Lecture Notes 8

Database Indexing allows us to cut down the number of rows/records that need to be examined when a select query with a where clause is executed.

Let’s try to understand what happens when a select query is executed on a database table without an index.

Suppose we have an Employee table, with fields ‘Employee\_Id’ and ‘Employee\_Name’. We want to execute the following select query on this table:

select \* from Employee where Employee\_Name = 'abc';

Data structures for indexing

1. B-trees
2. Hash Tables
3. R-tree
4. Bitmap Index

**So, what is the cost of having a database index?**

The first thing to note is that the index takes up additional space, so the larger the table, the bigger the index. Every time you perform an add, delete, or update operation, the same operation will need to be performed on the index as well.

**Creating an index**

The following snippet shows how to create an index on a single column and on multiple columns.

CREATE INDEX name\_index ON Employee (Employee\_Name)

CREATE INDEX name\_index ON Employee (Employee\_Name, Employee\_Age)

Partitioning Methods:

1. **Horizontal Partitioning**
2. **Vertical Partitioning**
3. **Directory-Based Partitioning**

**Partitioning Criteria:**

**Key or Hash-based Partitioning**

**List partitioning**

**Round-robin partitioning**

**Common Problems of Data Partitioning:**

**Joins and Denormalization**