Indexing in DBMS :-

- Indexing in used to optimize the performance of a database by minimizing the number of disk accesses Required when a query is processed.

- The index is a type of data structure. It is used to located and access the data in a database table quikly.

Index Structure :-

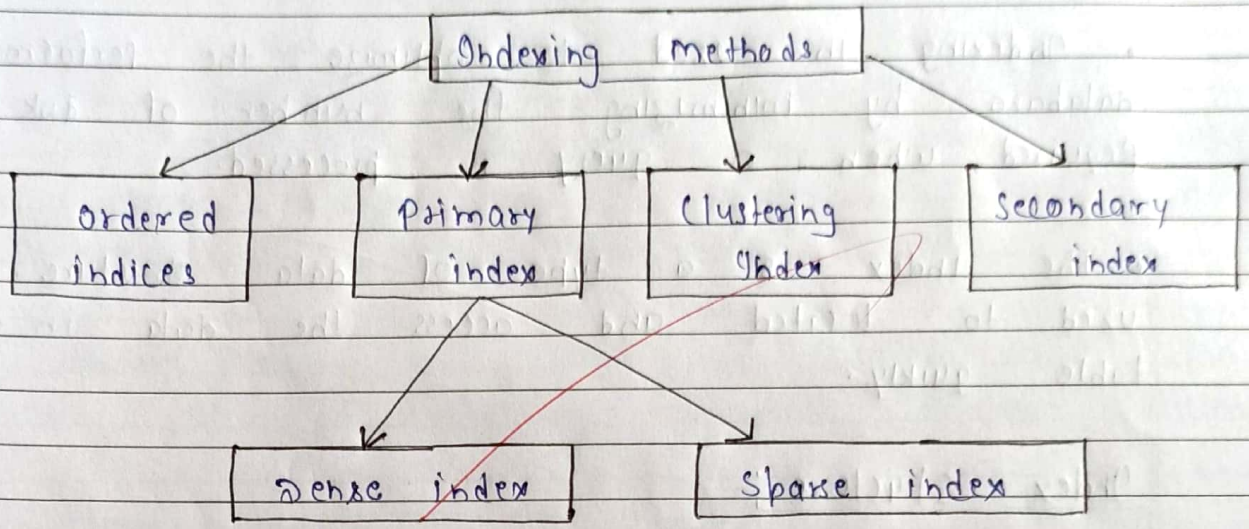
Index can be created using some database Columns.

search key	Data Reference

- The first Column of the database is the search key that Contains a Copy of the Primary key or Candidate key of the table. The values of the Primary key are stored in sorted order so that the Corresponding data can be accessed easily.

- The second Column of the database is the data Reference. It Contain a set of Pointers holding the address of the disk block where the value of the Particular key can be found.

Indexing methods :-



Ordered indices :-

the indices are usually sorted to make Searching faster. the indices which are sorted are known as ordered indices.

Example- Supose we have an employee table with thousands of Record and each of which is 10 bytes long.

- In the case of database with no index, we have to search the disk block from starting till it Reaches 543.

- In the case of an index, we will search using Indexes and the DBMS will Read the Record after Reading 542 * 2 = 1084 bytes.

Primary Index :-

- If the Index is Created on the basis of the Primary key of the table, then it is known as Primary Indexing. These Primary keys are Unique to Each Record and Contain 1:1 Relation between the Records.

- As Primary keys are stored in Sorted order, the Performance of the Searching Operation is quite efficient.

- The Primary index can be classified into two types.

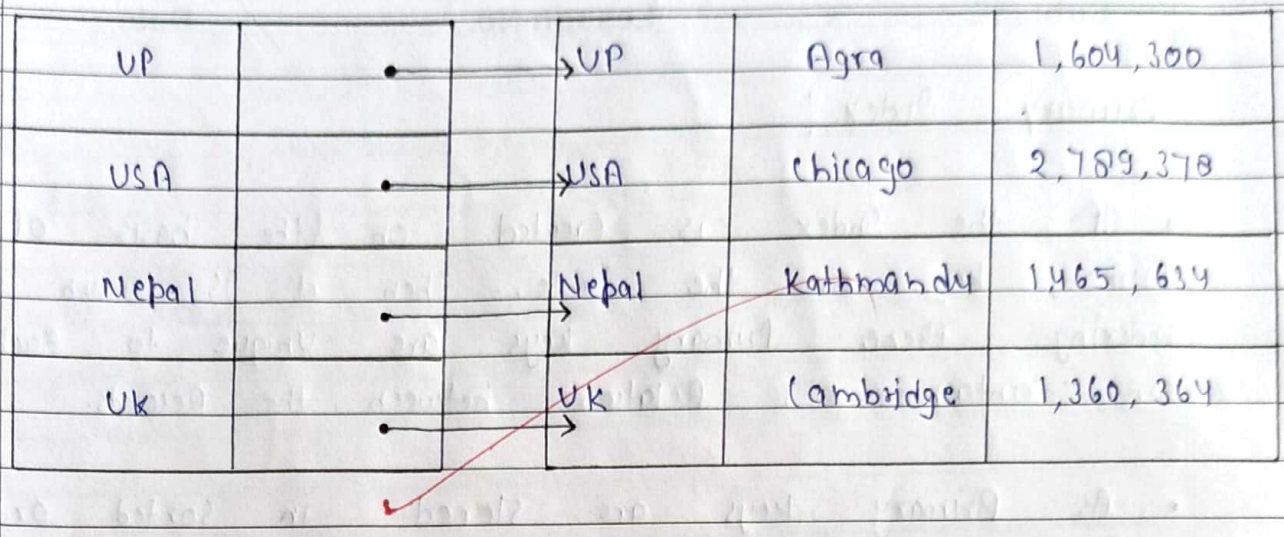
1. Dense index
2. Sparse index

1. Dense Index :-

- The dense index Contains an index Record for every search key value in the data file. it makes Searching faster.

- In this, the number of Records in the index table is same as the number of Records in the main table.

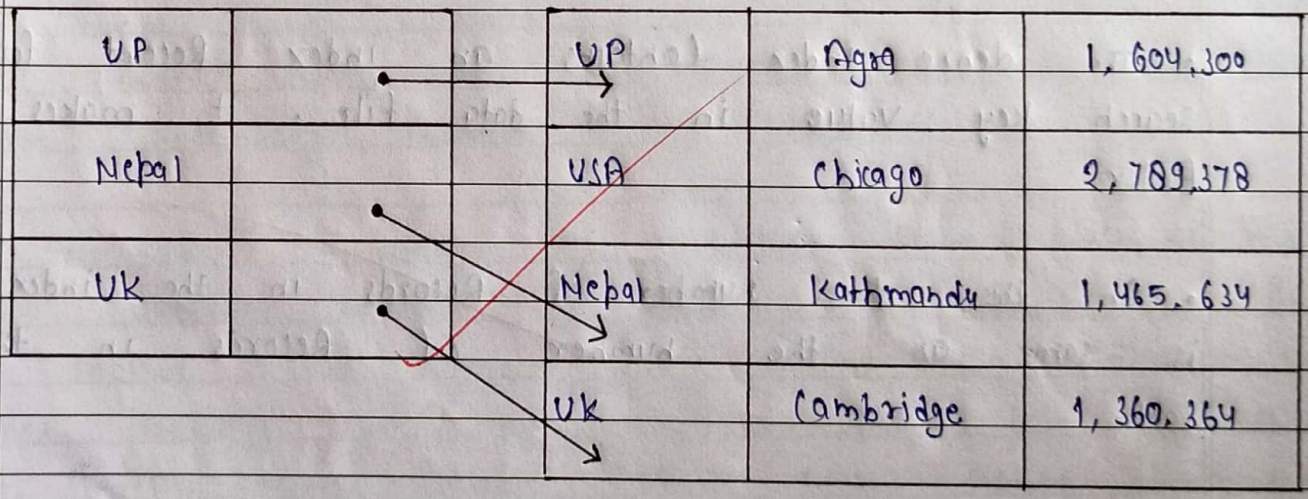
- It needs more space to store index Record itself. The index Records have the search key and a Pointer to the actual Record on the disk.



UP	UP	Agra	1,604,300
USA	USA	Chicago	2,789,378
Nepal	Nepal	Kathmandu	1,465,634
UK	UK	Cambridge	1,360,364

2. Sparse index :-

- In the data file, index Record appears only for a few items. Each item Points to a block.
- In this, instead of Pointing to each Record in the main table, the index Points to the Records in the main table in a gap.



UP	UP	Agra	1,604,300
Nepal	USA	Chicago	2,789,378
UK	Nepal	Kathmandu	1,465,634
	UK	Cambridge	1,360,364

Clustering Index :-

- The Records with have similar characteristic are grouped, and indexes are created for these group.

- A clustered index can be defined as an ordered data file. 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.

Secondary Index :-

In the sparse indexing, as the size of the table grow the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. The the secondary memory searches the actual database on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

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.

Stored Procedure :-

A stored Procedure is a prepared SQL code that you can save, so the code can be Reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored Procedure, and then just call it to Execute it.

you can Also Pass Parameters to a stored Procedure, so that the Stored Procedure can act based on the Parameter value (s) that is Passed.

Stored Procedure Syntax :-

```
CREATE PROCEDURE Procedure_name
AS
sql_statement
GO;
```

Execute a Stored Procedure :-

```
EXEC Procedure_name;
```

Stored Procedure to Example :-

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
```

Stored Procedure with one Parameter :-

The following SQL statement creates a stored Procedure that Selects Customers from a Particular city from the "Customers" table :

Example-

```
CREATE PROCEDURE SelectAllCustomers
@City nvarchar(30)
AS
SELECT * FROM Customers WHERE City = @City
GO;
```

Stored Procedure with multiple Parameters :-

Setting up multiple Parameter is very easy. Just list each Parameter and the datatype seperated by a comma as show below.

The following SQL statement creats a stored Procedure that Select Customers from a Particular city with a Particular Postal Code from the "Customers" table :

Example-

```
CREATE PROCEDURE SelectAllCustomers
@City nvarchar(30), @PostalCode nvarchar(10)
AS
SELECT * FROM Customers WHERE City =
@City AND PostalCode = @PostalCode
GO;
```

Cursor in SQL :-

- A machanism to navigate tuple - by - tuple over a Relation.
- Typically used inside triggers, stored Procedures.
- When we execute a Query, a Relation is Returned
- It is stored in Private work area for the Query.
- Cursor is a Pointer to this area.
- Move the cursor to Navigate over the tuples.
- Enables users to loop around a sleetion of data.
- use Complex action which would not be feasible in standard SQL Seleetion Queries.

Syntax for Cursors :-

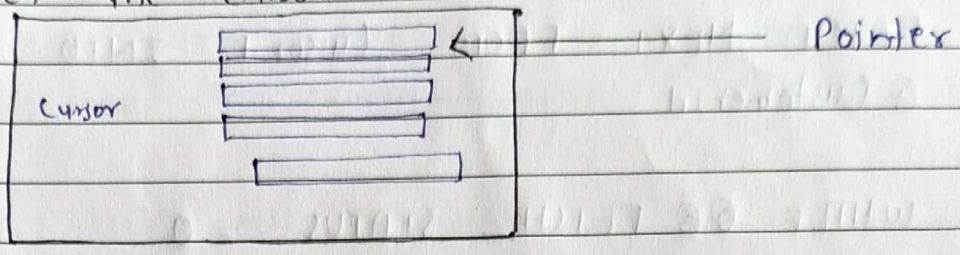
- Declare as a variable in the same way as standard variables.
- Identified as Cursor type SQL included

E.g.

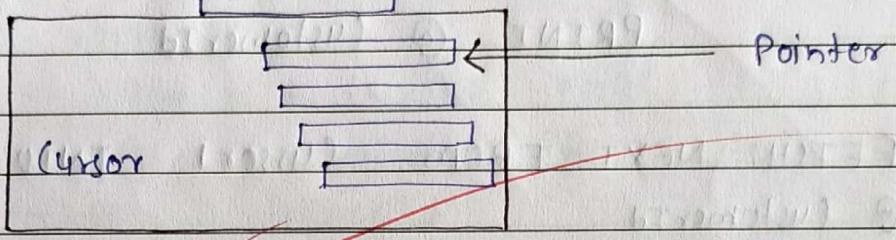
```
Cursor Cur_Emp is
Select Emp_id, Surname name, grade, Salary
from Employee
where grade is true;
```

Controlling Cursor :-

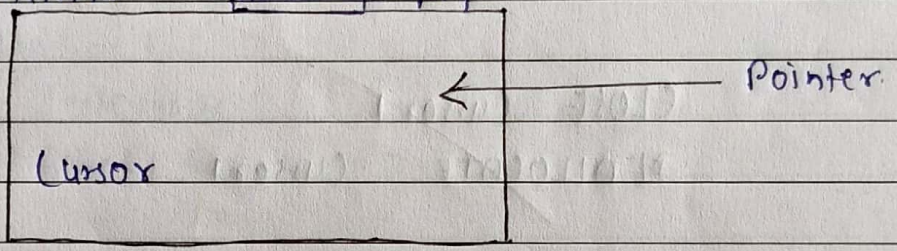
Open the Cursor.



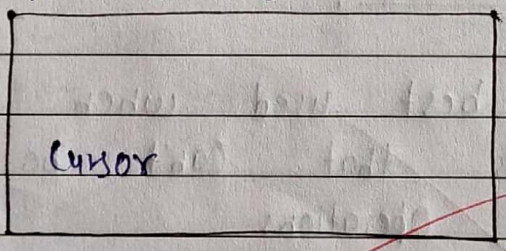
Fetch a row from the cursor.



Continue Until Empty.



Close the Cursor.



Example of Cursor :-

```
DECLARE @CustomerId INT

DECLARE Cursor1 CURSOR READ_ONLY
FOR
SELECT CustomerId
```

```
FROM Customers

OPEN Cursor1

FETCH NEXT FROM Cursor1 INTO
@CustomerId

WHILE @@ FETCH_ STATUS = 0
BEGIN
        PRINT @ CustomerId

FETCH NEXT FROM Cursor1 INTO
@ CustomerId
END

        CLOSE Cursor1
        DEALLOCATE Cursor1
```

Advantage of Cursor :-

- Cursor are best used when Performing row-by-row Operations that can't be accomplished with set-based Operations.
- Quick and dirty.

Disadvantage of Cursor :-

- A facter affecting Cursor speed is the Number of rows and Columns brought into the Cursor. Time how long it takes to Open your Cursor and fetch Statements.
- Speed and Performance issues.

Cursor Example in mySQL;-

```
DELIMITER $
DROP PROCEDURE IF EXISTS showCourseLecturesAlt $
CREATE PROCEDURE showCourseLecturesAlt (IN courseId INT)
BEGIN
  DECLARE lectSubject VARCHAR (128);
  DECLARE lectNum INT (2);
  DECLARE finishedFlag INT;
  DECLARE lectCursor CURSOR FOR
  SELECT Subject, num_lecture FROM lecture WHERE
  Course_lecture = courseId;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET
  finishedFlag = 1;
  OPEN lectCursor;
  SET finished Flag = 0;
  REPEAT
  FETCH lectCursor INFO lectSubject, lectNum;
  IF (finishedFlag = 0) THEN
     SELECT lectNum AS '.....', lectSubject AS
                                '.....';
  END IF;
  UNTIL (FinishedFlag = 1)
  END REPEAT;
  CLOSE lectCursor;
  END $
  DELIMITER;
```

Triggers :-

- Triggers are very Similar to stored procedures One big difference is: Trigger can not be manually Executed.

- They only execute in Response to a user Action, like an Insert is called when an Events occures in the table.

- Are used to: check the validity of the data that are inserted into a table.

- To Calculate derived values, to maintain the Sign in logs and the insertions in a table, Implementation Cost -> time Consuming.

Basic Commands :-

- Creation
- CREATE TRIGGER
- DELETE
- DROP TRIGGER < trigger Name >
- SHOW TRIGGER Code
- SHOW CREATE TRIGGER < trigger name >
- Show list of Created triggers
- SHOW TRIGGER
- Call / Execute a trigger
- We Cannot Call a trigger, like a Procedure
- It is Executed when an Event happens Call it.

Designing a trigger :-

- To design a trigger we have to determine
- On which table it would be applied
- With what event it will be linked
- E.g. INSERT, UPDATE, DELETE
- When it will be executed
- Before the event
- After the event
- Its functionality
- At the main body of the trigger we can write SQL code.

Trigger Creation :-

```
CREATE TRIGGER trigger_name trigger_time trigger_event
ON Table_name
FOR EACH ROW trigger_body
```

- trigger_name
- trigger_time -> when it will be executed (after or before)
- trigger_event -> the connected event
- ON table_Name. -> the table is belongs to
- FOR EACH ROW -> it will be executed for each row.
- trigger_body -> the SQL Scripts

Trigger Events :-

- The Triggers Events could be
- INSERT
- UPDATE
- DELETE
- TRUNCATE & DROP don't call a trigger.
- We could have at max 6 triggers in every table.

Good

10/06/22