

## Mapeando chaves primárias compostas

### Transcrição

Para termos a tabela de JOIN , criaremos uma classe que representará esse relacionamento e será reconhecido pelo Entity. A classe se chamará `PromocaoProduto` .

```
namespace Alura.Loja.Testes.ConsoleApp
{
    public class PromocaoProduto
    {
    }
}
```

A classe terá as propriedades `ProdutoId` , `Produto` , `PromocaoId` e `Promocao` :

```
namespace Alura.Loja.Testes.ConsoleApp
{
    public class PromocaoProduto
    {
        public int ProdutoId { get; set; }
        public Produto produto { get; set; }
        public int PromocaoId { get; set; }
        public Promocao Promocao { get; set; }
    }
}
```

Na classe `Promocao` , mudaremos o tipo da lista `IList<Produto>` para `IList<PromocaoProduto>` :

```
namespace Alura.Loja.Testes.ConsoleApp
{
    public class Promocao
    {
        public int Id { get; set; }
        public string Descricao { get; internal set; }
        public DateTime DataInicio { get; internal set; }
        public DateTime DataTermino { get; internal set; }
        public IList<PromocaoProduto> Produtos { get; internal set; }
    }
}
```

Na classe `Produto` também mudaremos o tipo da lista, de `IList<Promocao>` para `IList<PromocaoProduto>` :

```
namespace Alura.Loja.Testes.ConsoleApp
{
    public class Produto
    {
        public int Id { get; internal set; }
        public string Nome { get; internal set; }
        public decimal Preco { get; internal set; }
        public IList<PromocaoProduto> Promocoes { get; internal set; }
    }
}
```

```

public int Nome { get; internal set; }
public int Categoria { get; internal set; }
public int PrecoUnitario { get; internal set; }
public string Unidade { get; set; }
public IList<PromocaoProduto> Promocoes { get; set; }

public override string ToString()
{
    return $"Produto: {this.Id}, {this.Nome}, {this.Categoria}, {this.PrecoUnitario}";
}
}
}

```

Antes de adicionarmos a migração, é necessário comentarmos o trecho de código onde adicionamos um novo produto na promoção que simulamos na classe `Program` :

```

//promocaoDePascoa.Produtos.Add(new Produto());
//promocaoDePascoa.Produtos.Add(new Produto());
//promocaoDePascoa.Produtos.Add(new Produto());

```

Tentaremos adicionar a migração com o comando `Add-Migration Promocao` . Receberemos um erro informando que é necessário um **Id** para ser a chave primária da classe `PromocaoProduto` . Mas será que faz sentido? Não, por ser uma classe de `JOIN` .

Podemos criar uma **chave primária composta**, que será identificada pelo `ProdutoId` e `PromocaoId` . Na classe `LojaContext` iremos sobrescrever outro método, chamado `OnModelCreating()` .

Esse método é executado no evento de criação do modelo. Podemos configurar informado que a entidade `PromocaoProduto` **tem a chave composta**, com a composição de `ProdutoId` e `PromocaoId` . A configuração ficará da seguinte maneira:

```

namespace Alura.Loja.Testes.ConsoleApp
{
    public class LojaContext : DbContext
    {
        public DbSet<Produto> Produtos { get; set; }
        public DbSet<Compra> Compras { get; set; }
        public DbSet<Promocao> Promocoes { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder
                .Entity<PromocaoProduto>()
                .HasKey(pp => new { pp.PromocaoId, pp.ProdutoId });
            base.OnModelCreating(modelBuilder);
        }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=LojaDB;Trusted_Conn
        }
    }
}

```

Se executarmos o comando `Add-Migration Promocao`, a migração será criada com sucesso. Analisando a **classe de migração** `Promocao`, veremos que o no método `Up()` será criado as tabelas `Promocoes` e `PromocaoProduto`. Detalhe que na tabela `JOIN`, possui a chave composta, com as chaves estrangeiras de `Produtos` e `Promocoes`. Assim se deletarmos um produto ou uma promoção, os registros da tabela `PromocaoProduto` serão deletados em cascata.

Era isso que queríamos. Executaremos o comando `Update-Database` para que as evoluções da sistema sejam aplicadas no banco de dados. Também será adicionado na tabela de registros, que essa migração foi executada.