## Indexes in SQL

* Indexes are used by queries to find data from tables quickly. Indexes are created on tables and views. Index on a table or a view, is a very similar to an index that we find in a book.
* If you don’t have an index, and I ask you to locate a specific chapter in the book, you will have to look at every page starting from the first page of the book.
* On, the other hand, if you have the index, you lookup the page number of the chapter in the index and then directly go to that page number to locate the chapter.
* Obviously, the book index is helping to drastically reduce the time it takes to find the chapter.
* In the similar way, table and views indexes, can help the query to find data quickly.
* In fact, the existence of the right indexes, can drastically improve the performance of the query. If there is no index to help the query, then the query engine, checks every row in the table from the beginning to the end. This is called as Table scan. Table scan is bad for performance. The query engine (database engine) has to scan the whole table.
* When there is no index for the table then the query result in table scan.
* At the moment, the Employee table, does not have an index on SALARY column.
* SELECT \* FROM employee WHERE salary > 5000 AND salary < 7000;
* To find all employees, who have salary greater than 5000 and less than 7000, the query engine has to check each and every row in the table, resulting in a table scan, which can adversely affect the performance, especially if the table is large. Since there is no index, to help the query, the query engine performs an entire table scan.

### CREATING AN INDEX on a table

* CREATE INDEX IX\_tblName\_colName ON tblName (ColName ASC)
* The index stores salary of each employee, in the ascending order as shown below. The actual index may look-slightly different.
* Now, when the SQL server has to execute the same query, it has an index on the SALARY column to help this query. Salaries b/w the range of 5000 to 7000 are usually present at the bottom, since the salaries are arranged in an ascending order. SQL server picks up the row addresses from the index and directly fetch the records from the table, rather than scanning each row in the table. This is called as Index Seek.

### Different type of Indexes

1. Clustered
2. Non clustered
3. Unique
4. Filtered
5. XML
6. Full text
7. Spatial
8. Columnstore
9. Index with included columns
10. Index with computed columns

### Clustered index

* A clustered index determines the physical order of data in a table. For this reason, a table can have only one clustered index.
* CREATE TABLE [tablename]

(

[colname1] int Primary key,

…

)

* Note that colname1 is marked as primary key. Primary key, constraint create clustered indexes automatically if no clustered index is already exists on the table.
* To confirm execute: SHOW INDEXES FROM [tablename].
* A clustered index is analogous to a telephone directory, where the data is arranged by the last name. We just learnt that, a table can have only one clustered index. However, the index can contain multiple columns (a composite index), like the way a telephone directory is organized by last name and first name.
* Create a composite clustered index:

**CREATE CLUSTERED INDEX**

**IX\_EMP01\_P01F01**

**ON**

**EMP01 (P01F01 ASC, P01F02 DESC);**

### Non clustered index

* A non clustered index is analogous to textbook. The data is stored in one place, the index in another place. The index will have pointers to the storage location of the data.
* Since, the non-clustered index is stored separately from the actual data, a table can have more than one non-clustered index, just like how a book have an index by chapter at the beginning and another by common terms at the end.
* In the index itself, the data is stored in an ascending or descending order of the index key, which doesn’t in any way influence of data in the table.

### Clustered vs non-clustered index

* Only one clustered index per table, where as you can have more than one non-clustered index.
* Clustered index is faster than a non-clustered index, b/c, the non-clustered index has to refer back to the table, if the selected column is not present in the index.
* Clustered index determines the storage order of rows in the table, and hence doesn’t require additional disk space, but whereas a non-clustered index is stored separately from the table, additional storage space is required.

### Unique Index

* Unique index is used to enforce uniqueness of key values in the index.
* Note: By default, **PRIMARY** **KEY** constraint creates a **unique clustered index**.
* Primary key constraint bts uses unique index to enforce that constraint.
* UNIQUENESS is a property of an index, and both CLUSTERED and NON-CLUSTERED indexes cab be UNIQUE.
* **CREATE UNIQUE NonClustered INDEX**

**UIX\_EMP01\_P01F03**

**ON EMP01 (P01F03 ASC);**

* There are no major difference b/w a unique constraint and a unique index. In fact, when you add a unique constraint, a unique index gets created behind the scenes.
* So, there are two ways to create unique index:

1. By adding a unique constraint.
2. By using CREATE INDEX statement.

* When should you be creating a unique constraint over a unique index?

To make our intension clear, create a unique constraint, when data integrity is the objective. This makes the objective of the index very clear. In either cases, data is validated in the same manner, and the query optimizer does not differentiate b/w a unique index created by a unique constraint or manually created.

* Points to remember:
  1. By default, a primary key constraint, creates a unique clustered index, where as a UNIQUE constraint creates non-clustered index. These defaults can be changed if you wish to.
  2. A unique constraint or unique index cannot be created on an existing table, if the table contains duplicate values in the key columns. Obviously, to solve this, remove the key columns from the index definition or delete or update the duplicate values.

### Advantages of Indexes

1. SELECT statement with a where clause.
2. DELET and UPDATE statement.
3. ORDER BY clause
4. GROUP BY clause

### Disadvantages of indexes

* Additional disk space: Clustered index does not, require additional storage. Every non-clustered index require additional space as it is stored separately from the table. The amount of space required will depend on the size of the table, and the number and types of columns used in the index.
* Insert, update and delete queries can be slow: When DML (data manipulation language) statements (INSERT, UPDATE, DELETE) modifies data in the table, the data in all the indexes also needs to be updated. Indexes can help, to search and locate the rows, that we want to delete, but too many indexes to update can actually hurt the performance of data modification.
* What is covering query? If all the columns that you have requested in the SELECT clause of query, are present in the index, then there is no need to lookup in the table again. The required columns data can simply be returned from the index.
* A clustered index, always covers a query, since it contains all of the data in a table. A composite index is an index on two or more columns. Both clustered and non-clustered indexes can be composite indexes. To a certain extent, a composite index, can cover a query.

### Explain command insights

* Id – The no. of the query.
* Select\_type – the type of the SELECT statement being used.
* Table – the table for the output row.
* Type – the join type.
* Possible\_keys – which keys could be used.
* Key – the key used.
* Key\_len – the length of the key used.
* Ref – the columns compared to the index.
* Rows – how many rows will be examined.
* Extra – additional information about the query.

### Explain command

* Mysql explain statement can be used to explore the order that the optimizer chooses to process queries.
* The key piece of information is the product of the number of rows that will be assessed.