# Dell IT Academy

# ENTITY FRAMEWORK CORE

Operações CRUD

# Contexto

- EFCore segue os padrões Data Mapper, Repository e Unit of Work
- Objeto **DbContext** é baseado nesses padrões e possui uma API para
  - Gerenciar objetos em memória (inclusive com cache)
  - Manter a ligação entre o banco de dados e as entidades mapeadas no modelo relacional
  - Gerenciar a conexão com a base de dados
  - Gerenciar o contexto transacional

# Contexto

- Objeto **DbSet**
  - Representa uma coleção de entidades em um contexto de persistência
  - É obtida a partir do *DbContext*
  - Provê métodos para operações CRUD sobre um determinado tipo de entidade

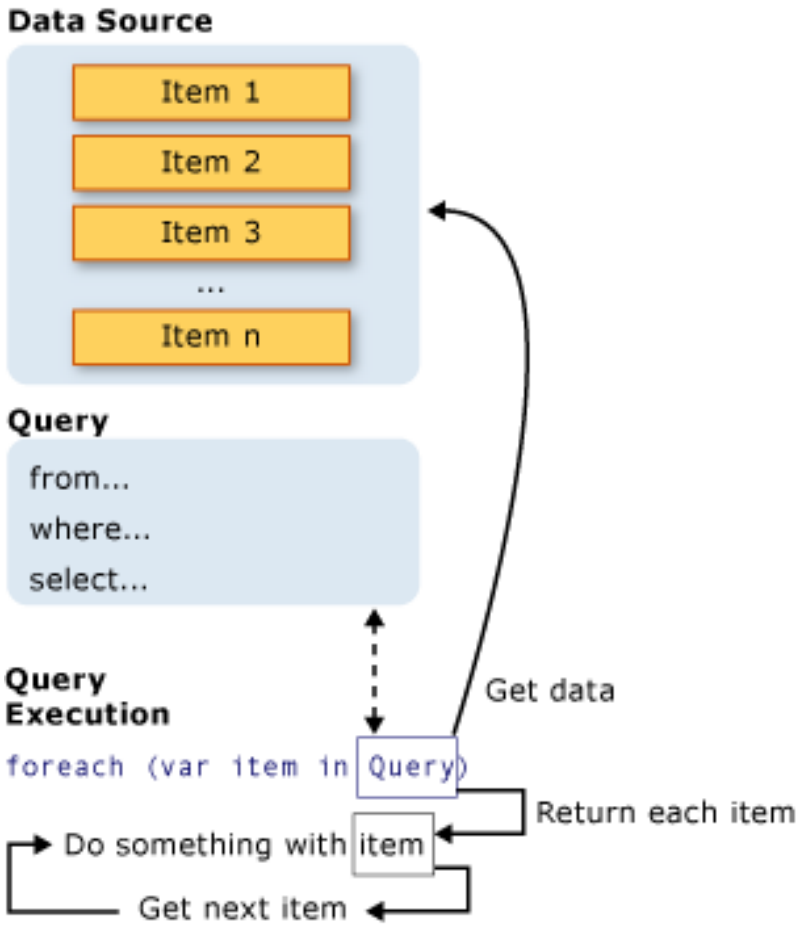
# CRUD – Modelo exemplo

```
CREATE TABLE [dbo].[Genres] (
    [GenreID]     INT            IDENTITY (1, 1) NOT NULL,
    [Description] NVARCHAR (MAX) NULL,
    [Name]        NVARCHAR (MAX) NULL,
    CONSTRAINT [PK_Genres] PRIMARY KEY CLUSTERED ([GenreID] ASC)
);


CREATE TABLE [dbo].[Movies] (
    [ID]          INT            IDENTITY (1, 1) NOT NULL,
    [Director]    NVARCHAR (MAX)  NULL,
    [GenreID]     INT            NOT NULL,
    [Gross]       DECIMAL (18, 2) NOT NULL,
    [Rating]      FLOAT (53)      NOT NULL,
    [ReleaseDate] DATETIME2 (7)   NOT NULL,
    [Title]       NVARCHAR (MAX)  NULL,
    CONSTRAINT [PK_Movies] PRIMARY KEY CLUSTERED ([ID] ASC),
    CONSTRAINT [FK_Movies_Genres_GenreID] FOREIGN KEY ([GenreID]) REFERENCES
[dbo].[Genres] ([GenreID]) ON DELETE CASCADE
);
```

# CRUD - Consultas

- Consultas são realizadas via LINQ – *Language Integrated Query*



# CRUD – Inserções e Alterações

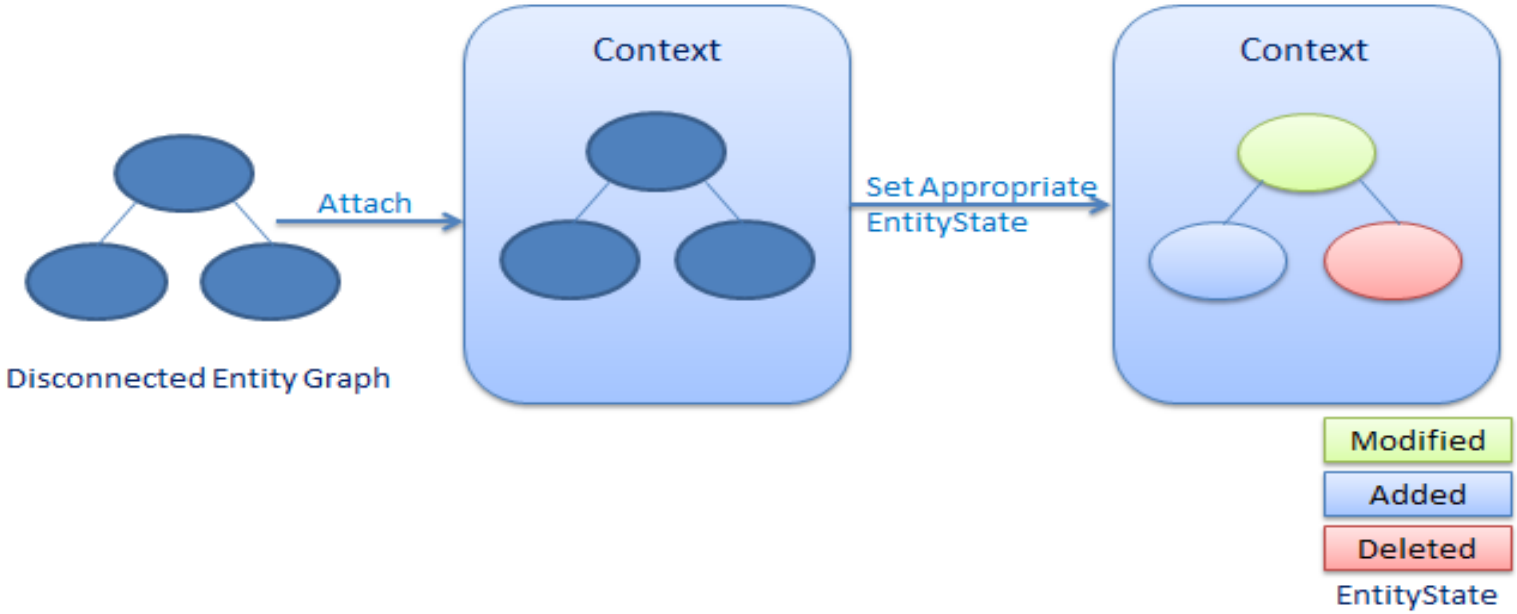
- Cada contexto possui um objeto *ChangeTracker* responsável por manter as informações de alterações sobre os objetos gerenciados
  - *DbSet.Add* para adicionar instâncias
  - *DbSet.Remove* para remover instâncias
- Alterações são persistidas através da operação *SaveChanges()*
  - CUIDADO: a maioria dos provedores implementam a operação de forma TRANSACIONAL de forma automática
  - Operações em cascata podem ser realizadas

# CRUD – Controle de Concorrência

- EFCore utiliza o modelo de controle de concorrência otimista (nenhum *lock* do BD é realizado automaticamente)
- Provê mecanismos de detecção e resolução de conflitos
- Ver exemplos em https://docs.microsoft.com/en-us/ef/core/saving/concurrency

# CRUD – Ambientes Desconectados

- Será necessário anexar (attach) à instância do contexto e configurar as alterações realizadas sobre as entidades.
- A enumeração *EntityState* é utilizada para sinalizar as alterações realizadas.



# CRUD – Ambientes Desconectados

## Comandos da classe DbSet

- ***DbSet.Add()*** – adiciona toda a estrutura ao contexto com a propriedade *Added* (já utilizado no modelo conectado)

- ***DbSet.Attach()*** – adiciona toda a estrutura ao contexto com a propriedade *Unchanged* configurada para todas as entidades e o usuário deve configurar cada elemento individualmente

- ***DbContext.Entry()*** – adiciona toda a estrutura a partir do estado informado

```
DbContext.Entry(disconnectedEntity).state =
        EntityState.Added/Modified/Deleted/Unchanged
```

# CRUD – Ambientes Desconectados

```
var novoFilme = new Movie();
novoFilme.Title= "La La Land";
//...

//create DBContext object
using (var contexto = new MovieContext())  {

    // adicionar ao contexto
    contexto.Entry(novoFilme).State =
System.Data.Entity.EntityState.Added;

   // persistir
   contexto.SaveChanges();
  }
```