Difference Between Clustered and Non-Clustered Index

In SQL Server, indexing plays a crucial role in enhancing data retrieval efficiency. Indexes are disk-based structures linked to tables or views, aiding in quicker row retrieval by organizing data in a structured manner. Two primary types of indexes in SQL Server are Clustered and Non-Clustered Indexes.

An index is a disk-based structure linked to a table or view that facilitates quicker row retrieval. A table or view's table or view's columns are used to create keys in an index. These keys are kept in a structure (B-tree) that enables SQL Server to quickly and effectively locate the row or rows that correspond to the key values.

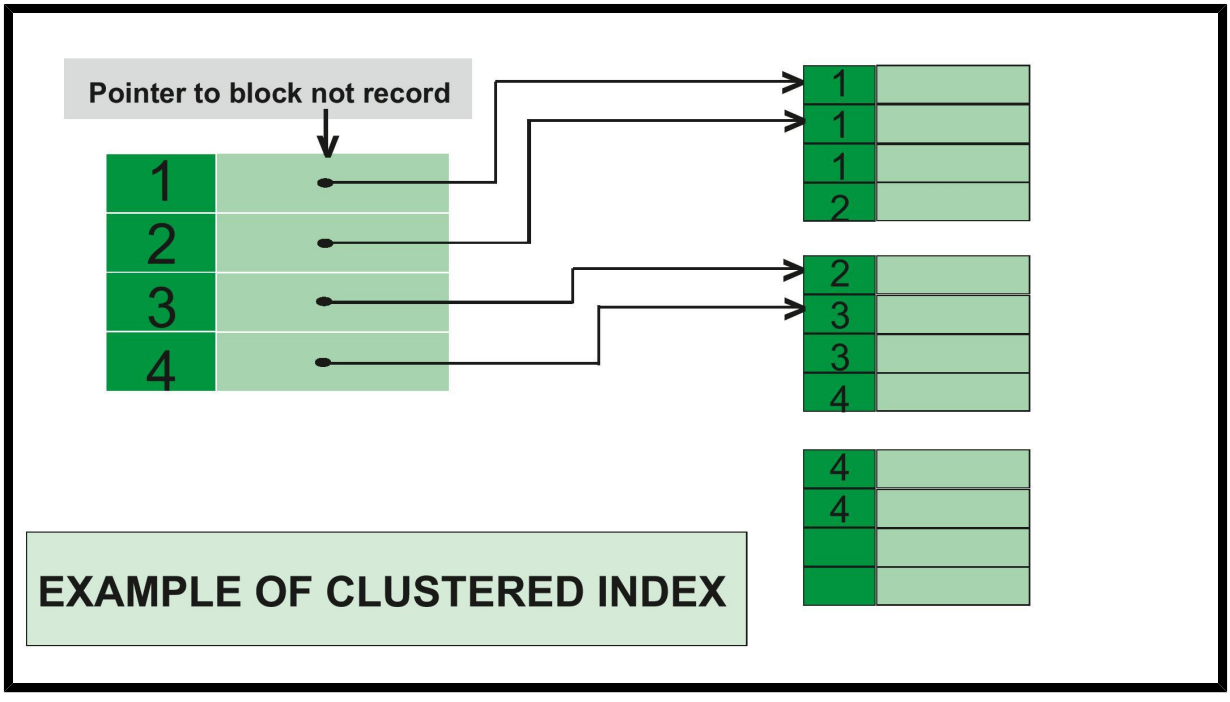
Clustered Index

A clustered index is created only when both the following conditions are satisfied:

- **The data or file that you are moving into secondary memory should be in sequential or sorted order** − This ensures efficient storage and retrieval.
- **There should be a key value** − The key value must be unique and cannot have repeated values to maintain data integrity and facilitate effective indexing.

Whenever you apply clustered indexing in a table, it will perform sorting in that table only. You can create only one clustered index in a table like a primary key. A clustered index is as same as a dictionary where the data is arranged in alphabetical order.

In a clustered index, the index contains a pointer to block but not direct data.



Example of Clustered Index

Example of Clustered Index

If you apply the primary key to any column, then automatically it will become a clustered index.

Create Table

```
-- Create the Student table
CREATE TABLE Student (
    Roll_No INT PRIMARY KEY,
    Name VARCHAR(50),
    Gender VARCHAR(30),
    Mob_No BIGINT
);
-- Insert data into the Student table
INSERT INTO Student
VALUES
    (4, 'ankita', 'female', 9876543210),
    (3, 'anita', 'female', 9675432890),
    (5, 'mahima', 'female', 8976453201);
```

In this example, Roll no is a primary key, it will automatically act as a clustered index. The output of this code will produce in increasing order of roll no.

Output

Roll_No	Name	Gender	Mob_No
4	ankita	female	9876543210
3	anita	female	9675432890
5	mahima	female	8976453201

Student table

You can have only one clustered index in one table, but you can have one clustered index on multiple columns, and that type of index is called a composite index.

Here, the Roll_No column serves as the primary key, automatically becoming the clustered index. The output of querying this table will present data in ascending order of Roll_No.

Key Differences

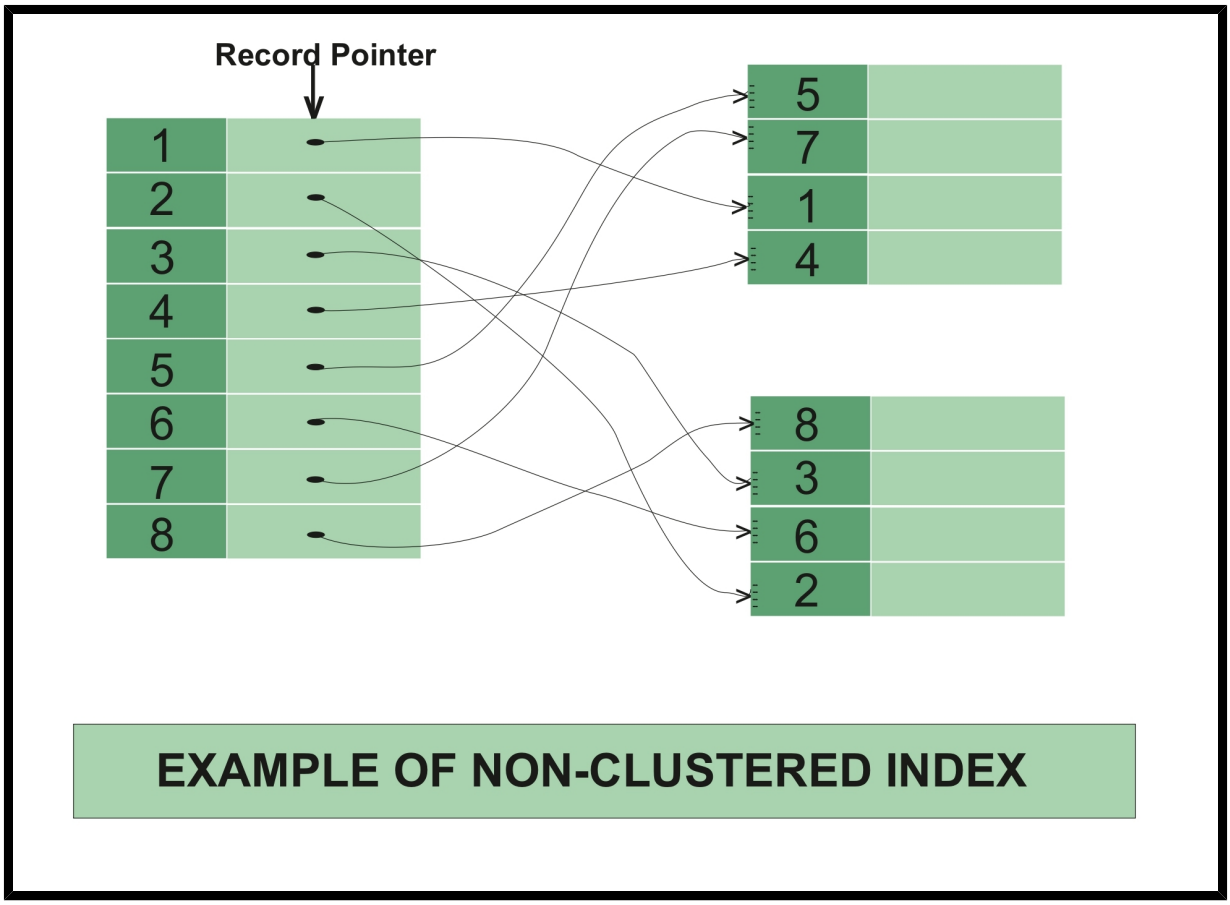
- **Only one clustered index is allowed per table** − A table can have only one clustered index, as it defines the physical order of the data rows.
- **The clustered index directly affects the physical ordering of data** − It determines the sequence in which data is stored on disk, affecting data retrieval performance.
- **It offers faster retrieval but may slow down insert and update operations** − While clustered indexes speed up data retrieval, they can impact the performance of data modification operations due to the need to maintain the physical order.

Non-Clustered Index

The non-Clustered Index is similar to the index of a book. The index of a book consists of a chapter name and page number, if you want to read any topic or chapter then you can directly go to that page by using the index of that book. No need to go through each and every page of a book.

The data is stored in one place, and the index is stored in another place. Since the data and non-clustered index is stored separately, then you can have multiple non-clustered indexes in a table.

In a non-clustered index, the index contains the pointer to data.



Example of Non Clustered Index

Example of Non-Clustered Index

The given code creates a table “Student” with columns “Roll_No”, “Name”, “Gender”, and “Mob_No”. The primary key is defined on the “Roll_No” column. Three rows are inserted into the “Student” table with different values for the columns. Finally, a nonclustered index “NIX_FTE_Name” is created on the “Name” column in ascending order.

The “CREATE TABLE” statement is used to create a new table “Student” with four columns “Roll_No”, “Name”, “Gender”, and “Mob_No”. The “Roll_No” column is defined as the primary key of the table.

The “INSERT INTO” statements are used to insert three rows of data into the “Student” table. Each row contains values for all the columns of the table. The first row has a roll number of 4, the name of “Afzal”, the gender of “male”, and a mobile number of 9876543210. The second row has a roll number of 3, the name of “Sudhir”, the gender of “male”, and a mobile number of 9675432890. The third row has a roll number of 5, name of “zoya”, the gender of “female”, and a mobile number of 8976453201.

Query

```
-- Create the Student table
CREATE TABLE Student (
    Roll_No INT PRIMARY KEY,
```

```
    Name VARCHAR(50),
    Gender VARCHAR(30),
    Mob_No BIGINT
);
-- Insert data into the Student table
INSERT INTO Student
VALUES
    (4, 'afzal', 'male', 9876543210),
    (3, 'sudhir', 'male', 9675432890),
    (5, 'zoya', 'female', 8976453201);
-- Create a non-clustered index on Name
CREATE NONCLUSTERED INDEX NIX_FTE_Name
ON Student
() (Name ASC);
```

Here, roll no is a primary key, hence there is automatically a clustered index. If we want to apply a non-clustered index in the NAME column (in ascending order), then a new table will be created for that column.

Output

Roll_No	Name	Gender	Mob_No
4	afzal	male	9876543210
3	sudhir	male	9675432890
5	zoya	female	8976453201

Student table

The row address is used because, if someone wants to search the data for Sudhir, then by using the row address he/she will directly go to that row address and can fetch the data directly.

In this example, the Name column is indexed in ascending order, allowing for efficient retrieval based on names.

Key Differences

- **Multiple non-clustered indexes are allowed per table** − A table can have multiple non-clustered indexes, each providing a different way to access the data.
- **Non-clustered indexes store data pointers, not the data itself** − These indexes contain references to the data rows rather than the actual data, allowing quick lookups without affecting the physical storage of the data.
- **They offer flexibility but may result in slower retrieval compared to clustered indexes** − While non-clustered indexes provide additional access paths, they can be less efficient for retrieval compared to clustered indexes, as they require an additional lookup to retrieve the actual data.

Difference Between Clustered and Non-Clustered Index

CLUSTERED INDEX	NON-CLUSTERED INDEX
A clustered index is faster.	A non-clustered index is slower.

The clustered index requires less memory for operations.	A non-clustered index requires more memory for operations.
In a clustered index, the clustered index is the main data.	In a non-clustered index, the index is a copy of data.
A table can have only one clustered index.	A table can have multiple non-clustered indexes.
The clustered index has the inherent ability to store data on the disk.	A non-clustered index does not have the inherent ability to store data on the disk.
Clustered index stores pointers to blocks, not data.	The non-clustered index stores both the value and a pointer to the actual row that holds the data.
In a clustered index, leaf nodes are actual data itself.	In a non-clustered index, leaf nodes are not the actual data itself; they only contain included columns.
In a clustered index, the clustered key defines the order of data within a table.	In a non-clustered index, the index key defines the order of data within the index.
A clustered index is a type of index in which table records are physically reordered to match the index.	A non-clustered index is a special type of index in which the logical order of the index does not match the physical stored order of the rows on the disk.
The size of the primary clustered index is large.	The size of the non-clustered index is comparatively smaller.
Primary keys of the table by default are clustered indexes.	The composite key, when used with unique constraints of the table, acts as a non-clustered index.

Conclusion

In Conclusion, the decision between clustered and non-clustered indexes primarily depends on the particular requirements of your database.Choosing between clustered and non-clustered indexes depends on the specific requirements of your database:

- **Clustered indexes are ideal for range queries and static data** − They are well-suited for queries that involve ranges of values and are efficient for static datasets where the data is not frequently modified.
- **Non-clustered indexes are suitable for optimizing various query types and dynamic data environments** − They provide flexibility for a wide range of query types and are effective in environments where data is frequently updated or where multiple access paths are needed.

Understanding these indexing techniques empowers database administrators to optimize data retrieval and performance effectively.

Index Clustering

- **Perfect for tables where range queries, in particular, place a high value on data retrieval efficiency** − Clustered indexes are especially effective in scenarios where queries require efficient access to ranges of data.

- **Ideal for tables with few updates or relatively static data** − Clustered indexes work best with data that does not change frequently, as the physical reordering of data can slow down insert and update operations.

Non-Clustered index

- **Allows for the optimization of various query types without changing the data's physical order on disk** − Non-clustered indexes enhance query performance by creating separate structures for data retrieval, without altering the actual data storage layout.
- **Ideal for tables where data changes often** − Non-clustered indexes are well-suited for tables with frequent inserts and updates, as they generally involve less overhead compared to clustered indexes for maintaining data order.