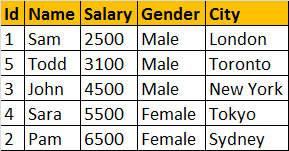
**Clustered and Non-Clustered indexes - Part 36**

[Please watch Part 35 - Indexes in SQL Server, before continuing with this session](http://csharp-video-tutorials.blogspot.com/2012/09/indexes-in-sql-server-part-35.html)  
  
**The following are the different types of indexes in SQL Server**  
1. Clustered  
2. Nonclustered  
3. Unique  
4. Filtered  
5. XML  
6. Full Text  
7. Spatial  
8. Columnstore  
9. Index with included columns  
10. Index on computed columns  
  
In this video session, we will talk about Clustered and Non-Clustered indexes.  
  
  
  
  
  
  
  
**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 tblEmployees table using the script below.**  
CREATE TABLE [tblEmployee]  
(  
 [Id] int Primary Key,  
 [Name] nvarchar(50),  
 [Salary] int,  
 [Gender] nvarchar(10),  
 [City] nvarchar(50)  
)  
  
Note that **Id**column is marked as **primary key**. Primary key, constraint create **clustered indexes automatically** if no clustered index already exists on the table and a nonclustered index is not specified when you create the PRIMARY KEY constraint.   
  
**To confirm this**, execute sp\_helpindex tblEmployee, which will show a unique clustered index created on the**Id** column.   
  
**Now execute the following insert queries**. Note that, the values for Id column are not in a sequential order.  
Insert into tblEmployee Values(3,'John',4500,'Male','New York')  
Insert into tblEmployee Values(1,'Sam',2500,'Male','London')  
Insert into tblEmployee Values(4,'Sara',5500,'Female','Tokyo')  
Insert into tblEmployee Values(5,'Todd',3100,'Male','Toronto')  
Insert into tblEmployee Values(2,'Pam',6500,'Female','Sydney')  
  
**Execute the following SELECT query**  
Select \* from tblEmployee  
  
**Inspite, of inserting the rows in a random order**, when we execute the select query we can see that all the rows in the table are arranged in an ascending order based on the Id column. This is because a clustered index determines the physical order of data in a table, and we have got a clustered index on the Id column.  
  
**Because of the fact that, a clustered index dictates the physical storage order** of the data in a table, a table can contain only one clustered index. If you take the example of **tblEmployee** table, the data is already arranged by the Id column, and if we try to create another clustered index on the **Name column**, the data needs to be rearranged based on the **NAME column**, which will affect the ordering of rows that's already done based on the ID column.  
  
**For this reason**, SQL server doesn't allow us to create more than one clustered index per table. The following SQL script, raises an error stating 'Cannot create more than one clustered index on table 'tblEmployee'. Drop the existing clustered index PK\_\_tblEmplo\_\_3214EC0706CD04F7 before creating another.'  
Create Clustered Index IX\_tblEmployee\_Name  
ON tblEmployee(Name)  
  
**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.  
  
**Let's now create a clustered index on 2 columns**. To do this we first have to drop the existing clustered index on the Id column.   
Drop index tblEmployee.PK\_\_tblEmplo\_\_3214EC070A9D95DB  
  
**When you execute this query**, you get an error message stating 'An explicit DROP INDEX is not allowed on index 'tblEmployee.PK\_\_tblEmplo\_\_3214EC070A9D95DB'. It is being used for PRIMARY KEY constraint enforcement.' We will talk about the role of unique index in the next session. To successfully delete the clustered index, right click on the index in the Object explorer window and select DELETE.  
  
**Now, execute the following CREATE INDEX query**, to create a composite clustered Index on the Gender and Salary columns.  
Create Clustered Index IX\_tblEmployee\_Gender\_Salary  
ON tblEmployee(Gender DESC, Salary ASC)  
  
**Now, if you issue a select query against this table** you should see the data physically arranged, FIRST by Gender in descending order and then by Salary in ascending order. The result is shown below.  
  
  
  
**Non Clustered Index:**  
A nonclustered index is analogous to an index in a 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 nonclustered index is stored separately from the actual data, a table can have more than one non clustered index, just like how a book can have an index by Chapters at the beginning and another index 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 the storage of data in the table.   
  
**The following SQL creates a Nonclustered** index on the NAME column on tblEmployee table:  
Create NonClustered Index IX\_tblEmployee\_Name  
ON tblEmployee(Name)  
  
**Difference between Clustered and NonClustered Index:**  
1. **Only one clustered index per table**, where as you can have more than one non clustered index  
2. **Clustered index is faster than a non clustered index**, because, the non-clustered index has to refer back to the table, if the selected column is not present in the index.  
3. **Clustered index determines the storage order of rows in the table**, and hence doesn't require additional disk space, but where as a Non Clustered index is stored seperately from the table, additional storage space is required.