.Net Cheat sheet (Entity-FrameWork Part)

Relations :

🥇OneToMany :

1/ Using Annotation

Entity1:	Entity2:
.. .. public type_mta_id_mta_entity_2 ent2FK { get; set; } [ForeignKey(" ent2FK")] public virtual Entity2 ent2 { get; set; }	.. .. public virtual IList ent1{ get; set; }

2/ Using Fluent Api

2 ways with Fluent Api :

1/ configuration file => apply config
2/ context

1/ configuration file method

step1 add entity Config File ent1config under infrastructure :

```
public class Entity1config : IEntityTypeConfiguration {

public void Configure(EntityTypeBuilder builder) {

builder.HasOne(p => p.ent2 )
.WithMany(p => p.ent1)
.HasForeignKey(p => p.ent2FK)

}
}
```

step2: apply configuration go to OnModelCreating method in Context class :

```
protected override void OnModelCreating(ModelBuilder
modelBuilder)
 {
 modelBuilder.ApplyConfiguration(new ent1config ());
}
```

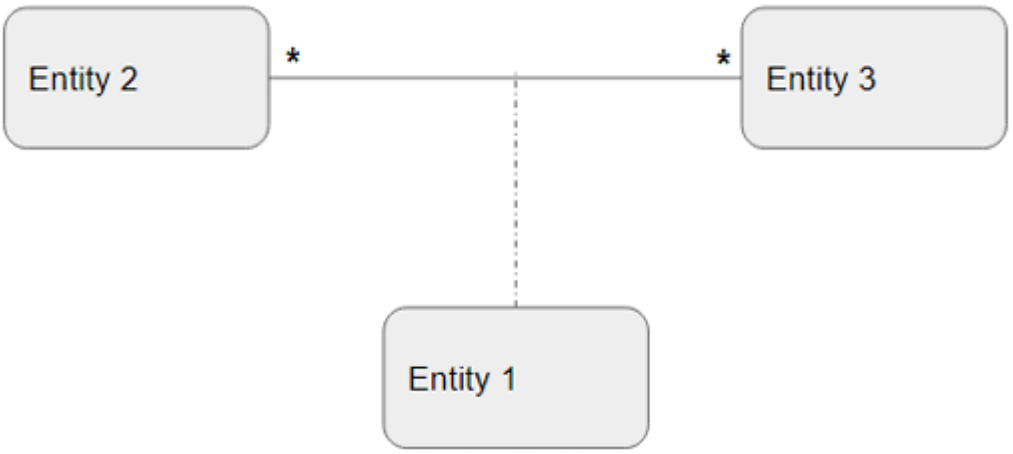
2/ config relation directly in OnModelCreating

```
modelBuilder.Entity<Entity1>()
  .HasOne(p => p.ent2 )
 .WithMany(p => p.ent1)
 .HasForeignKey(p => p.ent2FK)
```

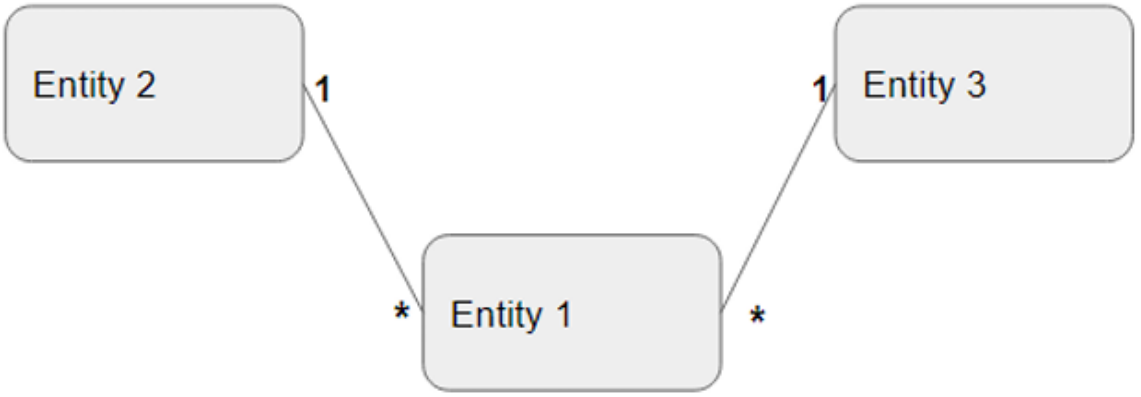
!!! RQ : ManyToMany config (rename associative table):

```
builder.hasmany(t=>t.entx)
  .withmany(p=>p.enty)
 .usingentity(p=>p.totable(“ name”));
```

🥈 Table porteuse:



Convert to two 1..* relations 👍



Step1 : config 2 OneToMany relations as previously mentioned .

Step2 : primary key (Example : 2 foreign keys + prop) :

```
builder.haskey(r=>new{
                r.ent1FK,
                r.ent2FK,
                r.date
        });
```

🥉 Complex type :

** Type Détenue :

1/ Using Annotation

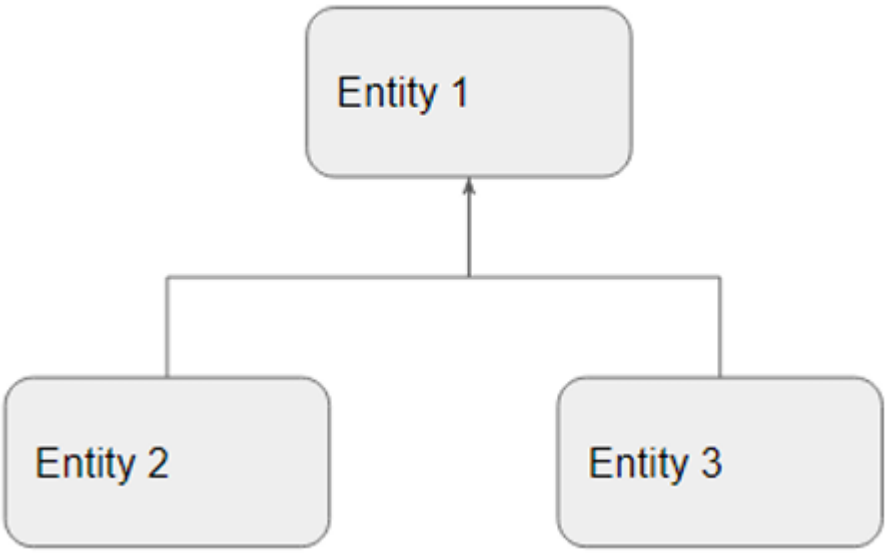
[Owned]
Entity3
string prop1
string prop2

Entity4
Entity3 prop

1/ Using Fluent Api

```
modelbuilder.Entity<Entity4>()
            .Ownsone(p=>p.prop1,
                     p=>prop2) ;
```

** Heritage configuration :



```
public class Entity1
{
    public int Id { get; set; }
    public string CommonProperty { get; set; }
}

public class Entity2 : Entity1
{
    public string Property2 { get; set; }
}

public class Entity3 : Entity1
{
    public string Property3 { get; set; }
}
```

1/ TPH:Using discriminator

Either Onmodel creating method or entity file configuration :

```
modelBuilder.Entity<Entity1>()
            .HasDiscriminator<string>("EntityType")
            .HasValue<Entity2>("Entity2")
            .HasValue<Entity3>("Entity3");
    }
```

RQ `:HasDiscriminator<string>("EntityType")` **pour spécifier que la colonne discriminante s'appelle "EntityType" et utilise le type string.**

!!!!!! Also We have to ADD DbSetOnly For Entity1👍

1/ TPH:Using discriminator

Either Onmodel creating method or entity file configuration :

```
modelBuilder.Entity<Entity1>().ToTable("Entities");

modelBuilder.Entity<Entity2>().ToTable("Entity2");

modelBuilder.Entity<Entity3>().ToTable("Entity3");
```

!!!!!! Don’t forget to add new DbSet for each entity 👍