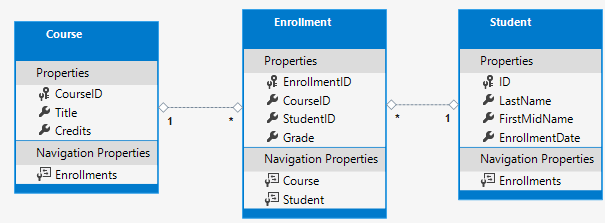
Cheat-Sheet for Contoso University

# Models

In this project we have 3 Models;



* Student(int ID, string Lastname, string Firstname, DateTime EnrollmentDate, ICollection<Enrollment> Enrollments)
* Course(int CourseID, string Title, int Credits, ICollection<Enrollment> Enrollments)
* Enrollment(int EnrollmentID, CourseID, StudentID, Grade? Grade, Course Course, Student Student)

Primary Key: ID ( PrimaryKey ) or CourseID( propertyName / PrimaryKey )

Foreign Key: CourseID ( propertyName / PrimaryKey ) or StudentID ( propertyName / PrimaryKey )

* You shouldn’t choose only one pattern for one project, because otherwise it gets really confusing. Despite this we will use both as a proof that both work.

# Create the Database Context

In this class we create one DbSet-Property for each entity we have (Student, Course, Enrollment). This creates a table and a row inside it named liked the entity. At least we added some code so the DB-Tables are set singular instead of plural.

# Dependencies Injection

ASP.net core has implemented Dependencies Injection by default. To register SchoolContext as a service, we added some lines to the startup.cs . Then we need to add some using statements and namespaces. Finally we set a String in the appsettings.json to get the connection.

# Add code to initialize the database with test data

We create a DbInitializer.cs file, which builds the db and adds some test data code into it. The class will only fill the db with test data when its empty. The created entities are structured in array, one for each entity.

# Create a controller and views

We created a Scaffolded-Controller named StudentsController.cs which is connected to the Student-Model and SchoolContext Datafile. It automatically generates a folder Students with 5 views. The StudentsController has generated a constructor with the SchoolContext as its parameter. This is the point where DI gets to its use.