Clustered ,NonClustered Indexes and Index Design considerations

Clustered Index

- A clustered index stores the actual data rows at the leaf level of the index.
- that would mean that the entire row of data associated with the primary key value of 123 would be stored in that leaf node.
- An important characteristic of the clustered index is that the indexed values are sorted in either ascending or descending order.
- As a result, there can be only one clustered index on a table or view.
- In addition, data in a table is sorted only if a clustered index has been defined on a table.

NonClustered Index

- Unlike a clustered indexed, the leaf nodes of a nonclustered index contain only the values from the indexed columns and row locators that point to the actual data rows, rather than contain the data rows themselves.
- This means that the query engine must take an additional step in order to locate the actual data.
- A row locator's structure depends on whether it points to a clustered table or to a heap. If referencing a clustered table, the row locator points to the clustered index, using the value from the clustered index to navigate to the correct data row.
- If referencing a heap, the row locator points to the actual data row.
- Nonclustered indexes cannot be sorted like clustered indexes.
- SQL Server support up to 999 nonclustered indexes.
- This certainly doesn't mean you should create that many indexes. Indexes can both help and hinder performance.
- In addition to being able to create multiple nonclustered indexes on a table or view, you can also add included columns to your index

Index types based on configuration

- **Composite index:** An index that contains more than one column. In both SQL Server you can include up to 16 columns in an index, as long as the index doesn’t exceed the 900-byte limit. Both clustered and nonclustered indexes can be composite indexes.
- **Unique Index:** An index that ensures the uniqueness of each value in the indexed column. If the index is a composite, the uniqueness is enforced across the columns as a whole, not on the individual columns. For example, if you were to create an index on the FirstName and LastName columns in a table, the names together must be unique, but the individual names can be duplicated.
- A unique index is automatically created when you define a primary key or unique constraint
- **Primary key:** When you define a primary key constraint on one or more columns, SQL Server automatically creates a unique, clustered index if a clustered index does not already exist on the table or view.
- **Unique:** When you define a unique constraint, SQL Server automatically creates a unique, nonclustered index.
- **Covering index:** A type of index that includes all the columns that are needed to process a particular query. For example, your query might retrieve the FirstName and LastName columns from a table, based on a value in the ContactID column. You can create a covering index that includes all three columns.

Index design consideration

- As beneficial as indexes can be, they must be designed carefully.
- They can take up significant disk space, you don’t want to implement more indexes than necessary.
- Indexes are automatically updated when the data rows themselves are updated, which can lead to additional overhead and can affect performance.
- For tables that are heavily updated, use as few columns as possible in the index, and don’t over-index the tables.
- on small tables because the query engine might take longer to navigate the index than to perform a table scan.
- For clustered indexes, try to keep the length of the indexed columns as short as possible.
- Try to implement your clustered indexes on unique columns that do not permit null values.
- When possible, implement unique indexes.
- For composite indexes, take into consideration the order of the columns in the index definition.
- Try to insert or modify as many rows as possible in a single statement, rather than using multiple queries.
- Create nonclustered indexes on columns used frequently in your statement’s predicates and join conditions.

Create Index Syntax

- -- Create a nonclustered index on a table or view
- CREATE INDEX index1 ON schema1.table1 (column1);

- -- Create a clustered index on a table and use a 3-part name for the table
- CREATE CLUSTERED INDEX index1 ON database1.schema1.table1 (column1);

- -- Create a nonclustered index with a unique constraint
- -- on 3 columns and specify the sort order for each column
- CREATE UNIQUE INDEX index1 ON schema1.table1 (column1 DESC, column2 ASC, column3 DESC);