05

Queries N + 1

Transcrição

[00:00] Fala, aluno. Tudo bom? Nas últimas aulas, nós criamos as DAOs e vimos como relacionar as tabelas no nosso banco de dados. Só que agora, o que nós queremos ver é como esse relacionamento das tabelas vai se comportar na nossa aplicação. Nós já fizemos a listagem de categorias, nós vimos que está ok. Só que para mim essa informação é pouca valiosa.

[00:24] Por que eu quero apenas no meu sistema listar as categorias se não é para relacionar à alguma coisa? Até porque, como eu falei anteriormente, essa tabela de categoria é chamada de tabela de domínio, é só porque ela é vinculada à alguma coisa. Nós vinculamos no banco de dados mas não vimos ela funcionando na aplicação.

[00:46] Qual é a ideia aqui? Hoje nós já listamos as categorias. Agora eu quero saber quais produtos que estão nas suas categorias. Então, dada uma categoria, quais produtos que estão abaixo dela? Quais produtos pertencem a ela, digamos assim? Então vamos modificar o nosso código para começarmos a trabalhar com essas informações. Então vamos lá.

[01:13] Eu vou continuar listando o nome da nossa categoria, porque lá na frente vai ficar mais fácil para percebermos esse relacionamento. Então vamos lá. Agora, o que eu quero fazer? Em `TestaListagemDeCategoria.java`, eu quero fazer um for, então quando eu pegar a primeira categoria, eu quero pegar os produtos.

[01:36] Então eu tenho que fazer um `for(Produto produto : "")`, também porque eu posso receber uma lista de produto, que eu quero fazer. E entre as

aspas duplas, eu vou fazer alguma coisa, vou recuperar esse produto. E o que eu quero fazer é o seguinte, quero mostrar, com `System.out.println(ct.getNome() + " - " + produto.getNome());`, o nome da categoria - vamos deixar bonito aqui com um traço e o nome do produto.

[02:14] Então já dá para entender o seguinte: peguei essa categoria. Quais produtos tem nessa categoria? Então se tiver mais de um produto na categoria, eu vou mostrar a categoria e o produto, a mesma categoria e o produto. Se não tem mais produto, ele vai pegar a próxima categoria e fazer a mesma coisa. Isso, aparentemente, para nós, vai funcionar.

[02:41] Mas o que eu vou fazer no `for(Produto produto : "")` para eu poder trabalhar com essa lista de produto, ou então com um único produto, enfim. O que eu posso fazer, nós relacionamos no nosso banco de dados, na nossa tabela de produtos, nós temos a `CATEGORIA_ID`, nós temos uma nova coluna na tabela Produto.

[03:05] Então talvez seria uma possibilidade eu procurar o produto, ou os produtos, pelo ID da categoria. Então como seria a ideia? Talvez na nossa `ProdutoDAO`, eu já passo a Connection e vou fazer o seguinte: `for(Produto produto : new ProdutoDAO(connection)`, tem um método `buscaCategoria()`, que busca por categoria. É uma opção já, já começa a ficar uma coisa bacana.

[03:35] Mas se eu estou no `ProdutoDAO`, então eu posso buscar esse produto e eu já passo a categoria para buscar o produto, então eu não preciso explicitar o buscar por categoria. Na leitura do método, já vamos conseguir entender: ele está recebendo a categoria por parâmetro e está buscando na DAO de produto, então estou buscando um produto por categoria.

[04:10] Então acho que a ideia já ficou mais válida, já ficou mais clara. Vamos criar esse método `public List<Produto> buscar(Categoria ct)`. Eu posso ter um ou mais produtos para a mesma categoria, então se eu tiver mais do que um produto, eu tenho que devolvê-lo em uma lista. Então vamos criar uma li de produtos.

[04:31] Eu vou - eu sei que não é uma boa prática copiar códigos, mas é porque os códigos, eles são bem parecidos mesmo, então eu vou copiar o código do `public List<Produto> listar()` e vocês já vão entender qual vai ser a diferença. Ele vai pedir para eu adicionar o `throws SQLException`, normal. Agora, o que eu vou mudar?

[04:54] Em `String sql = "SELECT ID, NOME, DESCRICAO FROM PRODUTO";`, é o seguinte, invés de ele fazer só um select normal, agora eu vou filtrar a minha consulta, eu vou usar `String sql = "SELECT ID, NOME, DESCRICAO FROM PRODUTO WHERE CATEGORIA_ID = ?";`, que a `CATEGORIA_ID` vai receber um parâmetro, então traga os meus produtos que estão naquela categoria.

[05:18] Para isso, eu preciso *setar* o ID dessa categoria. Ele vai ficar em `try(PreparedStatement pstm = connection.prepareStatement(sql))`. Vou fazer um `pstm.setInt(1, ct.getId())` naquele parâmetro, que se só existe um, eu vou pegar o ID da categoria. Não existe o `ct.getId()` na nossa classe de modelo ainda, deixa eu criar.

[05:41] Vou em `Categoria`, deixa eu criar ele, `public int getId()`, vou fazer um `return id;`. Agora o nosso método `ProdutoDAO` está compilando, não temos nenhum erro. Agora vamos voltar na nossa classe `TestaListagemDeCategoria` e vamos ver. Deixa eu ver o que ele está reclamando. Por mais que peguemos o `throws SQLException`, no main, ele pede para tratamos no momento de buscar por categoria, ele quer que isso seja tratado em um try catch.

[06:23] Nada que impeça, vamos fazer então essa melhoria. Agora o nosso código está compilando, se eu mandar buscar, eu vou verificar aqui os nossos retornos. Olha só, nós já tínhamos listado as nossas categorias, então basicamente o que ele faz, aqui ele listou e guardou em uma lista, a `listaDeCategoria`, agora ele itera sobre essa lista. Então vamos lá.

[06:58] Na categoria Eletrônicos, eu tenho o notebook e o videogame, na Eletrodomésticos eu tenho a geladeira e em Móveis eu tenho a cômoda. Parece que conseguimos vincular os nossos produtos às nossas categorias. Mas estamos utilizando duas queries, digamos assim, dois serviços de busca na mesma chamada, em `TestaListagemComCategoria`, na linha `List<Categoria> listaDeCategoria = categoriaDAO.listar();`.

[07:35] Vamos ver qual é o impacto disso, vamos colocar em `CategoriaDAO`, em `public List<Categoria> listar() throws SQLException`, um `System.out.println("Executando a query de listar categoria");`. Coloquei esse comando, e nessa query, nesse serviço de `ProdutoDAO`, o `public List<produto> buscar(Categoria ct)`, eu vou fazer um `System.out.println("Executando a query de buscar produto por categoria");`.

[08:20] Eu falei serviço, mas são as nossas DAOs, só para ficar claro. Então agora que botamos esses `System.out.println()` naqueles dois métodos das nossas DAOs, vamos executar de novo o código e vamos ver uma situação. Veja bem, foi executada a query de listar categorias, essa query foi uma vez e guardou em uma lista. E eu vou iterar a lista de categorias justamente para eu poder vincular os meus produtos àquela categoria.

[09:02] Então quando ele itera na lista de categoria e pega Eletrônicos, então ele vai e busca os produtos por categoria Eletrônicos. Ele trouxe o notebook e o video game. Iterou na lista de categoria, ele vai mais uma vez no banco, porque a nossa categoria mudou, é Eletrodomésticos, eu tenho um novo produto. E a mesma coisa para Móveis.

[09:32] Então em uma ação relativamente simples, que eu só quero relacionar duas tabelas, então os Produtos com as Categorias, nós vamos no banco de dados 1, 2, 3, 4 vezes. Isso porque temos poucos produtos e poucas categorias. Pense em N produtos e N categoria então pense na confusão que ia ficar e o quão pouco performático é em relação à nossa aplicação, em relação à

comunicação com o banco de dados. Então já vimos que esse tipo de listagem não é uma boa prática.

[10:20] Inclusive isso tem um nome, que chama queries N+1, porque nós temos uma query base, que é a `categoriaDAO.listar()`, que ele vai uma vez e busca todo mundo. Só que através dessa base, eu executo N queries para trazer as informações que eu preciso que no caso é de produto. Então, na primeira categoria, eu trago N produtos. Eu vou ao banco mais uma vez e trago N produtos, vou no banco mais uma vez e trago mais N produtos.

[10:53] Isso referenciando a categoria. Então isso não é uma boa prática e tem que melhorar, porque, se não, tende a crescer e isso vai gerar um prejuízo futuramente. Como já estamos trabalhando a um certo tempo nessa aula, não vamos entrar no Refactory nesse momento, só quero que vocês fiquem na cabeça como podemos melhorar o nosso código, o que podemos fazer para melhorar e já analisar realmente essa questão da performance.

[11:35] Talvez vocês possam brincar um pouco, adicionar mais categorias, adicionar mais produtos e ver quantas vezes vamos ter que ir no nosso banco de dados para trazer essas informações. Então, como falado, no próximo vídeo nós vamos resolver essa situação e por hoje nós vamos ficando por aqui. Então, aluno, espero que vocês tenham gostado e até a próxima aula.