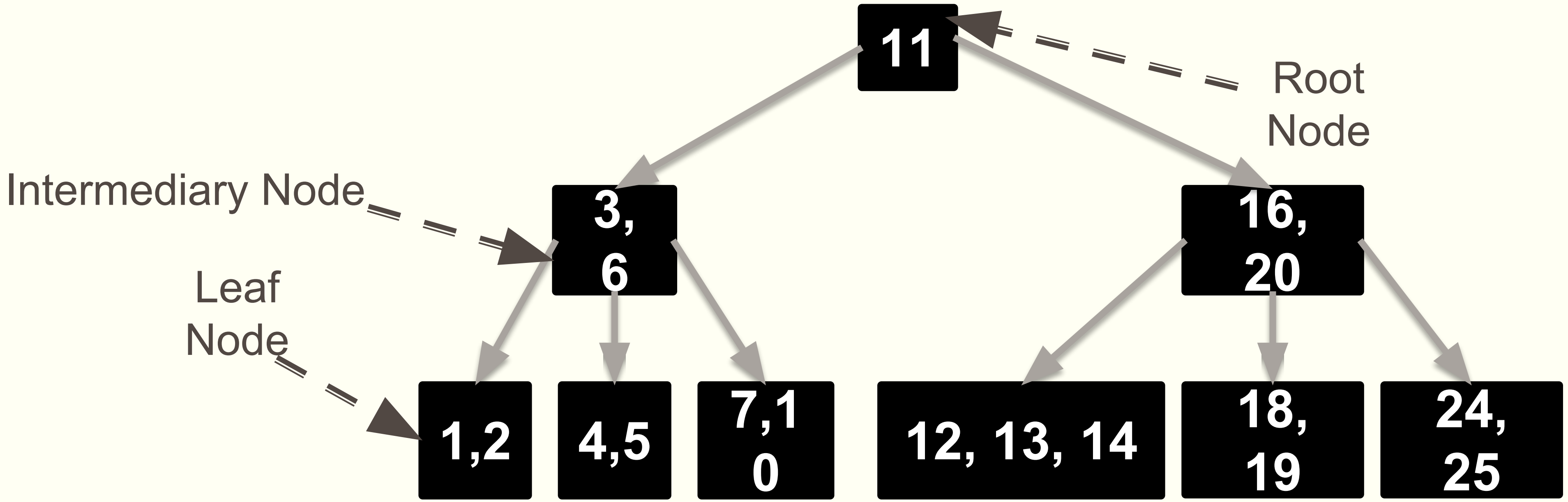
Dept	Name
CE	Raj
CE	Meet
EE	Mira
EE	Nita
EC	Ajay
EC	Amit
ME	Nayan
ME	Zeel

B-TREE

Section – 2

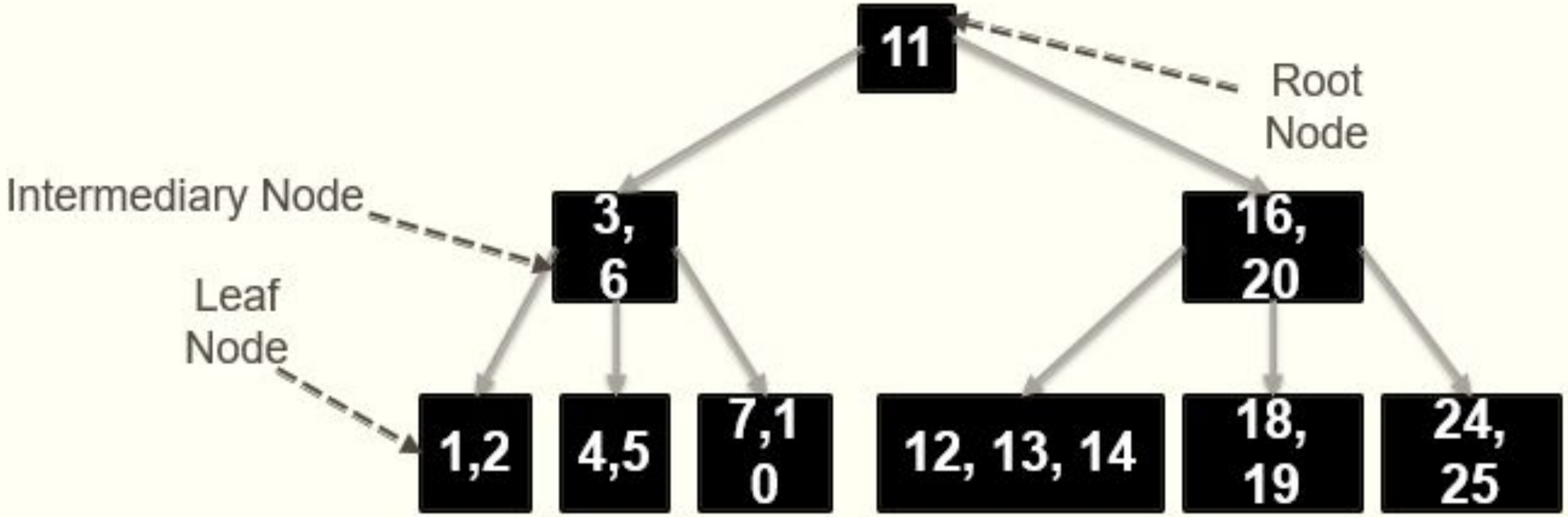
B-tree

- B-tree is a **data structure that store data in its node in sorted order**.



- B-tree **stores data in such a way that each node contains keys in ascending order**.
- Each of these **keys has two references to another two child nodes**.
- The **left side child node keys are less than the current keys** and the **right side child node keys are greater than the current keys**.

B-tree (How to search a particular node?)



- Suppose we want to search 18 in the above B tree structure.

- First, we will fetch for the intermediary node which will direct to the leaf node that can contain a record for 18.

- So, in the intermediary node, we will find a branch between 16 and 20 nodes.

- Then at the end, we will be redirected to the fifth leaf node. Here DBMS will perform a sequential search to find 18.

HASHING

Section – 3

Hashing

- For a huge database, it can be almost next to impossible to search all the index values through all its level and then reach the destination data block to retrieve the desired data.

- Hashing is a **technique to directly search the location of desired data on the disk without using index structure**.

- **Data is stored in the form of data blocks whose address is generated by applying a hash function** in the memory location where these records are stored known as a data block or data bucket.

- Hashing uses hash functions with search keys as parameters to generate the address of a data record.

- **Data bucket**: Data buckets are the memory locations where the records are stored.

- **Hash Function**: Hash function is a mapping function that maps all the set of search keys to actual record address. Generally, hash function uses primary key to generate the hash index – address of the data block.

- **Types of hashing methods** are **Static hashing and Dynamic hashing**

Static hashing

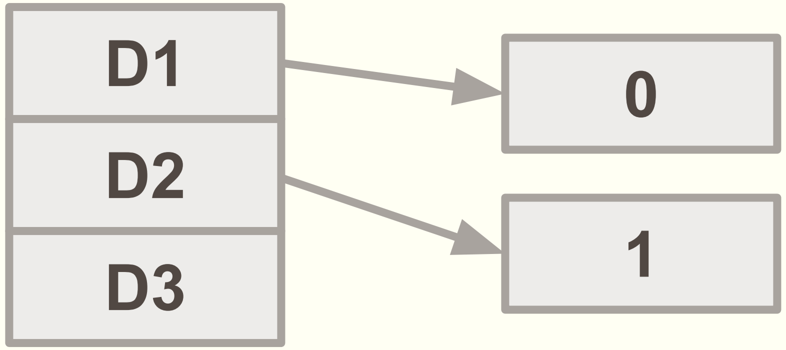
- In the static hashing, the **resultant data bucket address will always remain the same**.
- Therefore, if you generate an address for say Student_ID = 10 using hashing function mod(3), the resultant bucket address will always be 1. So, you will not see any change in the bucket address.
- Therefore, in this static hashing method, the number of data buckets in memory always remains constant.

Dynamic hashing

- The drawback of static hashing is that that it does not expand or shrink dynamically as the size of the database grows or shrinks.
- In dynamic hashing, data buckets grows or shrinks (added or removed dynamically) as the records increases or decreases.
- Dynamic hashing is also known as extended hashing.

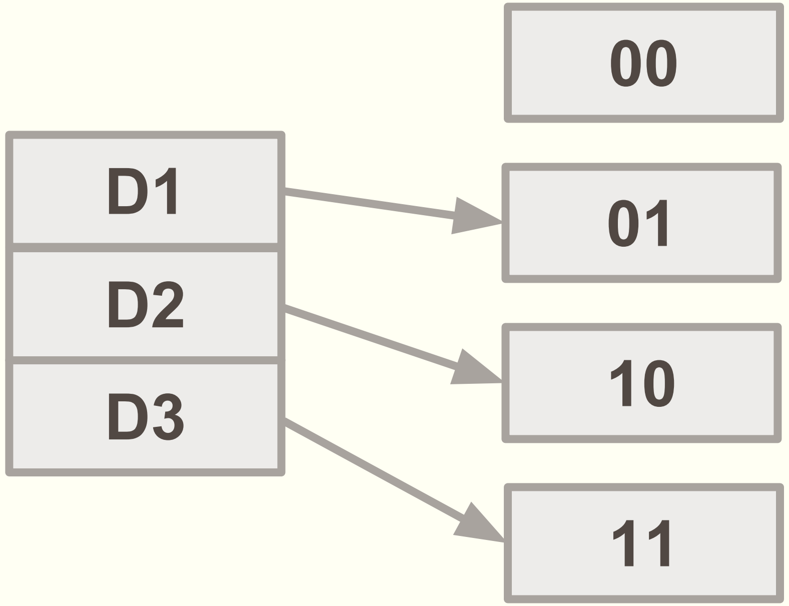
Dynamic hashing

- In dynamic hashing, the hash function is made to produce a large number of values.
- For Example, there are three data records D1, D2 and D3 .
- The hash function generates three addresses 0101, 1001 and 1010 respectively.
- This method of storing considers only part of this address – especially only first one bit to store the data.
- So it tries to load three of them at address 0 and 1.



Dynamic hashing

- But the problem is that no bucket address is remaining for D3.
- The bucket has to grow dynamically to accommodate D3.
- So it changes the address have 2 bits rather than 1 bit, and then it updates the existing data to have 2 bit address.
- Then it tries to accommodate D3.



Questions asked in GTU

1. Explain indexing and different types of indexes.
2. Explain B-tree.
3. Explain hashing.

Thank you….!!