Index

A note by
Shah Md. Arshad Rahman Ziban

Index

Definition

Indexing is a method that allows for quick access to data in a database. It creates a data structure that enables the database management system (DBMS) to find records without scanning the entire table.

Purpose of Indexes

Efficiency in Data Retrieval: Without indexing, the database management system (DBMS) must scan the entire table to find specific records, which can be time-consuming, especially for large datasets.

Faster Query Execution: Indexes allow the DBMS to quickly locate the required data without having to examine every row in the table, significantly speeding up query execution.

Reduced Resource Usage: By minimizing the amount of data that needs to be scanned, indexing reduces the load on system resources, leading to better overall performance.

Example:

Consider a table named `AdmissionInfo` that contains student records, including their names and CGPA (Cumulative Grade Point Average). If you want to find students with a CGPA between 3.50 and 4.00, the SQL query would look like this:

```
SELECT Name, CGPA FROM AdmissionInfo WHERE CGPA BETWEEN 3.50 AND 4.00;
```

Without Indexing:

- If the `CGPA` column is not indexed, the DBMS will perform a full table scan, checking each record in the `AdmissionInfo` table to see if the CGPA falls within the specified range. This can be very slow, especially if the table has thousands of records.

With Indexing:

If an index is created on the CGPA column using the following SQL command:

```
CREATE INDEX NewAdmissionInfo ON AdmissionInfo(CGPA);
```

Now, when the same query is executed, the DBMS can use the index to quickly locate the records that meet the criteria without scanning the entire table. This results in much faster query execution and improved system efficiency.

Conclusion:

Indexing is essential for optimizing data retrieval in databases. It allows for quick access to records, reduces the time taken for queries, and enhances the overall performance of the database system.

Types of Index

There are two types of index:

1. Clustered Index
2. Non-Clustered Index

Clustered Index

A Clustered Index can be created in the following ways:

By Defining a Primary Key Using the CREATE INDEX Statement

By Defining a Primary Key :

When a primary key is defined on a table, a clustered index is automatically created on that primary key. This is the most common method.

Example:

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,  -- This creates a clustered index on EmployeeID
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(50)
);
```

Inserting Data:

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Email)
VALUES (106, 'John', 'Doe', 'john.doe@example.com'),
       (103, 'Jane', 'Smith', 'jane.smith@example.com');
```

Output

EmployeeID	FirstName	LastName	Email
103	Jane	Smith	jane.smith@example.com
106	John	Doe	john.doe@example.com

Altering an Existing Table:

If you have an existing table without a clustered index, you can alter the table to add a clustered index using an ALTER TABLE statement:

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name PRIMARY KEY CLUSTERED (column_name);
```

Using the CREATE INDEX Statement:

You can manually create a clustered index on a table using the SQL command:

```
CREATE CLUSTERED INDEX index_name ON table_name(column_name);
```

When working with clustered indexes in a database, it's important to remember that a table can only have one clustered index at a time. If you attempt to create a second clustered index on the same table, you will encounter an error. To create a new clustered index, you must first drop the existing one.

Steps to Drop an Existing Clustered Index

Identify the Existing Clustered Index: Before dropping an index, you need to know its name. You can us the sp_helpindex function to list all indexes on a table.

```
EXEC sp_helpindex 'YourTableName';
```

Drop the Existing Clustered Index: Use the DROP INDEX statement to remove the existing clustered index. The syntax is as follows:

```
DROP INDEX IndexName ON YourTableName;
```

Create a New Clustered Index: After successfully dropping the existing clustered index, you can create a new one. Use the DROP INDEX statement to remove the existing clustered index. The syntax is as follows:

```
CREATE CLUSTERED INDEX NewIndexName ON YourTableName(ColumnName);
```

Summary

A table can only have one clustered index. To create a new clustered index, you must first drop the existing one. - Use the `DROP INDEX` statement to remove the existing clustered index. After dropping, you can create a new clustered index as needed.

Non Clustered Index

A Non-Clustered Index is a type of index in a database that improves the speed of data retrieval operations on a table. Unlike a clustered index, which determines the physical order of data in a table, a non-clustered index maintains a separate structure that points to the actual data rows. Here's a detailed overview of non-clustered indexes:

Key Characteristics of Non-Clustered Indexes

Separate Storage: Non-clustered indexes are stored separately from the actual data rows. This means that the index contains pointers to the data rather than the data itself.

Multiple Non-Clustered Indexes: A table can have multiple non-clustered indexes. This allows for efficient querying on different columns without affecting the physical order of the data.

Structure: Non-clustered indexes typically use a B-tree structure, which allows for efficient searching, inserting, and deleting operations.

Performance: Non-clustered indexes can significantly improve query performance, especially for read-heavy operations.

Use of Pointers: Each entry in a non-clustered index contains the indexed column value and a pointer (row locator) to the actual row in the table. This allows the database engine to quickly locate the data.

Creating a Non-Clustered Index

To create a non-clustered index, you can use the CREATE INDEX statement. Here's the syntax:

```
CREATE NONCLUSTERED INDEX IndexName
ON TableName (ColumnName1, ColumnName2, ...);
```

Dropping a Non-Clustered Index

```
DROP INDEX IndexName ON YourTableName;
```

Show Info About Index

```
SELECT name AS IndexName, type_desc AS IndexType
FROM sys.indexes
WHERE object_id = OBJECT_ID('Table_Name');
```

This output shows the index names and their types, helping us quickly identify all indexes associated with the table which we created and determine if they are clustered or non-clustered indexes.

Difference between clustered and non clustered index

Feature	Clustered Index	Non-Clustered Index
Definition	Determines physical data order	Separate structure with pointers
Physical Storage	Data and index are the same	Data stored separately
Number of Indexes	Only one per table	Multiple allowed
Performance	Fast for range queries	Fast for lookups, may require extra lookups
Storage Space	No additional space required	Requires additional storage
Use Cases	Best for primary keys, range queries	Best for foreign keys, search columns