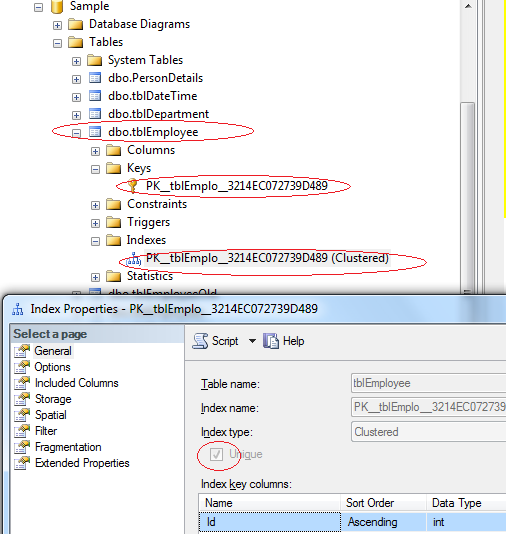
**Unique and Non-Unique Indexes - Part 37**

https://csharp-video-tutorials.blogspot.com/2012/09/unique-and-non-unique-indexes-part-37.html

**Unique index** is used to enforce uniqueness of key values in the index. Let's understand this with an example.  
  
**Create the Employee table using the script below**  
CREATE TABLE [tblEmployee]  
(  
 [Id] int Primary Key,  
 [FirstName] nvarchar(50),  
 [LastName] nvarchar(50),  
 [Salary] int,  
 [Gender] nvarchar(10),  
 [City] nvarchar(50)  
)  
  
**Since, we have marked Id column**, as the Primary key for this table, a UNIQUE CLUSTERED INDEX gets created on the Id column, with Id as the index key.   
  
**We can verify** this by executing the sp\_helpindex system stored procedure as shown below.  
Execute sp\_helpindex tblEmployee  
  
**Output:**  
https://3.bp.blogspot.com/-jP4G26uqFQE/UFIM8FMSmgI/AAAAAAAAAZw/oNFF36hw5q8/s1600/Unique+Clustered+Index.png  
  
  
**Since, we now have a UNIQUE CLUSTERED INDEX** **on the Id** **column**, any attempt to duplicate the key values, will throw an error stating 'Violation of PRIMARY KEY constraint 'PK\_\_tblEmplo\_\_3214EC07236943A5'. Cannot insert duplicate key in object dbo.tblEmployee'  
  
**Example**: The following insert queries will fail  
Insert into tblEmployee Values(1,'Mike', 'Sandoz',4500,'Male','New York')  
Insert into tblEmployee Values(1,'John', 'Menco',2500,'Male','London')

**Now let's try to drop the Unique Clustered index** on the Id column. This will raise an error stating - 'An explicit DROP INDEX is not allowed on index tblEmployee.PK\_\_tblEmplo\_\_3214EC07236943A5. It is being used for PRIMARY KEY constraint enforcement.'  
Drop index tblEmployee.PK\_\_tblEmplo\_\_3214EC07236943A5  
  
**So this error message proves that**, SQL server internally, uses the UNIQUE index to enforce the uniqueness of values and primary key.  
  
**Expand keys folder in the object explorer window**, and you can see a primary key constraint. Now, expand the indexes folder and you should see a unique clustered index. In the object explorer it just shows the '**CLUSTERED**' word. To, confirm, this is infact an UNIQUE index, right click and select properties. The properties window, shows the UNIQUE checkbox being selected.  
  
  
  
**SQL Server allows us to delete this UNIQUE CLUSTERED INDEX** from the object explorer. so, Right click on the index, and select DELETE and finally, click OK. Along with the UNIQUE index, the primary key constraint is also deleted.  
  
**Now, let's try to insert duplicate values** for the ID column. The rows should be accepted, without any primary key violation error.

Insert into tblEmployee Values(1,'Mike', 'Sandoz',4500,'Male','New York')  
Insert into tblEmployee Values(1,'John', 'Menco',2500,'Male','London')  
  
So, the UNIQUE index is used to enforce the uniqueness of values and primary key constraint.  
  
**UNIQUENESS is a property of an Index**, and both CLUSTERED and NON-CLUSTERED indexes can be UNIQUE.  
  
**Creating a UNIQUE NON CLUSTERED index** on the FirstName and LastName columns.  
Create Unique NonClustered Index UIX\_tblEmployee\_FirstName\_LastName  
On tblEmployee(FirstName, LastName)  
  
**This unique non clustered index**, ensures that no 2 entires in the index has the same first and last names. [In Part 9, of this video series](http://csharp-video-tutorials.blogspot.com/2012/08/unique-key-constraint-part-9.html), we have learnt that, a Unique Constraint, can be used to enforce the uniqueness of values, across one or more columns. There are no major differences between a unique constraint and a unique index.   
  
**In fact, when you add a unique constraint**, a unique index gets created behind the scenes. To prove this, let's add a unique constraint on the city column of the tblEmployee table.  
ALTER TABLE tblEmployee   
ADD CONSTRAINT UQ\_tblEmployee\_City   
UNIQUE NONCLUSTERED (City)  
  
**At this point, we expect a unique constraint to be created**. Refresh and Expand the constraints folder in the object explorer window. The constraint is not present in this folder. Now, refresh and expand the 'indexes' folder. In the indexes folder, you will see a UNIQUE NONCLUSTERED index with name UQ\_tblEmployee\_City.  
  
Also, executing EXECUTE SP\_HELPCONSTRAINT tblEmployee, lists the constraint as a UNIQUE NONCLUSTERED index.  
https://3.bp.blogspot.com/-NaYlKgWbZRU/UFIQbtrC9PI/AAAAAAAAAaA/KuXasoa7Pdk/s1600/Unique+Nonclustered+index.png  
  
  
**So creating a UNIQUE constraint**, actually creates a UNIQUE index. So a UNIQUE index can be created explicitly, using CREATE INDEX statement or indirectly using a UNIQUE constraint. So, when should you be creating a Unique constraint over a unique index.To make our intentions 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 between a unique index created by a unique constraint or manually created.  
  
**Note:**  
**1. By default, a PRIMARY KEY constraint**, creates a unique clustered index, where as a UNIQUE constraint creates a unique nonclustered index. These defaults can be changed if you wish to.  
  
**2. A UNIQUE constraint or a 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.

**3. By default, duplicate values are not allowed on key columns**, when you have a unique index or constraint. For, example, if I try to insert 10 rows, out of which 5 rows contain duplicates, then all the 10 rows are rejected. However, if I want only the 5 duplicate rows to be rejected and accept the non-duplicate 5 rows, then I can use IGNORE\_DUP\_KEY option. An example of using IGNORE\_DUP\_KEY option is shown below.  
CREATE UNIQUE INDEX IX\_tblEmployee\_City  
ON tblEmployee(City)  
WITH IGNORE\_DUP\_KEY