

---

# Acesso a dados com Microsoft Entity Framework

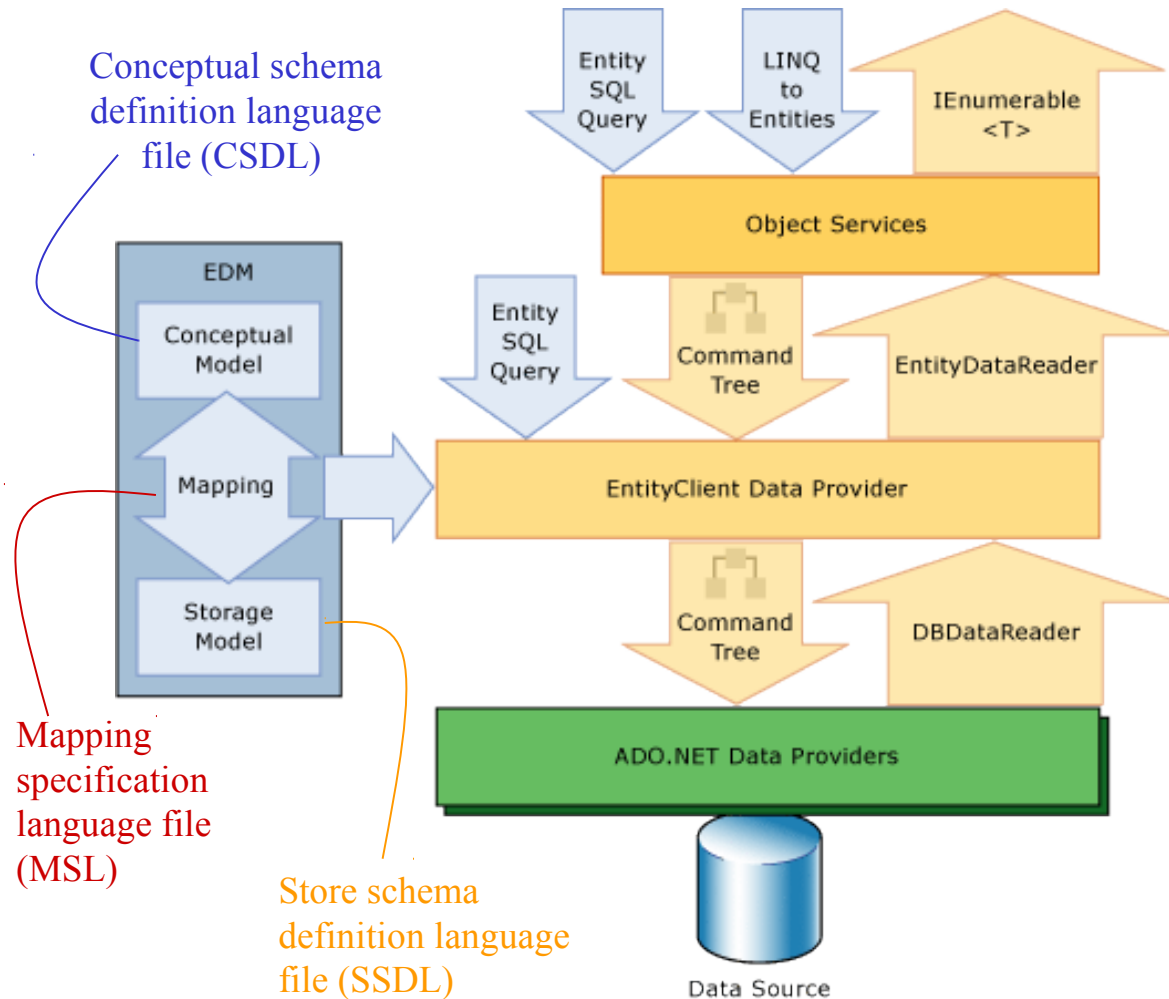
# Entity Framework – o que é?

---

## O que é?

- Uma API de acesso a dados não integrada na linguagem
- Uma ferramenta ORM
- Uma forma de acesso a dados directamente sobre o modelo conceptual, portanto, totalmente independente do SGBD utilizado.
- Uma implementação dos padrões “unit of work” e “Repository”.

# Entity Framework – arquitetura geral



Adaptado de “Introducing the Entity Framework“ (<http://msdn.microsoft.com/en-us/library/bb399567.aspx>)

# Entity Framework

---

Duas grandes formas de trabalhar:

- **ObjectContext**

- Entidades EF

- Entidades POCO (plain old CLR objects)

- **DBContext**

- Apenas entidades POCO



*Nestes slides apenas iremos considerar esta forma*

# Entity Framework

---

## Modelos:

- **Database SingleOrDefault**
  - O modelo (EDM) é gerado a partir do “reverse engineering” da base de dados e as classes são (auto) geradas a partir do modelo.
- **Model SingleOrDefault**
  - Cria-se o modelo primeiro. A BD é gerada a partir do modelo e as classes são (auto)geradas a partir do modelo
- **Code SingleOrDefault (nova base de dados)**
  - Criam-se as classe e mapeamentos manualmente. A base de dados é criada a partir do código. Usam-se “migrations” para fazer evoluir a BD
- **Code SingleOrDefault (base de dados existente)**
  - Criam-se as classe e mapeamentos manualmente. Existem algumas ferramentas de “reverse engineering” que facilitam este mapeamento (tem de se instalar EF Power tools)


# Entity Framework

---

Características essenciais:

- Implementação do padrão “Repository”

```
using (var ctx = new ASIEntities7()) {  
    // criar um aluno  
    var al = new Aluno { NumAl = 1111, Nome = "xico" };  
    var interesse =  
        new AlunosAssEst { NumAl = 1111, Interesse = "musica" };  
    al.AlunosAssEsts.Add(interesse);  
    ctx.Alunos.Add(al);  
}
```



# Entity Framework – características essenciais

---

- Implementação do padrão “unit of work”

```
using (var ctx = new ASIEntities7()) {  
    // criar um aluno  
    var al = new Aluno { NumAl = 1111, Nome = "xico" };  
    var interesse =  
        new AlunosAssEst { NumAl = 1111, Interesse = "musica" };  
    al.AlunosAssEsts.Add(interesse);  
    ctx.Alunos.Add(al);  
  
    ctx.SaveChanges(); // commit  
}
```



# Entity Framework – características essenciais

- Change tracking
  - Via virtual proxies.
  - Via Snapshots

```
using (var ctx = new ASIEntities7()) {  
    var al = (from a in ctx.Alunos where a.NumAl == 1111 select  
a).SingleOrDefault();  
  
    al.Nome = "ZeZe";  
  
    DbEntityEntry<Aluno> x = ctx.Entry<Aluno>(al);  
    DbPropertyValues original = x.OriginalValues;  
    DbPropertyValues corrente = x.CurrentValues;  
    EntityState state = x.State;  
    Console.WriteLine(string.Format("State: {0}, Old Value: {1}, New  
Value: {2}",  
state, original.GetValue<string>("Nome"),  
corrente.GetValue<string>("Nome")));  
  
    ctx.SaveChanges();  
    state = x.State;  
    Console.WriteLine( string.Format("State: {0}, Old Value: {1}, New  
Value: {2}",  
state, original.GetValue<string>("Nome"),  
corrente.GetValue<string>("Nome")));  
}
```

State: Modified, Old Value: xico, New Value: ZeZe

State: Unchanged, Old Value: ZeZe, New Value: ZeZe




# Entity Framework – características essenciais

---

- Lazy Loading
  - Via virtual proxies

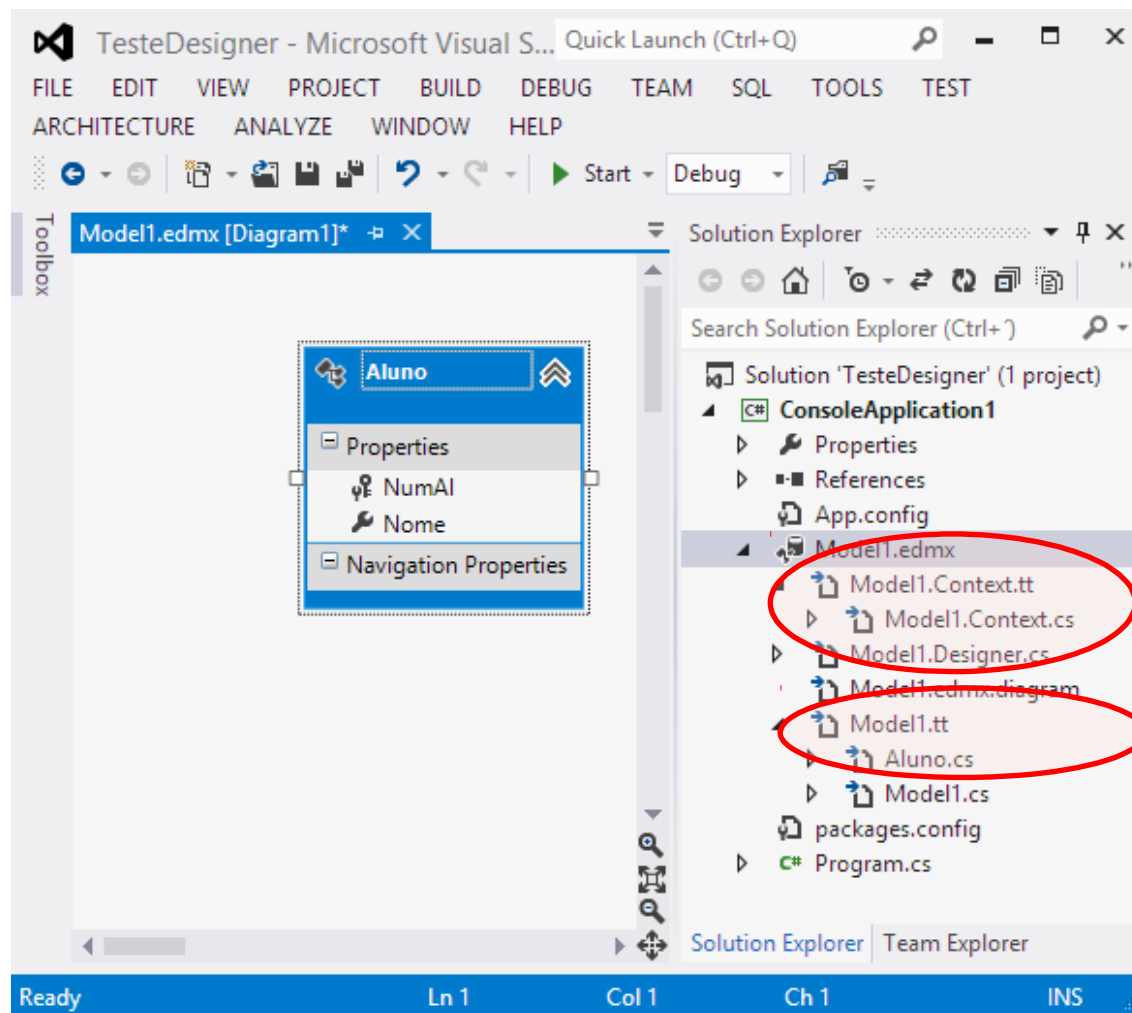
```
using (var ctx = new ASIEntities7()) {  
  
    var q = from a in ctx.Alunos  
            select a;  
  
    Console.WriteLine("Alunos existentes:");  
  
    foreach (var a in q) {  
  
        Console.WriteLine(a.NumAl);  
  
        foreach (var i in a.AlunosAssEsts) {  
            Console.WriteLine("{0}:{1}", i.nSeq, i.Interesse);  
        }  
    }  
}
```

Só agora é que AlunosAssEsts é acedido

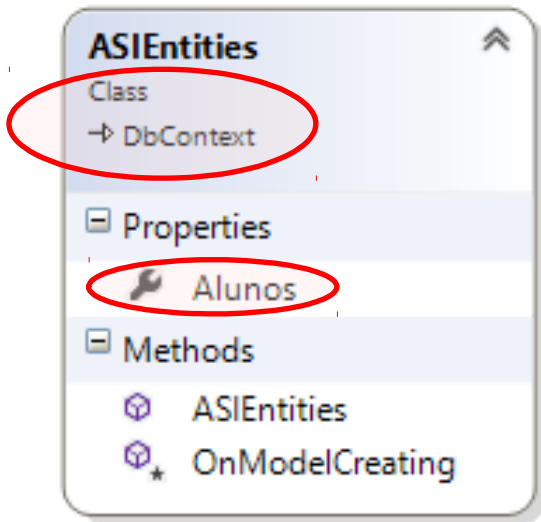


# Entity Framework Exemplo 1 – o modelo conceptual

**CREATE TABLE Alunos( NumAl int primary key, Nome varchar(60))**



# Entity Framework Exemplo 1 – as classes geradas



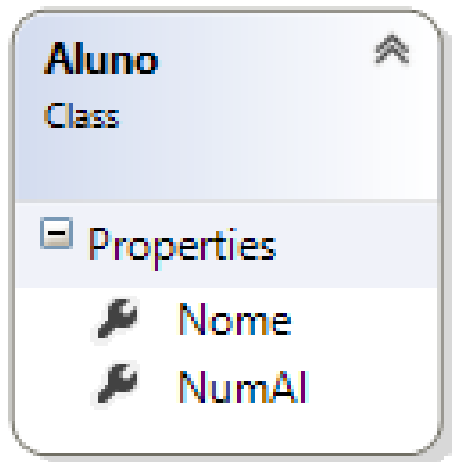
```
public partial class ASIEntities : DbContext
{
    public ASIEntities() : base("name=ASIEntities") {
    }

    protected override void
        OnModelCreating(DbModelBuilder modelBuilder){
        throw new
        UnintentionalCodeSingleOrDefaultException();
    }

    public DbSet<Aluno> Alunos { get; set; }
}
```

## Entity Framework Exemplo 1 – as classes geradas

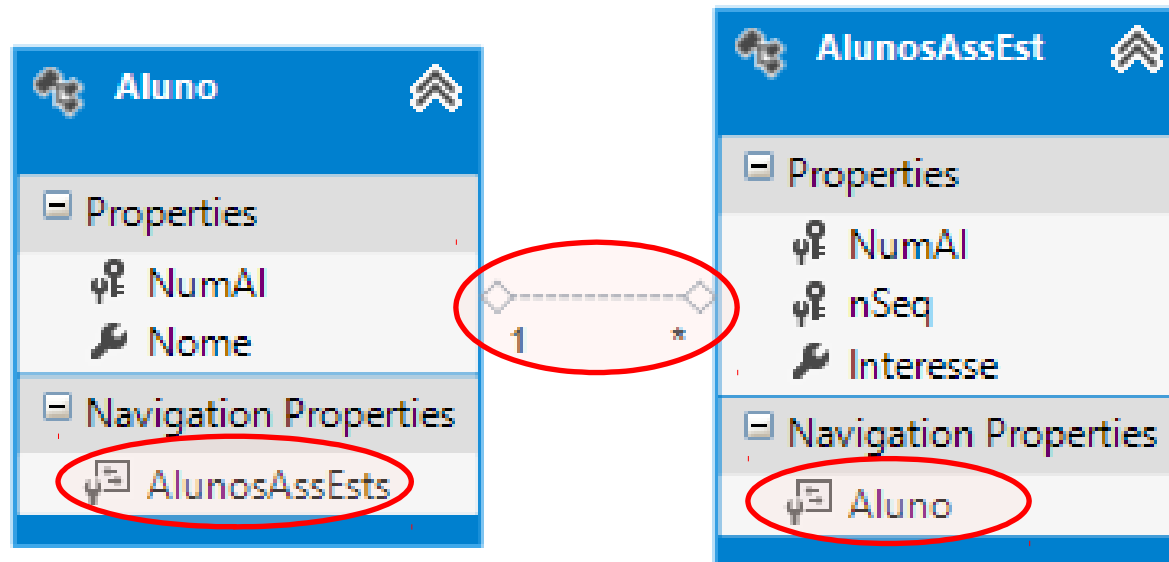
---



```
public partial class Aluno {  
    public int NumAl { get; set; }  
    public string Nome { get; set; }  
}
```

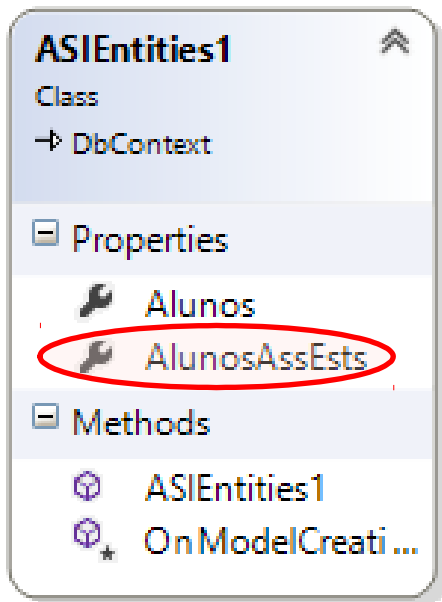
## Entity Framework Exemplo 2 (associação 1:1) – o modelo conceptual

```
CREATE TABLE AlunosAssEst(NumAl int references Alunos,  
                           nSeq int identity,  
                           Interesse varchar(10),  
                           Primary key(NumAl,nSeq)  
)
```



*Até à versão 6 só  
são reconhecidas  
associações  
baseadas na chave  
primária e não  
numa chave  
secundária (unique)*

## Entity Framework Exemplo 2 (associação 1:1) – o modelo conceptual

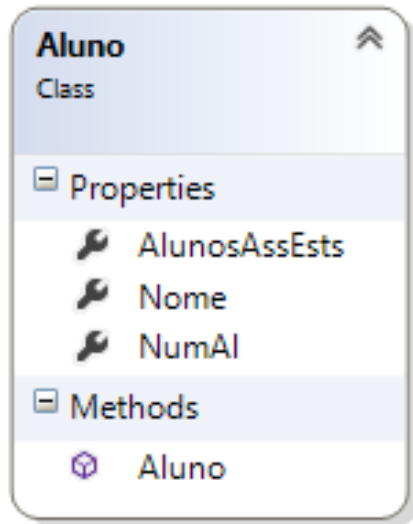


```
public partial class ASIEntities1 : DbContext
{
    public ASIEntities1(): base("name=ASIEntities1")
    {
    }

    protected override void OnModelCreating ...    {
        throw new
        UnintentionalCodeSingleOrDefaultException();
    }

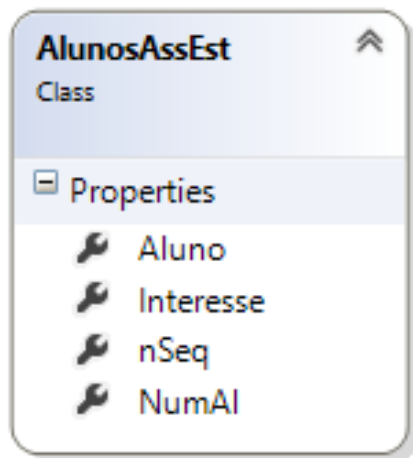
    public DbSet<Aluno> Alunos { get; set; }
    public DbSet<AlunosAssEst> AlunosAssEsts { get; set; }
}
```

# Entity Framework Exemplo 2 (associação 1:1) – o modelo conceptual



```
public partial class Aluno {  
    public Aluno() {  
        this.AlunosAssEsts = new HashSet<AlunosAssEst>();  
    }  
  
    public int NumAl { get; set; }  
    public string Nome { get; set; }  
  
    public virtual ICollection<AlunosAssEst> AlunosAssEsts { get; set; }  
}
```

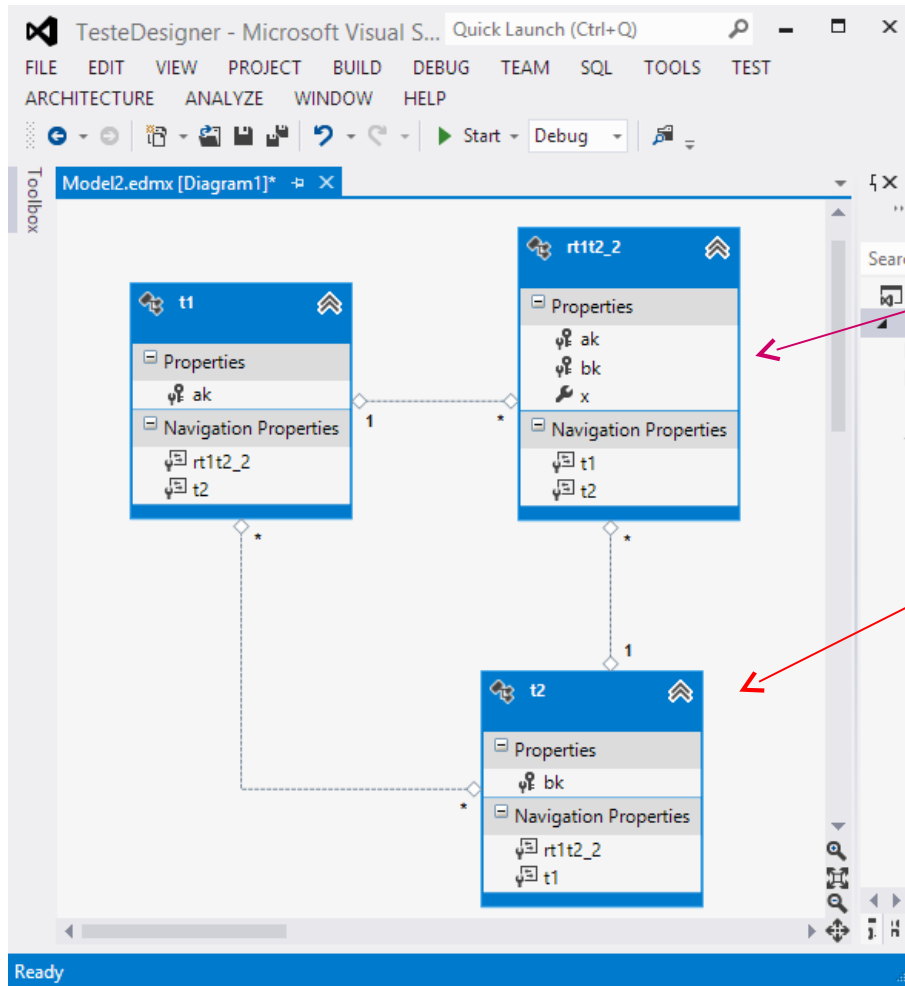
*Para lazy loading (virtual proxy)*



```
public partial class AlunosAssEst {  
    public int NumAl { get; set; }  
    public int nSeq { get; set; }  
    public string Interesse { get; set; }  
  
    public virtual Aluno Aluno { get; set; }  
}
```

*Ver slides ASI\_LAD (padrão virtual proxy)*

# Entity Framework Exemplo 4 (associação N:N) – o modelo conceptual



**create table t1(ak int primary key)**

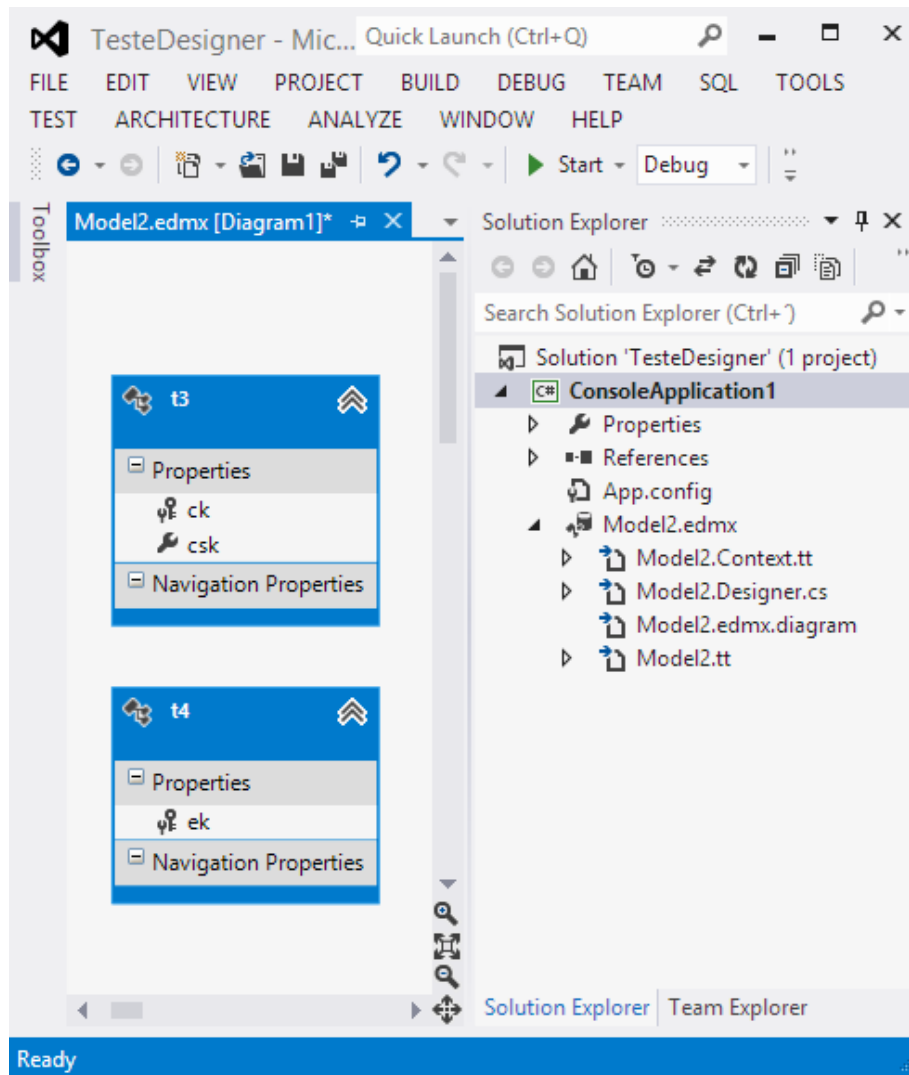
**create table t2(bk int primary key)**

**create table rt1t2\_2(ak int references t1,  
bk int references t2,  
primary key (ak,bk),  
x int)**

**create table rt1t2\_1(ak int references t1,  
bk int references t2,  
primary key (ak,bk))**



# Entity Framework Exemplo 4 (associação N:N) – o modelo conceptual



create table t3(ck int primary key,  
csk int unique)

create table t4(ek int primary key  
references t3(csk))

**CUIDADO!!!**

# Entity Framework – Estrutura do modelo EDM

---

```
<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx Version="1.0" xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx">
  <edmx:Runtime>
    <edmx:StorageModels>
      ...
    </edmx:StorageModels>
    <edmx:ConceptualModels>
      ...
    </edmx:ConceptualModels>
    <edmx:Mappings>
      ...
    </edmx:Mappings>
  </edmx:Runtime>
</edmx:Edmx>
```

# Entity Framework – Estrutura do modelo conceptual

---

```
<edmx:ConceptualModels>
  <Schema ...">
    <EntityContainer Name="ASIEntities7" p1:LazyLoadingEnabled="true">
      <EntitySet Name="Alunos" EntityType="ASIModel.Aluno" />
      <EntitySet Name="AlunosAssEsts" EntityType="ASIModel.AlunosAssEst" />
      <AssociationSet Name="FK__AlunosAss__NumAl__1273C1CD"
        Association="ASIModel.FK__AlunosAss__NumAl__1273C1CD">
        <End Role="Alunos" EntitySet="Alunos" />
        <End Role="AlunosAssEst" EntitySet="AlunosAssEsts" />
      </AssociationSet>
    </EntityContainer>
```

*continua* 

# Entity Framework – Estrutura do modelo conceptual

---

```
<EntityType Name="Aluno">
  <Key>
    <PropertyRef Name="NumAl" />
  </Key>
  <Property Name="NumAl" Type="Int32" Nullable="false" />
  <Property Name="Nome" Type="String" MaxLength="60" Unicode="false" FixedLength="false" />
  <NavigationProperty Name="AlunosAssEsts"
    Relationship="ASIModel.FK__AlunosAss__NumAl__1273C1CD" FromRole="Alunos"
    ToRole="AlunosAssEst" />
</EntityType>
<EntityType Name="AlunosAssEst">
  <Key>
    <PropertyRef Name="NumAl" />
    <PropertyRef Name="nSeq" />
  </Key>
  <Property Name="NumAl" Type="Int32" Nullable="false" />
  <Property Name="nSeq" Type="Int32" Nullable="false" p1:StoreGeneratedPattern="Identity" />
  <Property Name="Interesse" Type="String" MaxLength="10" Unicode="false" FixedLength="false" />
  <NavigationProperty Name="Aluno" Relationship="ASIModel.FK__AlunosAss__NumAl__1273C1CD"
    FromRole="AlunosAssEst" ToRole="Alunos" />
</EntityType>
```

*continua* 

# Entity Framework – Estrutura do modelo conceptual

---

```
<Association Name="FK__AlunosAss__NumAl__1273C1CD">
  <End Role="Alunos" Type="ASIModel.Aluno" Multiplicity="1" />
  <End Role="AlunosAssEst" Type="ASIModel.AlunosAssEst" Multiplicity="*" />
  <ReferentialConstraint>
    <Principal Role="Alunos">
      <PropertyRef Name="NumAl" />
    </Principal>
    <Dependent Role="AlunosAssEst">
      <PropertyRef Name="NumAl" />
    </Dependent>
  </ReferentialConstraint>
</Association>
</Schema>
</edmx:ConceptualModels>
```

*continua* 

# Entity Framework – Estrutura do modelo de armazenamento

---

```
<edmx:StorageModels>
  <Schema ...">
    <EntityContainer Name="ASIModelStoreContainer">
      <EntitySet Name="Alunos" EntityType="ASIModel.Store.Alunos" store:Type="Tables" Schema="dbo" />
      <EntitySet Name="AlunosAssEst" EntityType="ASIModel.Store.AlunosAssEst" store:Type="Tables"
        Schema="dbo" />
      <AssociationSet Name="FK__AlunosAss__NumAl__1273C1CD"
        Association="ASIModel.Store.FK__AlunosAss__NumAl__1273C1CD">
        <End Role="Alunos" EntitySet="Alunos" />
        <End Role="AlunosAssEst" EntitySet="AlunosAssEst" />
      </AssociationSet>
    </EntityContainer>
    <EntityType Name="Alunos">
      <Key> <PropertyRef Name="NumAl" /> </Key>
      <Property Name="NumAl" Type="int" Nullable="false" />
      <Property Name="Nome" Type="varchar" MaxLength="60" />
    </EntityType>
    ...
  </Schema>
</edmx:StorageModels>
```



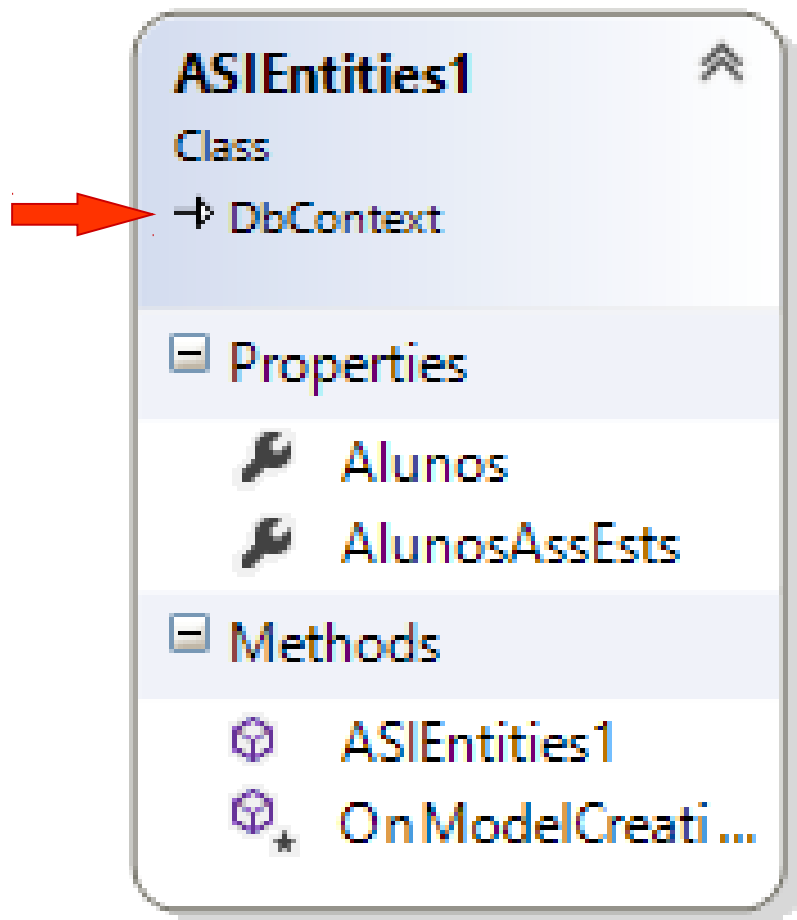
*continua*

# Entity Framework – Estrutura do modelo de “mapeamento”

---

```
<edmx:Mappings>
  <Mapping Space="C-S" xmlns="http://schemas.microsoft.com/ado/2009/11/mapping/cs">
    <EntityContainerMapping StorageEntityContainer="ASIModelStoreContainer"
                          CdmEntityContainer="ASIEntities7">
      <EntitySetMapping Name="Alunos">
        <EntityTypeMapping TypeName="ASIModel.Aluno">
          <MappingFragment StoreEntitySet="Alunos">
            <ScalarProperty Name="NumAl" ColumnName="NumAl" />
            <ScalarProperty Name="Nome" ColumnName="Nome" />
          </MappingFragment>
        </EntityTypeMapping>
      </EntitySetMapping>
      <EntitySetMapping Name="AlunosAssEsts">
        <EntityTypeMapping TypeName="ASIModel.AlunosAssEst">
          <MappingFragment StoreEntitySet="AlunosAssEst">
            <ScalarProperty Name="NumAl" ColumnName="NumAl" />
            <ScalarProperty Name="nSeq" ColumnName="nSeq" />
            <ScalarProperty Name="Interesse" ColumnName="Interesse" />
          </MappingFragment>
        </EntityTypeMapping>
      </EntitySetMapping>
    </EntityContainerMapping>
  </Mapping>
</edmx:Mappings>
```

## Entity Framework – As classes geradas



### O contexto (DbContext):

- Controla alterações das entidades
- Faz actualizações à BD
- Controla concorrência
- Controla conexões

### • Principais construtores:

**DbContext()**

*Connection string*

**DbContext(String)**

**DbContext(DbConnection, Boolean)**

*true para eliminar a conexão com dispose*




# Entity Framework – As classes geradas

---

## DbContext

- **Propriedades:**

**ChangeTracker**  *Permite acesso a características relacionadas com change tracking*

**Configuration**  *Permite acesso a configurações do contexto*

**Database**  *Cria instância da classe Database que permite operações sobre a BD base.*

- **Métodos:**

**Entry<TEntity>(TEntity)**  *Devolve entrada para acesso a informação sobre uma entidade*

**GetValidationErrors**  *Retorna erros de validação das entidades sujeitas a change tracking*

**SaveChanges**

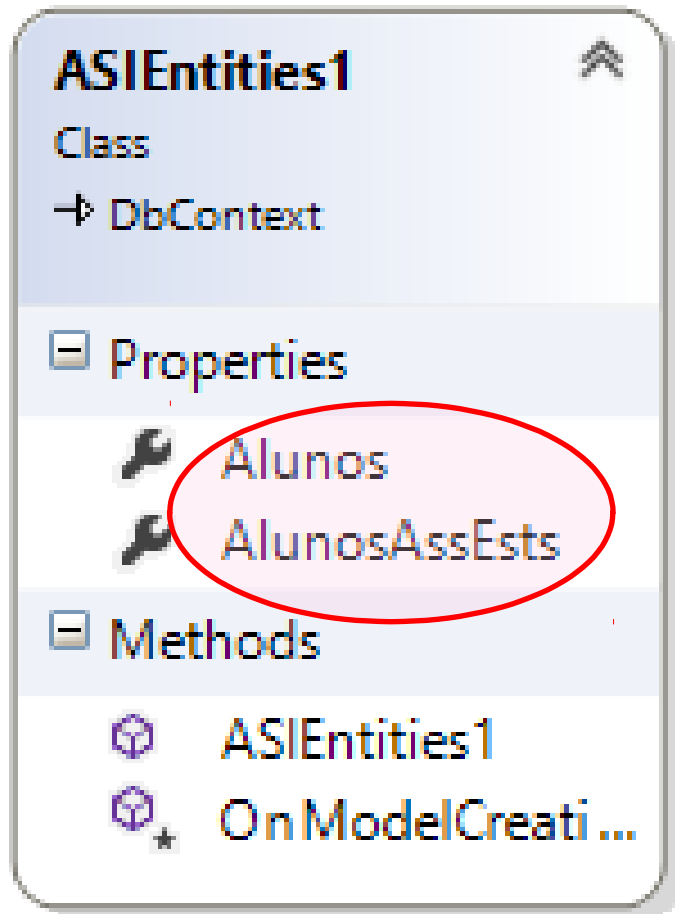
**Set<TEntity>()**  *Retorna DbSet para acesso a entidades do tipo TEntity*

# Entity Framework – As classes geradas

## A classe DBSet<Tentity>

- Alguns métodos:

```
Tentity Add(Tentity)
Tentity Attach(Tentity)
TDerivedEntity Create<TDerivedEntity>()
Tentity Find(Object[] keyValues)
Tentity Remove(Tentity)
DbQuery<TResult> Include(string path)
DbQuery<TResult> AsNoTracking()
void Load()
```



Implementa a interface: `public interface IDbSet<Tentity> : IQueryable<Tentity>, IEnumerable<Tentity>, IQueryable, IEnumerable where Tentity : class`

# Entity Framework

---

Estado das entidades e o método SaveChanges():

**Added** – a entidade existe no contexto (DbContext) mas não na BD (implica gerar INSERT)

**Unchanged** – a entidade existe no contexto e na BD e não foi alterada (não propagar nada para a BD)

**Modified** – a entidade existe na BD e foi modificada (gerar UPDATE)

**Deleted** – a entidade existe na BD e no contexto, mas foi marcada para remoção (gerar DELETE)

**Detached** – a entidade não está associada ao contexto (não existe “change tracking”)

# Entity Framework

---

```
using (var ctx = new ASIEntities7()) {  
    var al = (from a in ctx.Alunos  
              where a.NumAl == 1111  
              select a)  
              .SingleOrDefault();
```

```
    al.Nome = "ZeZe";
```

```
    DbEntityEntry<Aluno> x = ctx.Entry<Aluno>(al);  
    DbPropertyValues original = x.OriginalValues;  
    DbPropertyValues corrente = x.CurrentValues;  
    EntityState state = x.State;
```

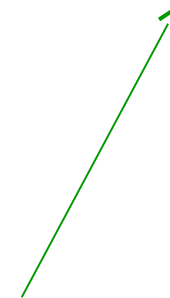
```
    Console.WriteLine(string.Format("State: {0}, Old Value: {1}, New Value: {2}",  
                                   state, original.GetValue<string>("Nome"), corrente.GetValue<string>("Nome")));
```

```
    ctx.SaveChanges();
```

```
    state = x.State;  
    Console.WriteLine(string.Format("State: {0}, Old Value: {1}, New Value: {2}",  
                                   state, original.GetValue<string>("Nome"), corrente.GetValue<string>("Nome")));
```

```
}
```

*State: Modified, Old Value: xico, New Value: ZeZe*



*State: Unchanged, Old Value: ZeZe, New Value: ZeZe*

# Entity Framework

---

```
using (var ctx = new ASIEntities7()) {
```

```
    //criar um aluno
```

```
    var al = new Aluno { NumAl = 1111, Nome = "xico" };
```

```
    var interesse = new AlunosAssEst { NumAl = 1111, Interesse = "musica" };
```

```
    al.AlunosAssEsts.Add(interesse);
```

```
    ctx.Alunos.Add(al); // todo o grafo é analisado
```

```
    ctx.SaveChanges();
```

```
}
```



# Entity Framework

---

```
using (var ctx = new ASIEntities7()) {
```

```
    var q = from a in ctx.Alunos
            select a;
```

```
    Console.WriteLine("Alunos existentes:");
```

```
    foreach (var a in q)
```

```
    {
```

```
        Console.WriteLine(a.NumAl);
```

```
        foreach (var i in a.AlunosAssEsts)
```

```
        {
```

```
            Console.WriteLine("{0}:{1}", i.nSeq, i.Interesse);
```

```
        }
```

```
    }
```

```
}
```

*Com Lazy loading, só agora é que a  
tabela AlunosAssEsts é acedida*



# Entity Framework

---

```
using (var ctx = new ASIEntities70) {
```

```
    ctx.Configuration.LazyLoadingEnabled = false;
```

```
    var q = from a in ctx.Alunos
            select a;
```

```
    Console.WriteLine("Alunos existentes:");
```

```
    foreach (var a in q) {
```

```
        Console.WriteLine(a.NumAl);
```

```
        if(!ctx.Entry(a).Collection(al => al.AlunosAssEsts).IsLoaded)
            ctx.Entry(a).Collection(al => al.AlunosAssEsts).Load();
```

```
        Console.WriteLine(a.GetType().ToString());
```

```
        foreach (var i in a.AlunosAssEsts) {
```

```
            Console.WriteLine("{0}:{1}", i.nSeq, i.Interesse);
```

```
        }
```

```
    }
```

```
}
```

*Multiplicidade (\*)*



*Carregamento a pedido*



# Entity Framework

---

```
using (var ctx = new ASIEntities7()) {  
    ctx.Configuration.LazyLoadingEnabled = false;
```

```
    var interesses = (from i in ctx.AlunosAssEsts  
        where i.Interesse.Equals("futebol")  
        select i);
```

```
    Console.WriteLine("Alunos com o interesse futebol");
```

```
    foreach (var i in interesses) {
```

```
        if (!ctx.Entry(i).Reference(inter => inter.Aluno).IsLoaded)  
            ctx.Entry(i).Reference(inter => inter.Aluno).Load();
```

```
        Console.WriteLine(i.Aluno.Nome);
```

```
    }
```

```
}
```

*Multiplicidade (1)*



*Carregamento a pedido*





# Entity Framework

---

```
using (var ctx = new ASIEntities7()) {
```

```
    ctx.Configuration.LazyLoadingEnabled = false;
```

```
    var q = (from a in ctx.Alunos.Include(a => a.AlunosAssEsts)
            select a) ;
```

```
    foreach (var a in q)
    {
```

```
        Console.WriteLine(a.NumAl);
```

```
        foreach (var i in a.AlunosAssEsts)
```

```
        {
```

```
            Console.WriteLine("{0}:{1}", i.nSeq, i.Interesse);
```

```
        }
```

```
    }
```

```
}
```



*Carregamento imediato  
(eager loading)*

# Entity Framework

---

Ter também em conta:

**ctx.Configuration.ProxyCreationEnabled = false;**

**Não existe lazy loading**

**Apenas haverá (se permitido) change tracking por snapshot**

**ctx.Configuration.AutoDetectChangesEnabled = false;**

**Não existe change tracking automático por snapshot**

**(usar ctx.ChangeTracker.DetectChanges())**

# Entity Framework

---

## Criação de proxies

```
using (var ctx = new ASIEntities7()) {  
    Aluno aSemProxy = new Aluno(); // Não é proxy
```

```
    Console.WriteLine(aSemProxy.GetType().ToString());
```

```
    var al = (from a in ctx.Alunos // proxy  
              where a.NumAl == 1111  
              select a)  
              .SingleOrDefault();
```

```
var realobj =  
    ObjectContext.GetObjectType  
        (al.GetType());  
Permite converter o proxy num objecto do  
tipo da entidade
```

```
    Console.WriteLine(al.GetType().ToString());
```

```
    Aluno nonAttachedAl = ctx.Alunos.Create(); // proxy
```

```
    Console.WriteLine(al.GetType().ToString());
```

```
}
```

# Entity Framework

---

Quando Automatic change tracking está activo. `DbContext.ChangeTracker.DetectChanges()` é chamado nas seguintes situações:

- Quando se chama:
  - `DbSet.Add`
  - `DbSet.Find`
  - `DbSet.Remove`
  - `DbSet.Local`
  - `DbContext.SaveChanges`
  - `DbSet.Attach`
  - `DbContext.GetValidationErrors`
  - `DbContext.Entry`
  - `DbChangeTracker.Entries`
- Quando se executa uma query contra `DbSet` (imediatamente antes da query ser executada sobre o contexto, para se garantir que os resultados reflectem as últimas alterações)

Quando Automatic change tracking está desactivado, apenas ocorre snapshot change tracking quando é chamado explicitamente o método `DbContext.ChangeTracker.DetectChanges()`.

# Entity Framework

---

```
using (var ctx = new ASIEntities7()) {  
  
    ctx.Configuration.AutoDetectChangesEnabled = false;  
  
    var al = (from a in ctx.Alunos  
              where a.NumAl == 1111  
              select a)  
              .SingleOrDefault();  
  
    ctx.ChangeTracker.DetectChanges();  
  
    ctx.SaveChanges();  
  
}
```



**Com proxy, não adianta. Com snapshot só agora é feito o change tracking**

## Entity Framework

---

**Para o EF criar proxies para change tracking, as classes têm de obedecer aos seguintes requisitos:**

- **A classe deve ser pública e não “sealed”**
- **Todas as propriedades devem estar marcadas como virtuais**
- **Todas as propriedades devem ter gettere e setters públicos.**
- **Todas as propriedades de navegação que sejam coleções devem ser do tipo `ICollection<T>`.**

# Entity Framework

---

**Falar sobre funcionamento de lazy loading e change tracking com mais pormenor**

**Diferença entre proxies e referências directas para a entidade. E como obter a entidade a partir do proxy.**

**Como funciona a serialização dos proxies**

**Complex types**

**MergeOption**

**EQL**

```
using (var context = new BloggingContext()){  
  
    var blog = context.Blogs.Find(1);  
    var entityType =ObjectContext.GetObjectType(blog.GetType());  
}
```

# Entity Framework

---

## Utilização “desligada”

Aluno al;

```
using (var ctx = new ASIEntities7()) {  
    al = (from a in ctx.Alunos  
          where a.NumAl == 1111  
          select a)  
        .SingleOrDefault();
```

```
    al.Nome = "Zeca Formiga";  
}
```

```
using (var ctx1 = new ASIEntities7()) {  
  
    ctx1.Entry(al).State = EntityState.Modified;  
    ctx1.SaveChanges();  
}
```

*O objecto fica “attached”, como se se tivesse executado `ctx1.Alunos.Attach(al)`*



*Esmaga o que estiver na BD.*

*Pode gerar excepções `DBUpdateConcurrencyException`*





# Entity Framework

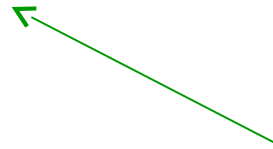
---

## Utilização “desligada”

Aluno al;

...

```
using (var ctx1 = new ASIEntities7()) {  
  
    ctx1.Entry(al).State = EntityState.Modified;  
  
    DbEntityEntry<Aluno> entry = ctx1.Entry<Aluno>(al);  
  
    var databaseValues = entry.GetDatabaseValues();  
    entry.OriginalValues.SetValues(databaseValues);  
  
    ctx1.SaveChanges();  
  
}
```



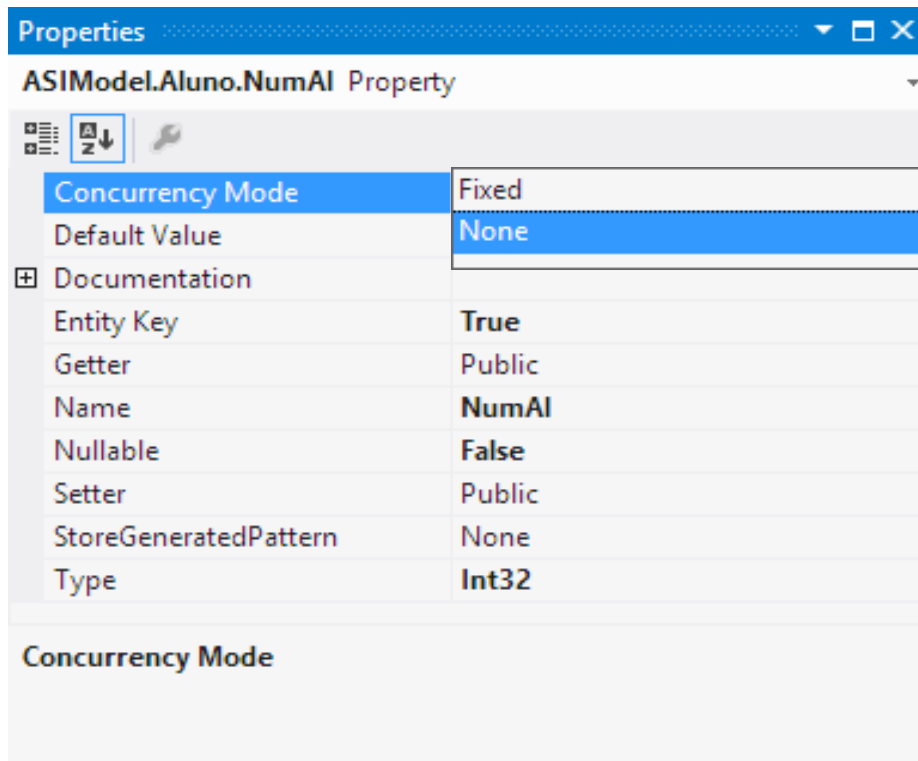
*Esmaga o que estiver na BD.  
Assim já não gera DBUpdateConcurrencyException*

# Entity Framework

---

Controlo de concorrência optimista por omissão não activado.

Para activar, definir Concurrency mode nas propriedades que devem ser testadas.



*Sempre que for possível usar colunas rowversion, fazê-lo e testar concorrência apenas com elas*

# Entity Framework

---

Lidar com erros de concorrência:

```
bool falha;
do
{
    falha = false;
    try { ctx1.SaveChanges(); }
    catch (DbUpdateConcurrencyException e)
    {
        falha = true;
        // ignorar as alterações feitas no contexto e usar a informação corrente na
        // BD (estado = unchanged)
        e.Entries.Single().Reload();
    }

} while (falha);
```

# Entity Framework

---

Lidar com erros de concorrência:

```
bool falha;
do
{
    falha = false;
    try { ctx1.SaveChanges(); }
    catch (DbUpdateConcurrencyException e)
    {
        falha = true;
        // esmagar as alterações na BD
        var entry = e.Entries.Single();
        var dbValues = entry.GetDatabaseValues();
        entry.OriginalValues.SetValues(dbValues);
    }

} while (falha);
```

# Entity Framework


---

## Lidar com erros de concorrência:

```
bool falha;
do
{
    falha = false;
    try { ctx1.SaveChanges(); }
    catch (DbUpdateConcurrencyException e)
    {
        falha = true;
        // deixar que a aplicação cliente decida
        var entry = e.Entries.Single();
        var dbValues = entry.GetDatabaseValues();
        entry.CurrentValues.SetValues( MergeValues(entry.OriginalValues,
                                                    entry.CurrentValues,
                                                    dbValues));

        entry.OriginalValues.SetValues(dbValues);
    }
} while (falha);
```

*Um método que fará a conciliação das duas versões*



*MergeValues(entry.OriginalValues,  
entry.CurrentValues,  
dbValues));*

# Entity Framework

---

## Queries sem associação ao contexto:

```
using (var ctx = new ASIEntities7()) {  
    var al = (from a in ctx.Alunos.AsNoTracking()  
              where a.NumAl == 1111  
              select a)  
              .SingleOrDefault();  
  
    Console.WriteLine(al.Nome);  
    al.Nome = "zeze";  
  
    ctx.SaveChanges();  
}
```

# Entity Framework

---

## Execução de queries SQL:

```
using (var ctx = new ASIEntities7()) {  
    var sqlquery = ctx.Alunos.SqlQuery("select * from Alunos");  
  
    foreach (var a in sqlquery)  
        Console.WriteLine(a.Nome);  
  
}
```

*Só permite acessos a Alunos.*



*O tipo devolvido é Aluno*



*(objectos “attached”)*

# Entity Framework

---

## Execução de queries SQL:

```
using (var ctx = new ASIEntities7()) {  
  
    var sqlquery = ctx.Database.SqlQuery<Aluno>("select * from Alunos");  
  
    foreach (var a in sqlquery)  
        Console.WriteLine(a.Nome);  
  
}
```

*(objectos não “attached”)*



# Entity Framework

---

```
using (var ts = new TransactionScope()) {  
  
    using (var ctx = new ASIEntities7())  
    {  
        var al = (from a in ctx.Alunos  
                   where a.NumAl == 1111  
                   select a)  
                   .SingleOrDefault();  
  
        al.Nome = "xico ze";  
  
        ctx.SaveChanges();  
    }  
    ts.Complete();  
}
```

*As conexões Entity Framework alistam-se  
na transacção TransactionScope*

# Entity Framework

---

```
using (var ts = new TransactionScope()) {  
    using (var ctx = new ASIEntities7()) {  
        ctx.Database.Connection.Open();  
        var al = (from a in ctx.Alunos  
                  where a.NumAl == 1111  
                  select a)  
                  .SingleOrDefault();  
        al.Nome = "xico";  
        using (var ctx1 = new ASIEntities7()) {  
            ctx1.Database.Connection.Open();  
            var al1 = (from a in ctx1.Alunos  
                      where a.NumAl == 4444  
                      select a)  
                      .SingleOrDefault();  
            al1.Nome = "xxxx";  
            ctx1.SaveChanges();  
        }  
        ctx.SaveChanges();  
    }  
    ts.Complete();  
}
```

**A transacção é distribuída**

*Pode fazer sentido quando se fazem sequências longas de interrogações*

# Entity Framework

```
EntityConnection cn = new EntityConnection("name = ASIEntities7");  
cn.Open();
```

```
using (var ts = new TransactionScope()) {  
    using (var ctx = new ASIEntities7(cn)) {
```

```
        var al = (from a in ctx.Alunos  
                    where a.NumAl == 1111  
                    select a)  
                    .SingleOrDefault();  
        al.Nome = "aaaa";  
        using (var ctx1 = new ASIEntities7(cn)) {  
            var al1 = (from a in ctx1.Alunos  
                        where a.NumAl == 4444  
                        select a)  
                        .SingleOrDefault();  
            al1.Nome = "bbbb";  
            ctx1.SaveChanges();  
        }  
        ctx.SaveChanges();  
    }  
    ts.Complete();  
}
```

Acrescentar construtor

```
public ASIEntities7(System.Data.Common.DbConnection cn)  
    : base(cn,false) {  
}  
a ASIEntities7
```

# Entity Framework

---

```
using (var ts = new TransactionScope(T)) {  
    using (var ctx = new ASIEntities7()) {  
        var al = (from a in ctx.Alunos  
                    where a.NumAl == 1111  
                    select a)  
                    .SingleOrDefault();  
        al.Nome = "xxxx";  
        using (var ctx1 = new ASIEntities7(ctx.Database.Connection)) {  
            var al1 = (from a in ctx1.Alunos  
                        where a.NumAl == 4444  
                        select a)  
                        .SingleOrDefault();  
            al1.Nome = "yyyy";  
            ctx1.SaveChanges();  
        }  
        ctx.SaveChanges();  
    }  
    ts.Complete();  
}
```

Acrescentar construtor

```
public ASIEntities7(System.Data.Common.DbConnection cn)  
    : base(cn,false) {  
}  
a ASIEntities7
```

---

# Bibliografia

*Julia Lerman, Rowan Miller, Programming Entity Framework:DbContext, O'Reilly Media, Inc., 2012 (DbContext)*

*Stefano Mostarda, Marco De Sanctis, Daniele Bochicchio, Entity Framework 4 in Action, Manning Publications Co., 2011 (ObjectContext)*

*Entity Framework 6 API,  
[http://msdn.microsoft.com/en-us/library/system.data.entity\(v=vs.113\).aspx](http://msdn.microsoft.com/en-us/library/system.data.entity(v=vs.113).aspx)  
em 2013/11/27 (ainda para a versão beta)*