**2.Create a set of tables, add foreign key constraints and incorporate referential integrity.**

**Referential Integrity constraints in DBMS**

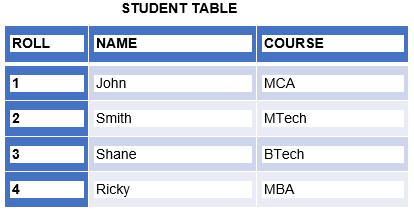
A referential integrity constraint is also known as **foreign key constraint**. A foreign key is a key whose values are derived from the Primary key of another table.

The table from which the values are derived is known as **Master or Referenced** Table and the Table in which values are inserted accordingly is known as **Child or Referencing** Table, In other words, we can say that the table containing the **foreign key** is called the **child table**, and the table containing the **Primary key/candidate key** is called the **referenced or parent table**. When we talk about the database relational model, the candidate key can be defined as a set of attribute which can have zero or more attributes.

**The syntax of the Master Table or Referenced table is:**

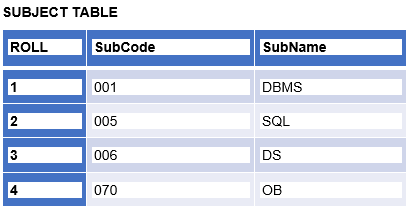
1. CREATE TABLE Student (Roll **int** PRIMARY KEY, Name varchar(25) , Course varchar(10) );

Here column Roll is acting as **Primary Key,** which will help in deriving the value of foreign key in the child table.



**The syntax of Child Table or Referencing table is:**

1. CREATE TABLE Subject (Roll **int** references Student, SubCode **int**, SubName varchar(10) );



In the above table, column Roll is acting as **Foreign Key,** whose values are derived using the Roll value of Primary key from Master table.

Foreign Key Constraint OR Referential Integrity constraint.

There are two referential integrity constraint:

**Insert Constraint:** Value cannot be inserted in CHILD Table if the value is not lying in MASTER Table

**Delete Constraint:** Value cannot be deleted from MASTER Table if the value is lying in CHILD Table

Suppose you wanted to insert Roll = 05 with other values of columns in SUBJECT Table, then you will immediately see an error "**Foreign key Constraint Violated**" i.e. on running an insertion command as:

**Insert into SUBJECT values(5, 786, OS); will not be entertained by SQL due to Insertion Constraint** ( As you cannot insert value in a child table if the value is not lying in the master table, since Roll = 5 is not present in the master table, hence it will not be allowed to enter Roll = 5 in child table )

Similarly, if you want to delete Roll = 4 from STUDENT Table, then you will immediately see an error "**Foreign key Constraint Violated**" i.e. on running a deletion command as:

**Delete from STUDENT where Roll = 4; will not be entertained by SQL due to Deletion Constraint.** ( As you cannot delete the value from the master table if the value is lying in the child table, since Roll = 5 is present in the child table, hence it will not be allowed to delete Roll = 5 from the master table, lets if somehow we managed to delete Roll = 5, then Roll = 5 will be available in child table which will ultimately violate insertion constraint. )

**ON DELETE CASCADE.**

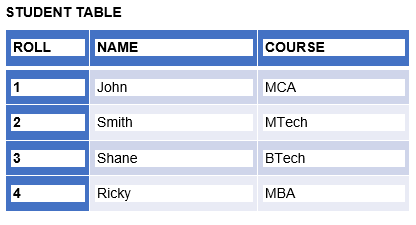
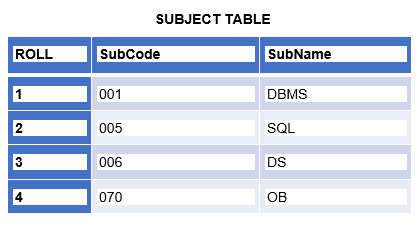
As per deletion constraint: Value cannot be deleted from the MASTER Table if the value is lying in CHILD Table. The next question comes can we delete the value from the master table if the value is lying in the child table without violating the deletion constraint? i.e. The moment we delete the value from the master table the value corresponding to it should also get deleted from the child table.

The answer to the above question is YES, we can delete the value from the master table if the value is lying in the child table without violating the deletion constraint, we have to do slight modification while creating the child table, i.e. by adding **on delete cascade**.

**TABLE SYNTAX**

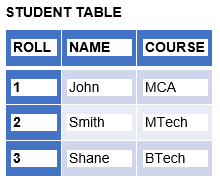
1. CREATE TABLE Subject (Roll **int** references Student on delete cascade, SubCode **int**, SubName varchar(10) );

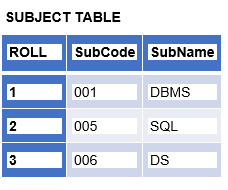
In the above syntax, just after references keyword( used for creating foreign key), we have added on delete cascade, by adding such now, we can delete the value from the master table if the value is lying in the child table without violating deletion constraint. Now if you wanted to delete Roll = 5 from the master table even though Roll = 5 is lying in the child table, it is possible because the moment you give the command to delete Roll = 5 from the master table, the row having Roll = 5 from child table will also get deleted.

The above two tables STUDENT and SUBJECT having four values each are shown, now suppose you are looking to delete Roll = 4 from STUDENT( Master ) Table by writing a SQL command: **delete from STUDENT where Roll = 4;**

The moment SQL execute the above command the row having Roll = 4 from SUBJECT( Child ) Table will also get deleted, The resultant **STUDENT and SUBJECT** table will look like:





From the above two tables STUDENT and SUBJECT, you can see that in both the table Roll = 4 gets deleted at one go without violating deletion constraint.

Sometimes a very important question is asked in interviews that: Can Foreign Key have NULL values?

The answer to the above question is YES, it may have NULL values, whereas the Primary key cannot be NULL at any cost. To understand the above question practically let's understand below the concept of delete null.

**ON DELETE NULL.**

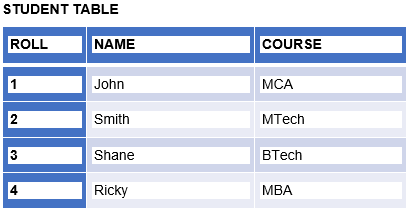
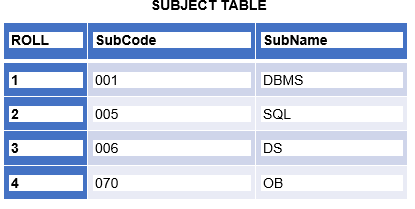
As per deletion constraint: Value cannot be deleted from the MASTER Table if the value is lying in CHILD Table. The next question comes can we delete the value from the master table if the value is lying in the child table without violating the deletion constraint? i.e. The moment we delete the value from the master table the value corresponding to it should also get deleted from the child table or can be replaced with the NULL value.

The answer to the above question is YES, we can delete the value from the master table if the value is lying in child table without violating deletion constraint by inserting NULL in the foreign key, we have to do slight modification while creating child table, i.e. by adding **on delete null**.

**TABLE SYNTAX:**

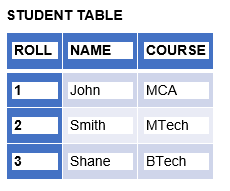
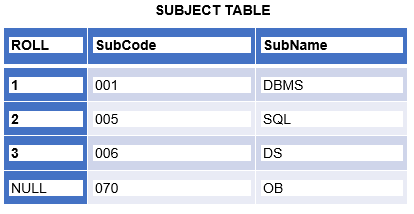
1. CREATE TABLE Subject (Roll **int** references Student on delete **null**, SubCode **int**, SubName varchar(10) );

In the above syntax, just after references keyword( used for creating foreign key), we have added on delete null, by adding such now, we can delete the value from the master table if the value is lying in the child table without violating deletion constraint. Now if you wanted to delete Roll = 4 from the master table even though Roll =4 is lying in the child table, it is possible because the moment you give the command to delete Roll = 4 from the master table, the row having Roll = 4 from child table will get replaced by a NULL value.

The above two tables STUDENT and SUBJECT having four values each are shown, now suppose you are looking to delete Roll = 4 from STUDENT( Master ) Table by writing a SQL command: **delete from STUDENT where Roll = 4;**

The moment SQL execute the above command the row having Roll = 4 from SUBJECT( Child ) Table will get replaced by a NULL value, The resultant **STUDENT and SUBJECT** table will look like:

From the above two tables STUDENT and SUBJECT, you can see that in table STUDENT Roll = 4 get deleted while the value of Roll = 4 in the SUBJECT table is replaced by NULL. This proves that the Foreign key can have null values. If in the case in SUBJECT Table, column Roll is Primary Key along with Foreign Key then in that case we could not make a foreign key to have NULL values.