

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

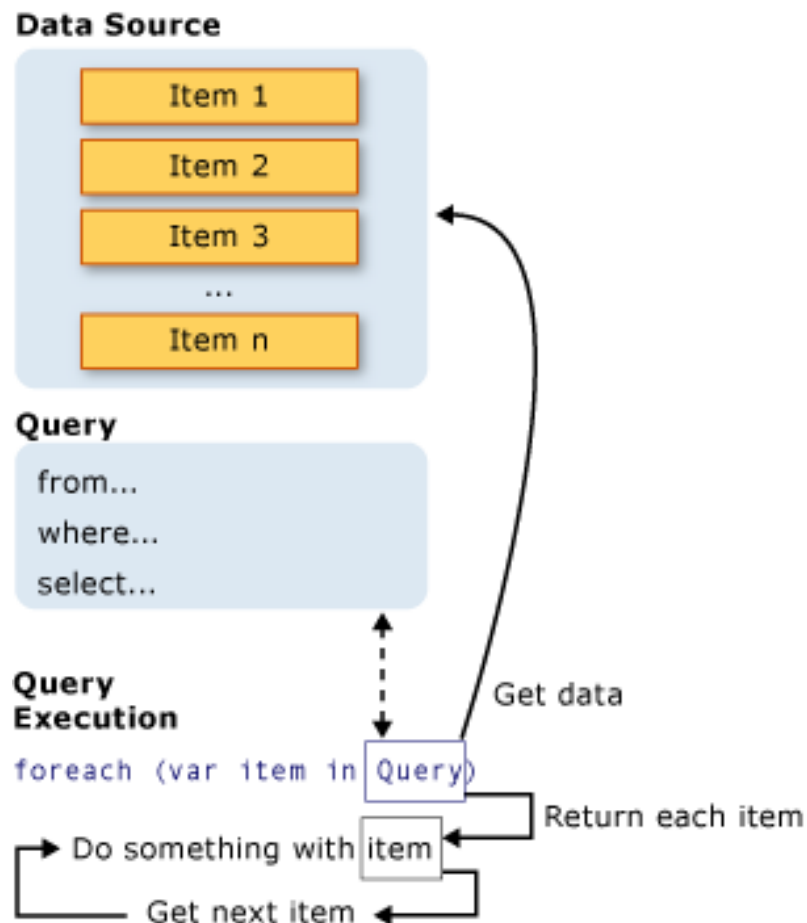
CRUD - Modelo exemplo

```
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 - Consultas

- Consultas podem incluir dados relacionados a outras entidades
- Três formas de tratar os relacionamentos:
 - Eager loading – carregar dados associados como parte da consulta inicial
 - Explicit loading – carregar dados associados de forma explícita quando desejado
 - Lazy loading – carregar dados associados de forma transparente quando necessários
- Ver exemplos em <https://docs.microsoft.com/en-us/ef/core/querying/related-data/>

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
- Para desabilitar explicitamente o mecanismo de tracking, utiliza-se o método *.AsNoTracking()* sobre o objeto *DbSet*
 - Otimização para contextos de consultas somente para leitura de dados

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
- Para desabilitar explicitamente o mecanismo de tracking, utiliza-se o método *.AsNoTracking()* sobre o objeto *DbSet*
 - Otimização para contextos de consultas somente para leitura de dados

CRUD - Controle de Transações

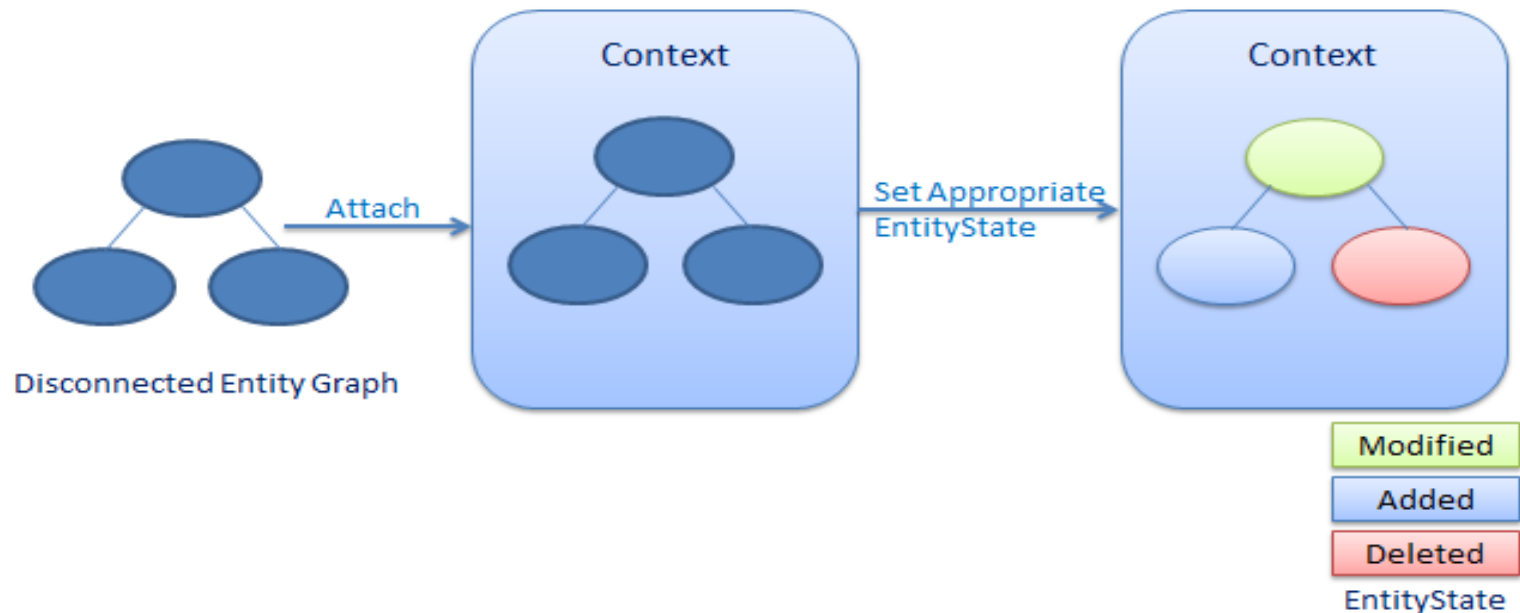
- Uma transação permite que diversas operações sejam processadas de forma atômica
 - Se todas operações foram executadas com sucesso, dizemos que a transação realizou um *commit*
 - Se alguma operação falhou, então todas devem ser desfeitas, dizemos que a transações realizou um *rollback*
- A operação *SaveChanges()* é transacional
 - Modificar a configuração e tratar manualmente uma transação se for um requisito do sistema
- Ver exemplos em <https://docs.microsoft.com/en-us/ef/core/saving/transactions>

CRUD - Controle de Concorrência

- EFCore utiliza o modelo de controle de concorrência otimista
 - Utiliza *concurrency tokens*
 - 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*
- ***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()*** – retorna a referência à estrutura do objeto conhecido pelo contexto

DbContext.Entry(entidadeNoContexto).CurrentValues.SetValues(entidadeForaContexto)

Ver exemplos em <https://docs.microsoft.com/en-us/ef/core/saving/disconnected-entities>