12/12/24

⊙ Insertion using EF:

```
→ class Program
      {
          static void Main (....)
          { Student student = new Student();
            Student. Name = "Ali"
            Student .RollNumber = "1"
              Student . CGPA = 4.0f
```

:DBSet
In Memory
Table

```
using(My ApplicationContext context =
                    new MyApplicationContext())
{
    context.Students.Add(student);
    context.SaveChanges();
}
```

→ For Department:

```
Main(...)
{ Department d= new Department();
  d.Id= 1;
  d.Name= "Software Engineering";
  using(MyApplicatio context =
             new MyApplicationContext())
  {  context.~~Student~~ Departments .Add(d);
     context.SaveChanges();
  }
}
```

→ All students from table:

→ Reading -- LINQ (Language Integrated Query.

~~Below~~ In using ---

```
{
var data=context.students.ToList();
foreach (var item in data)
  {  Console.WriteLine(item.Name);
  }
```

⊙ //print all departments

```
var data = context.Departments.ToList();
foreach (var data1 in data)
{
    Console.WriteLine(data1.Name);
}
```

⊙ Where CGPA is —:

```
var data = context.Students.where(s=>
                              s.CGPA
                                >3)
                              .ToList()
```

⊙ Only RollNo or Name Required:

```
var data = context.Students.Whare(s=> s.CGPA>3)
                      .Select(a=> a.RollNumber)
                                 .ToList().
```

→ Select both Name & Roll No.

```
   //     //     //
           .select(a=> new { N=a.Name,
                             R=a.RollNumber})
                                  .ToList()
```

⊙ CGPA >3 & Name = 'S%'

and return Name, Roll, CGPA

```
//    //
var data = context.Studants .where
                    (s=> s.CGPA>3 ,
 s.Name.Startwith('S')         s.Name Like 'S%').
                    Select(a=> ... RollNumber
```

N = a.Name, C = a.CGPA)

Correct
→ " " context Students.
where(s=> s.Name.startsWith('s')),
where(s=> s.CGPA > 3).Select(a=>
new {N=a.Name, R=a.RollNumber,
C=a.CGPA})

→ Update Data:

→ First we get data and assign new values and call save changes.

```
var x= context.students.First();   (returns 1st object)
  x.CGPA= 1f;                      First(s=>S.Name =="Test"
context.Savechanges();
```

→ Delete Data:

→ Get Data First:

```
var y= context.students.First();
 context.Students.Remove(y);
 Context.Save Changes();
```

: Can use where condition
: then edit or remove by using loop

: var y= context.Students.single(s=>s.Name =='Ali')

:Single
(If more than one exists it throw exception)

: First
(If more than one exists, it returns first only).

: We can call savechanges() only

→ Student Belongs to Department

: Class Student
```
{
    ;;;

    ;;

    public Department
            Department {get; set;}
}
```
: Department
many ↓↑ one
student
(Many to one)

: Class Department
```
{
    ;;

    public List<students> Students
                    {get; set;}
}
```
: Now run Migrations:

dotnet ef migrations add Add column

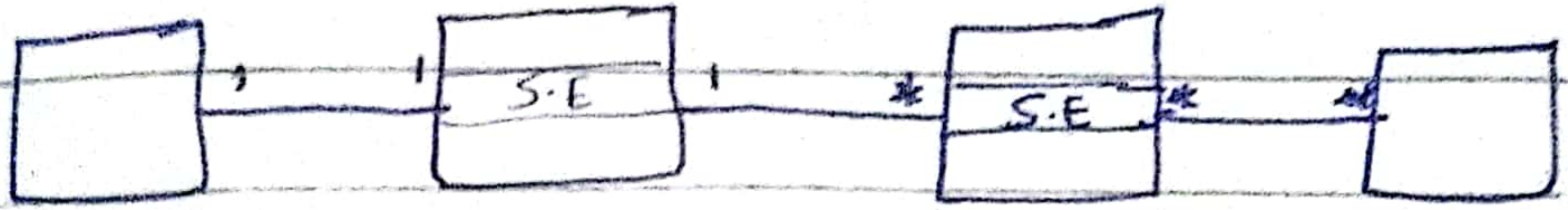
⊙ <u>Indexing</u>:

→ Record of Data (Book Example
Index: Page Number)

→ In DB Index is taking space but make our work easy.

→ Cascading delete both data in local and foreign key

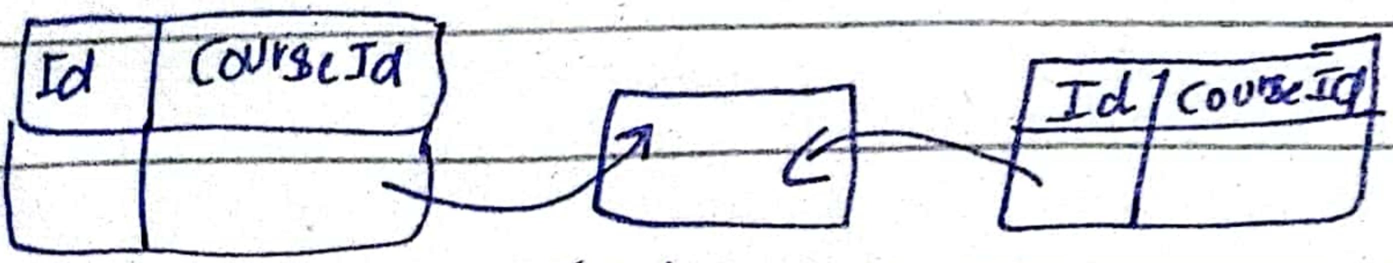
; onDelete: ReferentialAction.Cascade.



Dept Detail Dept Student course

F.K(Dept) F.K(DeptDetail) List<Courses> List<Students>

- If relations is many to many.



(Automatically created)