03

3 - Conclusao

Transcrição

Aqui, encerramos o curso de **LINQ: Crie queries poderosas em C# parte 2**.

Faremos uma breve retrospectiva do que vimos ao longo das aulas:

1) Relatório de Vendas Paginado

- Quebramos um relatório grande em partes menores.
- `Take()` : utilizado para pegar uma quantidade limite de linhas
- Calculando saltos: vimos como calcular números específicos de linhas que devem ser pulados para atingir um ponto exato do relatório
- Função `Math.Ceiling` : função secundária que calcula o valor exato de páginas que existem no relatório

2) Subconsultas

- Conceito de subconsulta: trazer uma consulta menor dentro de uma maior
- A subconsulta simplifica a leitura do código
- Extração de subconsultas: traz a subconsulta dentro de uma variável de consulta, assim, é possível utilizar a variável de consulta em uma consulta
- A subsonculta pode ser utilizada em vários pontos de uma consulta `LINQ` . Por exemplo, na cláusula `where` , `select` e `groupby`

3) Clientes que compraram os produtos mais vendidos

- Projeção de Dados é aquilo que utiliza uma origem de dados, tabela ou lista. Fizemos uso da projeção junto com a cláusula `select` para transformar uma informação em outra informação. Por exemplo, calculamos o total de uma nota fiscal por meio da multiplicação de um valor unitário pela quantidade
- Para obter os produtos mais vendidos usamos uma consulta `LINQ` que utiliza um método de soma, o `Sum()`
- Somamos as vendas por cada produto e com isso aprendemos que o resultado pode ser um erro, uma vez que alguns produtos não possuem nenhuma venda o que quebra a execução de uma consulta gerando uma exceção. Vimos como solucionar tal situação
- Usamos uma variável interna e reutilizamos ela em uma consulta `LINQ` para facilitar tanto a visualização quanto a leitura do código
- Para definir essa variável interna utilizamos o operador `Let` .
- Aprendemos também a pegar o resultado de uma query e a utilizá-lo em outra consulta secundária

4) Análise de afinidades

- Análise de Afinidade (ou Análise de Carrinho) corresponde aos produtos sugeridos por determinada página ao efetuarmos uma busca para comprar um objeto específico
- A relação da análise de afinidade se resume a:

" Quem comprou produto X Também comprou produto Y"

- Na prática para criar esse tipo de análise utilizamos o `self join`

5) Execução Adiada x Execução Imediata

- Vimos a diferença entre uma execução adiada e uma execução imediata. Uma consulta `LINQ` não é necessariamente executada quando é criada, mas pode ser executada em alguns momentos específicos.
- Uma consulta `LINQ` após declarada pode ser afetada pela mudança não só da origem de dados, como também devido a alteração de uma variável de memória na qual essa consulta depende. Então, quando modificamos uma variável em memória e esta é utilizada pela consulta `LINQ` , veremos que ocorre uma modificação
- O **cache** serve para guardar o resultado de uma consulta `LINQ` que não muda, assim, poderemos guardar os valores de configuração do sistema em uma variável em memória e dessa maneira não é preciso executar a consulta a todo instante

6) LINQ Paralelo

- QRCode: é um requisito da área de Marketing, uma espécie de código de barra, mas capaz de guardar mais informações.
- O `QR Code` produz imagens a partir de informações, processo que demora um pouco para concretizar-se. Desta maneira, utilizamos o `LINQ` com o recurso de paralelismo para agilizar a tarefa.
- Como os arquivos são normalmente salvos em série, aproveitamos um recurso do `LINQ` , o `ForAll` , para quebrar as tarefas em blocos e salvar vários arquivos ao mesmo tempo.

7) LINQ to Entities e Stored Procedures

- O `LINQ` do `Entities` não é limitado apenas a Entidades ou tabelas do banco de dados
- Ao em vez de fazer uma portabilidade do `LINQ` do `Entities` para o `C#` , vamos utilizar a `Stored Procedure` como origem da dados, como se fosse uma entidade do `Entity Framework` , uma lista ou tabela em memória
- Aprendemos que é preciso atualizar o modelo do `Entity Framework` para que ele possa refletir o objeto disponível
- Criamos uma consulta `LINQ` utilizando `stored procedure` e fizemos isso usando `join` e `group by` . Podemos misturar o `LINQ do Entities` com o `stored procedure` e utilizamos também a `stored procedure` com o `join` , outras entidades e tabelas do sistema
- Vimos que é possível utilizar colunas e propriedades diferentes, ao mesmo tempo, em uma cláusula `groupby` de uma consulta `LINQ`

8) LINQPad

- A parte final do curso foi dedicada a ferramenta `LINQPad` . Uma consulta `LINQ` não é executada somente no **Visual Studio**, ela pode ser executada também no **LINQPad**.
- Começamos a trabalhar no `LINQPad` com expressões simples
- Conseguimos executar códigos oriundos de diferentes linguagens
- O `LINQPad` fornece para nós a possibilidade de visualizar uma sintaxe de método ou sintaxe utilizando `lambda`
- Vimos que o LINQ do XML é suportado pelo LINQPad
- Por fim, configuramos o `LINQPad` para que ele suporte uma consulta `LINQ` do `Entities` o que nos auxilia muito, pois, não é preciso abrir a todo instante o **Visual Studio**. Assim, o `LINQPad` também é utilizado como ferramenta de produtividade