03

3 - Linq to entities groupby

Transcrição

Agora o pedido do cliente é criar um relatório para listar os álbuns mais vendidos de um artista. Para fazer isso vamos começar pelos itens de nota fiscal. Conforme vimos, as vendas iniciam pela nota fiscal. Portanto, para realizar o pedido do cliente vamos introduzir uma `query` no arquivo `Program.cs` e que puxe o item nota fiscal, a `var query = from inf in contexto.ItemsNotaFiscal` . Introduzimos também o `select inf` e depois adicionamos o `foreach(var inf in collection)` , este último significa: "para cada nota fiscal na `query` será listado o título do álbum, o nome da faixa e a quantidade vendida para o item". Ainda, acrescentamos a formatação e temos:

```
Console.WriteLine("{0}\t{1}\t{2}\ t{3}", inf.Faixa.Album.Título, inf.Faixa.Nome, inf.Quantidade, inf
```

O resultado é o seguinte:

```
var query = from inf in contexto.ItemsNotaFiscal
            select inf;

foreach (var inf in query)
{
    Console.WriteLine("{0}\t{1}\t{2}\ t{3}", inf.Faixa.Album.Título, inf.Faixa.Nome, inf.Quantidade,

}

Console.ReadKey();
```

Rodando isso, teremos um resultado que traz todos os itens de nota fiscal e para cada item o nome do álbum, o nome da faixa, a quantidade e o preço unitário.

Agora, vamos aplicar o filtro, portanto, acrescentaremos a cláusula `where` e junto disso, iremos adicionar os dados: nota fiscal, faixa, álbum, artista e nome. O `where` será igual a `Led Zeppelin` :

```
where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
```

O código ficará assim:

```
var query = from inf in contexto.ItemsNotaFiscal
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
            select inf;

foreach (var inf in query)
{
    Console.WriteLine("{0}\t{1}\t{2}\ t{3}", inf.Faixa.Album.Título, inf.Faixa.Nome, inf.Quantidade,

}

Console.ReadKey();
```

Rodando a aplicação temos o seguinte resultado:

```
Led Zeppelin III        Since I've Been Loving You      1       0.99
Led Zeppelin III        Out On The Tiles        1       0.99
Led Zeppelin III        That's The Way  1       0.99
Led Zeppelin III        Bron-Y-Aur Stomp        1       0.99
Led Zeppelin III        Bron-Y-Aur Stomp        1       0.99
Led Zeppelin III        Hats Off To (Roy) Harper        1       0.99
Physical Graffiti [Disc 2]      Down By The Seaside     1       0.99
Physical Graffiti [Disc 2]      Ten Years Gone  1       0.99
Physical Graffiti [Disc 2]      Night Flight    1       0.99
Physical Graffiti [Disc 2]      Boogie With Stu 1       0.99
Physical Graffiti [Disc 2]      Sick Again      1       0.99
Presence        Achilles Last Stand     1       0.99
Presence        For Your Life   1       0.99
Presence        For Your Life   1       0.99
Presence        Hots On For Nowhere     1       0.99
Presence        Tea For One     1       0.99
The Song Remains The Same (Disc 1)      Rock & Roll     1       0.99
The Song Remains The Same (Disc 1)      Celebration Day 1       0.99
The Song Remains The Same (Disc 1)      Dazed And Confused      1
The Song Remains The Same (Disc 2)      No Quarter      1       0.99
The Song Remains The Same (Disc 2)      Stairway To Heaven      1
```

O resultado é o nome do álbum, o nome da faixa, a quantidade e o preço unitário. O único problema é que a formatação está esquisita, para arrumá-la deixaremos cada item inserido no `Console` disposto em uma linha e depois do `Titulo`, vamos adicionar o `PadRight()`. Passamos para ele um valor que equivale a quantidade de caracteres, `25`. Usaremos o `inf.Faixa.Nome` o `PadRight()` também com o valor de `25`. Teremos:

```
var query = from inf in contexto.ItemsNotaFiscal
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
            select inf;

foreach (var inf in query)
{
    Console.WriteLine("{0}\t{1}\t{2}\ t{3}",
        inf.Faixa.Album.Titulo.PadRight(40)inf.Faixa.Nome.PadRight(25), inf.Quantidade,
        inf.PrecoUnitario);

}

Console.ReadKey();
```

Rodando isso temos os elementos alinhados:

```
Led Zeppelin III                    Immigrant Song              1       0.99
Led Zeppelin III                    Friends                     1       0.99
Led Zeppelin III                    Celebration Day             1       0.99
Led Zeppelin III                    Since I've Been Loving You  1       0.99
Led Zeppelin III                    Out On The Tiles            1       0.99
Led Zeppelin III                    That's The Way              1       0.99
Led Zeppelin III                    Bron-Y-Aur Stomp            1       0.99
Led Zeppelin III                    Bron-Y-Aur Stomp            1       0.99
Led Zeppelin III                    Hats Off To (Roy) Harper    1       0.99
Physical Graffiti [Disc 2]          Down By The Seaside         1       0.99
Physical Graffiti [Disc 2]          Ten Years Gone              1       0.99
Physical Graffiti [Disc 2]          Night Flight                1       0.99
Physical Graffiti [Disc 2]          Boogie With Stu             1       0.99
Physical Graffiti [Disc 2]          Sick Again                  1       0.99
Presence                            Achilles Last Stand         1       0.99
Presence                            For Your Life               1       0.99
```

Agora, temos que agrupar as vendas por álbum. No `SQL` isso é feito utilizando a cláusula `group by`. Entretanto, a `sintaxe` é diferente no `LINQ`! Assim, vamos inserir o `group` junto do `ItemsNotaFiscal`. Utilizaremos também a variável `inf` antes do `by` que é obrigatório e, por fim, iremos inserir a propriedade:

```
group inf by inf.Faixa.Album
```

O interessante é que na sintaxe do `SQL` o `group by` é utilizável com valores variados, não só com valores escalares, assim, é possível agrupar também por objetos. Por exemplo, vamos escrever o agrupamento por `Album` e teremos:

```
group inf by inf.Faixa.Album
```

Após o agrupamento é preciso dizer onde vamos colocar o valor agrupado. Portanto, faz-se necessário adicionar uma variável que receba os valores agrupados, no caso, usar o `into` e depois a variável `agrupado`. Assim, teremos o `group`, `by` e `into` que são obrigatórios na sintaxe do `LINQ:

```
var query = from inf in contexto. ItemsNotaFiscal
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
            group inf by inf.Faixa.Album into agrupado
            select inf;
```

Ao adicionarmos o `agrupado`, o `select inf` não ficará mais disponível para ser utilizado. Então, vamos usaremos apenas a variável `agrupado` - que equivale ao objeto `Album`. A partir do `agrupado`, acessaremos uma chave, assim, adicionamos `Key` e junto disso colocamos também o `Título`. Dessa maneira teremos:

```
select agrupado.Key.Título
```

Para que a visualização fique mais fácil, criaremos um objeto anônimo, portanto, adicionaremos o `select new` para englobar o `agrupado.Key.Título`. Falta ainda adicionar o nome da propriedade:

```
TituloDoAlbum = agrupado.Key.Título
```

Ficaremos com:

```
var query = from inf in contexto. ItemsNotaFiscal
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
```

```
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
            group inf by inf.Faixa.Album into agrupado
            select new
            {
                TituloDoAlbum = agrupado.Key.Título
            };
```

Vamos alterar também o `foreach`, portanto, trocaremos o `inf` que se refere ao `ItemsNotaFiscal` e iremos hamá-lo de `agrupado`. Como a sintaxe foi partida, vamos reorganizar o que está no `Console.WriteLine()`. Desta forma, apagaremos a seguinte linha:

```
inf.Faixa.Album.Título
```

Depois, vamos substitui-la por:

```
agrupado.TituloDoAlbum
```

Teremos:

```
foreach (var agrupado in query)
{

Console.WriteLine("{0}\t{1}\t{2}t{3}",
    agrupado.TituloDoAlbum.PadRight(40),
    inf.Quantidade,
    inf.PrecoUnitario);
}
```

Agora, para obter os valores totais é preciso multiplicar a `quantidade` por `PrecoUnitario`. Desta forma, adicionamos junto do objeto anônimo `select new` o `TotalPorAlbum` que vamos equivaler a `agrupado.Sum`, pois é a partir do `agrupado` que conseguimos acessar o método `Sum()`. Dentro disto, colocaremos a expressão que será utilizada para chegar ao valor total dos itens. Por isso, escreveremos:

```
a => a.Quantidade * a.PrecoUnitario
```

Uma vez que a expressão é introduzida é preciso substituir o `inf.Quantidade` por `agrupado.Total.PorAlbum`. Ainda, será preciso alterar a formatação para apenas `"{0}\t{1}"` :

```
var query = from inf in contexto. ItemsNotaFiscal
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
            group inf by inf.Faixa.Album into agrupado
            select new
            {
                TituloDoAlbum = agrupado.Key.Título
                TotalPorAlbum = agrupado.Sum(a => a.Quantidade * a.PrecoUnitario)
            };

foreach (var agrupado in query)
{
```

```
Console.WriteLine("{0}\t{1}",
    agrupado.TituloDoAlbum.PadRight(40),
    agrupado.TotalPorAlbum);
}
```

Ao rodarmos o código, o resultado já aparecerá, mas nosso desejo é que ele seja mostrado na ordem de vendas. Para fazer isso, vamos utilizar o `orderby` que inserimos abaixo do `group`. Assim, vamos adicionar o `orderby` e ao lado, o `agrupado.Sum()`. Também passaremos a seguinte expressão:

```
agrupado.Sum(a => a.Quantidade * a.PrecoUnitario)
```

Teremos:

```
var query = from inf in contexto. ItemsNotaFiscal
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
            group inf by inf.Faixa.Album into agrupado
            orderby agrupado.Sum(a => a.Quantidade * a.PrecoUnitario)
            select new
            {
                TituloDoAlbum = agrupado.Key.Título
                TotalPorAlbum = agrupado.Sum(a => a.Quantidade * a.PrecoUnitario)
            };
```

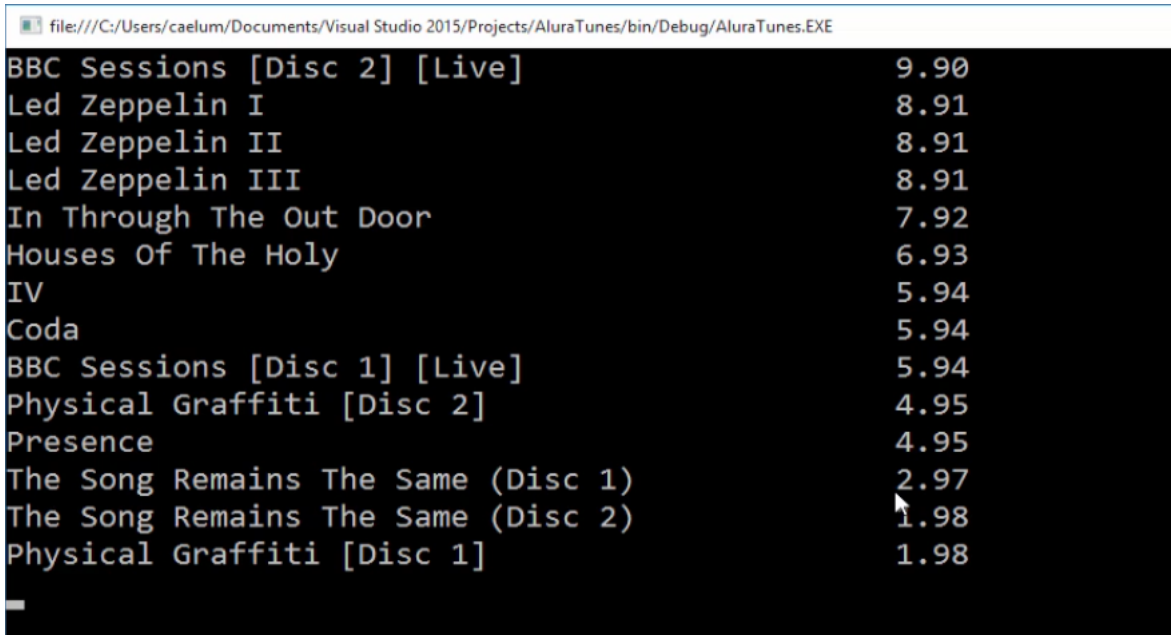
Ao rodar o código, teremos o seguinte:

```
BBC Sessions [Disc 1] [Live]             5.94
Physical Graffiti [Disc 1]               1.98
BBC Sessions [Disc 2] [Live]             9.90
Coda                                     5.94
Houses Of The Holy                       6.93
In Through The Out Door                  7.92
IV                                       5.94
Led Zeppelin I                           8.91
Led Zeppelin II                          8.91
Led Zeppelin III                         8.91
Physical Graffiti [Disc 2]               4.95
Presence                                 4.95
The Song Remains The Same (Disc 1)       2.97
The Song Remains The Same (Disc 2)       1.98
```

Podemos alterá-lo para que os elementos sejam mostrados de maneira decrescente usando o `descending`:

```
var query = from inf in contexto. ItemsNotaFiscal
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
            group inf by inf.Faixa.Album into agrupado
            orderby agrupado.Sum(a => a.Quantidade * a.PrecoUnitario)
                descending
```

O resultado será o seguinte:



Ou seja, alcançamos o resultado perfeito! Vamos observar o código para verificar se algo nele ainda está problemático. Observe o todo:

```
var query = from inf in contexto. ItemsNotaFiscal
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
            group inf by inf.Faixa.Album into agrupado
            orderby agrupado.Sum(a => a.Quantidade * a.PrecoUnitario)
                descending
            select new
            {
                TituloDoAlbum = agrupado.Key.Título,
                TotalPorAlbum = agrupado.Sum(a => a.Quantidade * a.PrecoUnitario)
            };
foreach (var agrupado in query)

Console.WriteLine("{0}t{1},
    agrupado.TituloDoAlbum.PadRight(40),
    agrupado.TotalPorAlbum);
)
```

Note que ocorre a repetição desse trecho:

```
agrupado.Sum(a => a.Quantidade * a.precoUnitario)
```

Quando esse tipo de redundância sucede a praxe é englobar as expressões em uma variável. Assim, é preciso utilizar outra cláusula chamada `let` que permite criar variáveis internas em uma consulta `LINQ` , passíveis de serem utilizadas em outros setores.

Por exemplo, podemos usar uma mesma variável em distintos locais. Vamos fazer isso para que a seguinte expressão não se repita:

```
agrupado.Sum(a => a.Quantidade * a.precoUnitario
```

Transformaremos essa expressão em uma variável `vendasPorAlbum` :

```
let vendasPorAlbum = agrupado.Sum(a => a.Quantidade * a.PrecoUnitario)
```

Junto, colorecamos o `orderby` e a variável `vendasPorAlbum`. Repetiremos o procedimento no `TotalPorAlbum`:

```
var query = from inf in contexto. ItemsNotaFiscal
            where inf.Faixa.Album.Artista.Nome == "Led Zeppelin"
            group inf by inf.Faixa.Album into agrupado
            let vendasPorAlbum = agrupado.Sum(a => a.Quantidade * a.PrecoUnitario)
            orderby vendasPorAlbum
                descending
            select.new
            {
                TituloDoAlbum = agrupado.Key.Título,
                TotalPorAlbum = vendasPorAlbum
            };
```

Rodando isso novamente, teremos o mesmo resultado, mas evitando a redundância!