07

Usando Inner Join

Transcrição

[00:00] Fala, aluno. Tudo bom? Nas aulas anteriores, nós vimos como trabalhar com relacionamento entre duas ou mais tabelas. Nós criamos a chave estrangeira e tentamos trazer isso para o lado da nossa aplicação. A abordagem que utilizamos, nós vimos que não foi muito boa, porque nós listamos as categorias e depois nós procuramos os produtos por essas categorias.

[00:24] Com isso, nós vimos que estávamos fazendo muitas requisições para o banco de dados e isso poderia ter um problema de performance futuramente. Então vamos ter que melhorar o nosso código. Primeiro de tudo, eu não quero mais listar todas as categorias para depois buscar os produtos, eu queria, de alguma forma, trazer já as categorias com os produtos, então eu não precisaria mais fazer `new ProdutoDAO(connection).buscar(ct)`.

[00:52] Então, para isso, vamos comentar esse trecho de código e vamos pensar em uma forma de ter um método que vai listar as categorias já com os produtos. Então vou mandar criar esse `listarComProdutos()` na nossa `CategoriaDAO`. Criou, ele vai continuar nos retornando uma lista de categorias, então `public List<Categoria> listarComProdutos)` e um `return null;`. Deixa eu botar o `throws SQLException`, que vamos precisar.

[01:22] Eu vou usar como exemplo esse código do `listar()` categorias, que ele não vai ficar igual, mas nós vamos utilizar algumas coisas dele. Motivação: fazer um *select* em Categoria e já trazer os nossos produtos. Para a nossa sorte, nós já temos algo no SQL que faz isso para nós, ele é chamado de Inner join. O

Inner join, para quem não conhece, já vai entender como funciona mais ou menos essa questão do Inner join.

[01:59] O que importa é que para eu usar o Inner join, eu preciso colocar um *alias* na minha categoria, preciso botar o Inner join, vou botar esse Inner join com produto, que vai ter o *alias* P e tenho que fazer o seguinte - nós já vamos entender essas palavras-chaves. Eu quero os produtos, na verdade eu quero trazer os produtos que estão relacionados com categorias.

[02:38] Isso eu vou representar dessa maneira: `String sql = "SELECT C.ID, C.NOME, P.ID, P.DESCRICAO FROM CATEGORIA C INNER JOIN" + "PRODUTO P ON C.ID = P.CATEGORIA_ID";` . Como nós vamos trabalhar com essa linha? Eu quero trazer o ID da categoria, o nome da categoria, o ID produto, o nome do produto e a descrição do produto.

[03:13] Basicamente eu estou falando o seguinte: traz as informações com ID e nome da categoria, ID e nome e descrição do produto onde existir vínculo entre produto e categoria. Pode não ficar claro aqui agora, olhando a query, mas vamos executar isso no nosso banco de dados. Vou em MySQL 8.0 Command Line Client. Se eu fizer um `SELECT *FROM CATEGORIA;` , eu tenho eletrodomésticos, eletrônicos e outros.

[03:48] `SELECT * FROM PRODUTO` , vamos ver, eu tenho quatro produtos também, eu tenho o notebook, a geladeira, a cômoda e o videogame. A categoria deles, você pode ver, eu estou usando a categoria de eletrônicos, de eletrodoméstico e móveis. A categoria outros não está vinculada, não tem nenhum produto que pertence à categoria outros.

[04:14] Quando executo aquela query com Inner join, que vocês já vão entender, `SELECT * FROM CATEGORIA C INNER JOIN PRODUTO P ON C.ID = P.CATEGORIA_ID` , eu vou trazer só os que têm vínculo realmente. Você vê, eu não tenho nenhum produto que esteja vinculado à categoria de outros, então ela nem aparece para mim.

[04:53] Essa é a ideia do Inner join, trazer só os que estão juntos, os que estão vinculados entre categorias e produto. Então essa é a ideia da nossa query, essa é a ideia do nosso Inner join. Com isso, eu já consigo melhorar a nossa chamada na nossa aplicação. Vou dar um "Run As > Java Application", não é para ter nenhum problema. Vamos ver. Ele trouxe as categorias, está listando apenas uma vez.

[05:26] Mas esse já era um cenário que era esperado. Quando listávamos apenas categorias, nós só íamos uma vez no banco de dados, então ainda não avançamos tanto. Só que agora eu vejo que ele trouxe eletrônicos duas vezes, eletrodomésticos e móveis. A motivação disso é o que nós vimos, eu tenho dois produtos que estão na categoria de eletrônicos, só que eu não quero trazer eletrônicos duas vezes, para mim só basta uma vez.

[05:57] Vocês vão ver que quando eu trago o resultado do nosso banco de dados, ele está instanciando a mesma categoria duas vezes, então é eletrônicos e eletrônicos. Isso não faz muito sentido, eu só quero instanciar a categoria quando elas forem diferentes. Para resolver isso, como vamos fazer? Eu vou criar uma referência em `Categoria`, e vai ser `Categoria = ultima = null;`. O que eu vou fazer?

[06:26] Eu vou sempre vincular a `ultima = categoria`. Na verdade, a `ultima` sempre vai receber uma categoria. Vocês já vão entender o porquê. Para eu instanciar uma categoria, eu quero que só ocorra quando a última for igual a nulo porque significa que eu ainda não instanciei nenhuma categoria.

[06:49] Se eu estou atribuindo categoria à `ultima`, toda vez, `if(ultima == null)` é porque não teve atribuição, então eu quero que ele instancie a categoria ou `if(ultima == null || !ultima.getNome().equals(andObject))`, ou se o nome da última seja diferente, e é por isso que eu vou negar, que o nome que vem do banco de dados.

[07:17] Então eu pego o nome da `ultima` - lembrando que a `ultima` sempre ter a atribuição da categoria. Se o nome da `ultima` for diferente do que vem do

banco de dados, significa que o que está vindo do banco de dados é outra categoria, então eu vou instanciar essa categoria. Então esse trecho de código fica dentro pode ficar dentro do `if` e acho que a nossa lógica já fica 100% e eu não vou ter mais essa repetição de eletrônicos.

[07:44] Eu não estou instanciando eletrônico duas vezes. Vamos testar? Vou rodar o programa. A nossa lógica deu certo e agora está rodando 100%. Só que agora você deve estar me perguntando: categorias ok, e produto? Produto está aqui, com o nosso Inner join, ele traz produto também, então nós temos que criar o produto.

[08:08] Para isso eu vou instanciar um `Produto produto`, ele vai receber no construtor. Se o Index 1 é o ID da categoria e o Index 2 é o nome da categoria, o 3 vai ser o ID de produto, o 4 vai ser o nome de produto e o 5 vai ser a descrição do produto, então fica `= new Produto(rst.getInt(3), rst.getString(4), rst.getString(5));`.

[08:45] Isso nós não estamos inventando, é simplesmente a ordem do nosso resultado da nossa query. Então é 1, 2, 3, 4 e 5. Coluna descrição é o 5. Quando eu criei esse `Produto produto`, eu preciso vincular ele à categoria. Como eu vou fazer isso? Faz sentido para vocês quando eu vou em Categoria, eu falo que uma categoria tem vários produtos?

[09:10] Faz sentido, tanto é que é verdade, na categoria eletrônicos eu tenho dois produtos, posso ter três produtos. Então ele vai ser uma `private List<Produto> produtos = new ArrayList<Produto>();` na classe `Categoria`. Então agora eu tenho `Categoria` recebendo uma lista de produtos. Aqui já começa a desenhar a ideia.

[09:36] Eu tenho um produto, eu preciso adicionar uma lista de produtos. Como eu vou fazer? Eu chamo um `ultima.adicionar(produto);`. Eu não tenho esse `adicionar`, mas nós criamos, não tem problema. O `adicionar`, em `Categoria`, vai fazer o seguinte, `public void adicionar(Produto produto)`, e

vai pegar a lista de produtos e vai adicionar produto, `produtos.add(produto);` , quantos produtos forem.

[10:03] E agora eu tenho, na minha `CategoriaDAO` , um método `adicionar()` e a única coisa que ele está fazendo é criar uma categoria, ele vai criar só a categoria quando a `ultima` foi igual a nulo, então ele não instanciou nenhuma categoria ainda, ou quando for uma categoria nova, quando for a mesma categoria, eles não vão instanciar, mas o produto, ele sempre vai adicionar naquela categoria.

[10:34] Acho que a lógica ficou bem bacana, temos agora a criação da Categoria e o vínculo aos seus produtos. Vamos testar. Quando eu quero testar esse código, eu já posso descomentar esse `try` porque nós vamos precisar dele novamente. Só que com a diferença que eu não vou precisar mais chamar o método de `buscar(ct)` no `ProdutoDAO` .

[11:02] Porque agora basta eu ter um `for(Produto produto : ct.getProdutos())` , que é uma lista, que nós vamos ver o resultado. Não tenho o `getProdutos()` ainda, não tem problema. Vamos em `Categoria` , escrevo o `public List<Produto> getProdutos()` , e já mando ele gerar `return produtos;` . Agora o `getProdutos()` já está funcionando, já está compilando.

[11:25] Como não tenho mais aquele buscar, passando a categoria, eu não preciso do `catch (SQLException e)` , então eu só preciso mesmo do `for` e é bom que o nosso código já fica até mais bacana, mais sucinto, fica menor. Deixa eu tirar os `imports` que não estão sendo utilizados.

[11:48] Se eu mandar executar esse código, nós vamos ver que o resultado dele é o mesmo que nós tínhamos quando estávamos buscando os produtos com outra chamada, uma outra query. Agora nós temos que o principal objetivo é esse, nós temos a execução de apenas uma única query e vinculando certo, eu tenho eletrônicos, que tem notebook e vídeo game, eletrodomésticos, geladeira e móveis, cômoda.

[12:23] O outros nem aparece, porque eu não tenho nenhum produto vinculado àquela categoria. Então a ideia é exatamente o nosso código, ele ficar mais sucinto, ficar mais performático ao banco de dados e agora não tenho mais o que fazer, o nosso objetivo era esse mesmo. É isso, aluno, espero que vocês tenham gostado e até a próxima aula.