Indexes

Indexes are special data structures associated with tables or views that help speed up the query.

- **What is Index?**
  - An index is a pointer to data in a table.
  - An index in a database is very similar to an index in the back of a book.
  - An index helps to speed up SELECT queries and WHERE clauses.
  - Indexes can be created or dropped with no effect on the data.

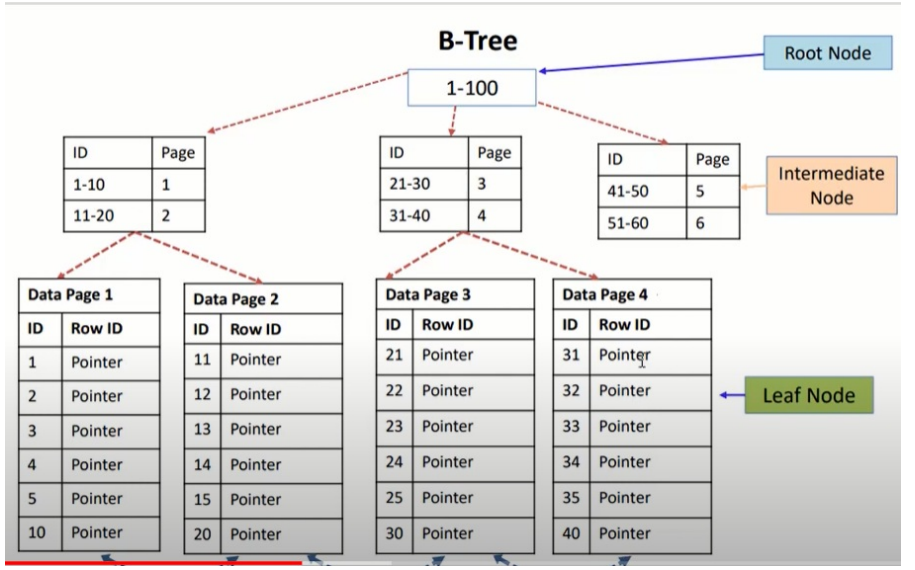
Types of Indexes

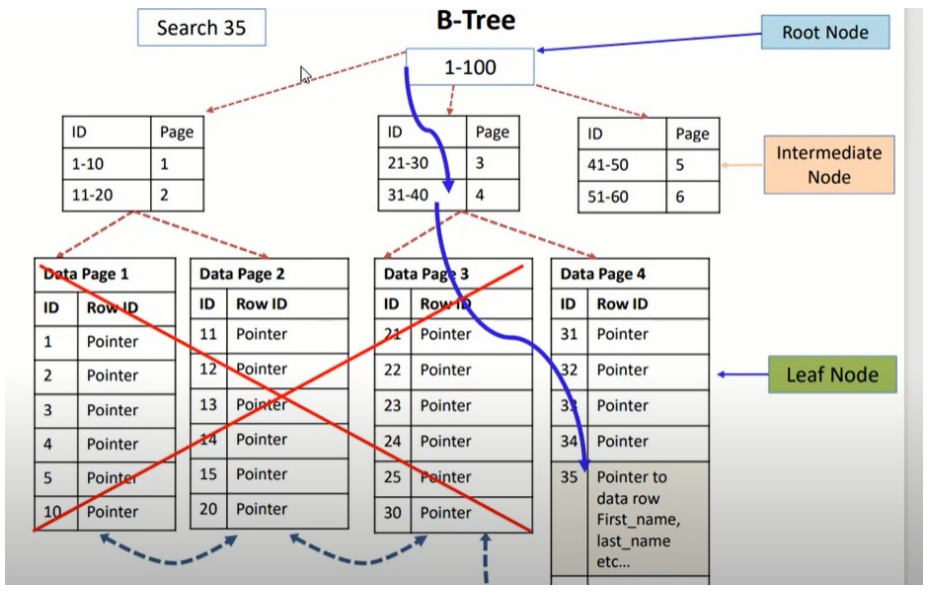
Clustered index

- Each table has **only one clustered index** because data rows can be only sorted in one order.
- A clustered index is a special index which physically orders the data according to the indexed columns.
- The leaf nodes of the index store the data for the rest of the columns in the table.

Non-clustered index

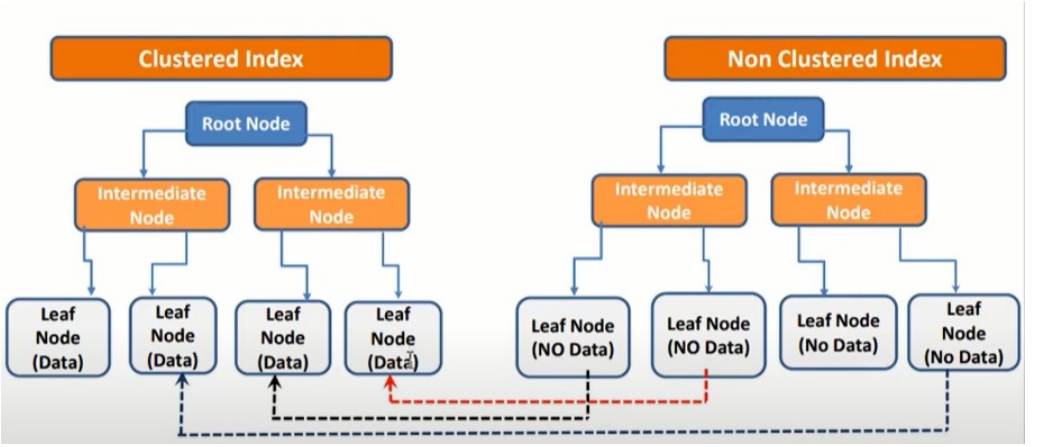
- A table may have one or more non clustered.
- A Non Clustered index is just like the index of a book.
- It points back to the actual page that contains the data. (In other words, it points back to the clustered index)





Non Clustered index

- A non-clustered index doesn't sort the physical data inside the table.
- In fact, a non-clustered index is stored at one place and table data is stored in another place.
- This is similar to a textbook where the book content is located in one place and the index is located in another.
- This allows for more than one non-clustered index per table.



SQL Server unique index

- A **unique index** ensures the index key columns do not contain any duplicate values.
- A unique index may consist of one or many columns.
- A unique index can be **clustered** or **non-clustered**.

```
SQLQuery1.sql - PCI...(PCI01\admin (59))*   SQLQuery2.sql - PCI...(PCI01\admin (52))*
CREATE CLUSTERED INDEX emp_idx
ON employee(emp_id ASC)

--After Indexing
select * from employee where Emp_ID=10000000
```

Where to Apply Index

- Indexes are meant to speed up the performance of a database, so use indexing whenever it significantly improves the performance of your database.
- Check query and find reason for slow performance.
- Find column in query which is used frequently for searching.

Disadvantages of Indexing

- In case of update(change in indexed column) and delete a record, the database might need to move the entire row into a new position to keep the rows in sorted order.

Cursor is a Temporary Memory or Temporary Work Station. It is Allocated by Database Server at the Time of Performing DML(Data Manipulation Language) operations on Table by User. Cursors are used to store Database Tables.

There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors.

These are explained as following below.

Implicit Cursors:

Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.

Explicit Cursors :

Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

```
## DECLARE s1 CURSOR FOR SELECT * FROM studDetails
```

1. OPEN s1name

```
FETCH FIRST FROM s1
FETCH LAST FROM s1
FETCH NEXT FROM s1
FETCH PRIOR FROM s1
FETCH ABSOLUTE 7 FROM s1
FETCH RELATIVE -2 FROM s1
```

2. **Clo** LOSE cursor_name

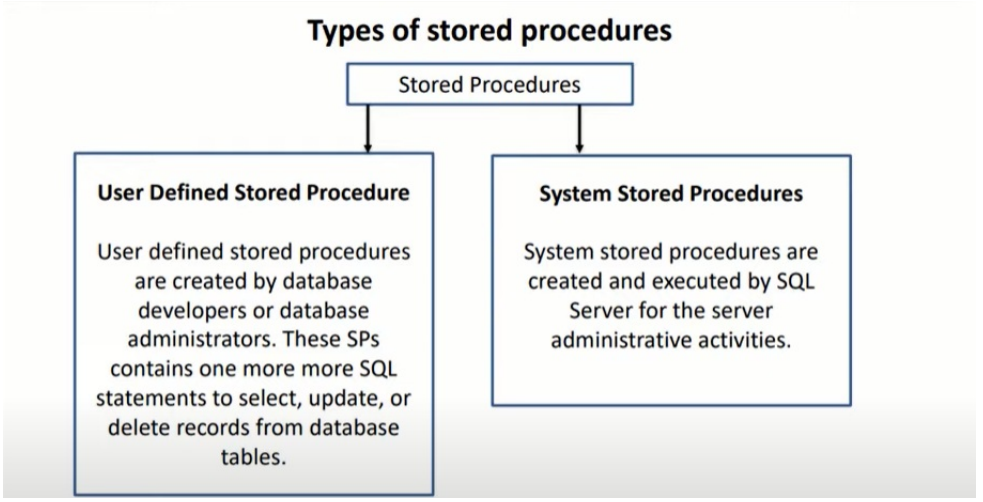
```
CLOSE s1

DEALLOCATE s1
```

1. View :
A view is a virtual table that not actually exist in the database but it can be produced upon request by a particular user. A view is an object that gives the user a logical view of data from a base table we can restrict to what user can view by allowing them to see an only necessary column from the table and hide the other database details. View also permits users to access data according to their requirements, so the same data can be access by a different user in a different way according to their needs.

2. Cursor :
A cursor is a temporary work area created in memory for processing and storing the information related to an SQL statement when it is executed. The temporary work area is used to store the data retrieved from the database and manipulate data according to need. It contains all the necessary information on data access by the select statement. It can hold a set of rows called active set but can access only a single row at a time. There are two different types of cursors –
1. Implicit Cursor
2. Explicit Cursor

4.	There are two types of view i.e. Simple View (created from single table) and Complex Cursor (created from multiple tables).	Cursor has two types i.e. Implicit Cursor (pre-defined) and Explicit Cursor(user defined).
5.	View is a database object similar to table so it can be used with both SQL and PL/SQL.	Cursor is defined and used within the block of stored procedure which means it can be only used with PL/SQL.
6.	General Syntax of Creating View : CREATE VIEW "VIEW_NAME" AS "SQL Statement";	General Syntax of Creating View : CURSOR cursor_name IS select_statement;



- A stored procedure is a precompiled set of one or more SQL statements which perform some specific task.
- A stored procedure should be executed stand alone using EXEC
- A stored procedure can return multiple parameters
- A stored procedure can be used to implement transact

```
GO
--HOW TO CREATE STORED PROCEDURE
Create Proc spDepartList
AS
BEGIN
select * from employee where deptID=1;
END


spDepartList
Execute spDepartList
EXEC spDepartList
```

```
--PARAMETERS in  STORED PROCEDURE
--PARAMETER ARE TWO TYPES INPUT PARAMETER & OUTPUT paramater
Alter Proc spDepartList
@deptt_id   int,
@emp_name   Varchar(100)
AS
BEGIN
    Select * from employee where deptID=@deptt_id;
    Select * from employee where empName=@emp_name
END

spDepartList @emp_name='joshef',@deptt_id=2
```

```
Use MyDatabase
GO
--OUTOUT PARAMETER
CREATE PROC spAddDigit
@Num1   INT,
@Num2   INT,
@Result INT OUTPUT
AS
BEGIN
    SET @Result=@Num1+@Num2;
END

Declare @EId INT
EXEC spAddDigit 23,27,@EId OUTPUT;
SELECT @EId;
```

➢ 2nd Highest Salary

Main Query

SELECT MAX(SALARY) FROM EMPLOYEE WHERE SALARY <

7666

SELECT MAX(SALARY) FROM EMPLOYEE WHERE SALARY

Less than 7666=6000

100 % Results Messages

	emp_id	salary
1	1	3000
2	2	2345
3	3	7666
4	4	3000

Using Set Operator Except in sub query

➢ 2nd Highest Salary

Main Query

SELECT MAX(SALARY) FROM EMPLOYEE WHERE SALARY IN

(SELECT salary FROM employee EXCEPT SELECT MAX(salary) FROM employee);

Results Messages

	salary
1	2345
2	3000
3	5000
4	6000

100 % Results Messages

	emp_id	salary
1	1	3000
2	2	2345
3	3	7666
4	4	3000
5	5	7666

```
Use MyDatabase
GO
--2nd Highest salary Using Corelated sub query
SELECT Salary FROM Employee pq WHERE 2=
    (SELECT COUNT(DISTINCT Salary)
    FROM Employee cq WHERE pq.Salary<=cq.Salary);
```

column salary(int, null)

Using Correlated sub query

➢ 2nd Highest Salary

```
SELECT  Salary FROM Employee pq WHERE 2=(SELECT COUNT(DISTINCT Salary)
FROM Employee cq WHERE pq.Salary<=cq.Salary);
```

Step-2

Employee pq

Emp_id	salary
1	3000
2	2345
3	7666
4	3000

Less than

Employee cq

Emp_id	salary
1	3000
2	2345
3	7666
4	3000

5th Highest Salary

Distinct Count=5

Using CTE-Common Type Expression with DENSE_RANK() function

- The CTE is preferred to use as an alternative to a Sub query/View.
- A CTE (Common Table Expression) defines a temporary result set which you can then use in a SELECT statement. but they are not exactly as temporary table or table variable.
- This function returns the rank of each row within a result set partition, with no gaps in the ranking values...

```
Use MyDatabase
GO
WITH ctResult AS
(
    SELECT emp_id, SALARY, DENSE_RANK() OVER (ORDER BY SALARY DESC) AS SalaryRank
    FROM EMPLOYEE
)
SELECT TOP 1 emp_id,SALARY FROM ctResult WHERE SalaryRank = 3;
```