**Index**

Used to find data from the table quickly. If index is not there then for each query, it needs to scan each and every row in the table.

How it is fast?

It’s basically it forms a kind of B-tree (more than 2 child) or a variant like a B+ tree for a column where nodes and leafs will be made.

Creating an Index

On creating an index for a column (not a primary key) we will mapping row address to that column. So, B tree is formed for that column by either Ascending order or descending order with row address mapped. The index is stored in its own distinct storage structure, separate from the table's data.

So, When I perform a search based on an indexed column, the database uses the index to quickly locate the relevant rows.

Inserting a new row

In this case Indexed column will also have an update. Son tree rebalancing will happen. During this rebalance we should not allow read operation which may cause corruption. This will be prevented by locking

Types of Indices

* Clustered
* Non-Clustered
* Unique
* Filtered
* XML
* Full text
* Spatial
* ColumnStore
* Index with included column
* Index with computed column

**Clustered and NonClustered Index:**

Clustered index determines physical order of data. So, Primary key is automatically created as clustered index. We can have only one Clustered index in a table. Since Primary key will act an index, we will not map row address (row itself sorted based on primary key order)

Non clustered index can be created for other columns, so that we will be arranging it in form of tree, and we will map row address to that column.

**Unique Index:**

Column which is unique we can create Unique index. By default whenever we add a unique constraints for a column, Unique Non clustered index will be created.

**Composite Indexes:**

Indexes created on multiple columns can improve performance for complex queries.

