04

Consultas com select new

Transcrição

[00:00] Olá, pessoal. No vídeo anterior, paramos nessa consulta de um relatório, que tem que trazer várias colunas distintas, conforme aquele relatório de vendas que tínhamos proposto.

[00:11] O problema é que o código ficou dessa maneira, o retorno aqui é um *list* de *array* de *object* que, embora funcione, é meio ruim, porque fica esquisito, nós não sabemos direito o que tem dentro deste *array*, já que é *object*, pode ser qualquer coisa, enfim. No código para imprimir aqui, acabou ficando meio estranho também.

[00:32] A ideia agora é melhorarmos esse código da consulta. Antes disso, só mostrando aqui, eu fiz uma pequena mudança naquela nossa classe "CadastroDePedido", só para termos mais registros.

[00:44] Naquele método "popularBancoDeDados", eu criei mais duas categorias, só tinha a categoria de celulares. Criei uma categoria chamada videogames, uma categoria chamada informatica e criei três produtos, tem o celular, tem o videogame e tem o macbook. Salvei tudo isso no banco de dados.

[01:01] Aqui, no código de criar o pedido, eu criei dois pedidos. O primeiro pedido tem dois itens, o primeiro item é o celular, o segundo item é o videogame. O outro pedido, só para simular que são vários pedidos, com vários itens, esse segundo pedido tem apenas um item, o terceiro produto, um notebook.

[01:17] Só para termos mais informações no código. Se rodarmos esse *main*, continua da mesma maneira, mas agora ele está trazendo vários produtos.

[01:28] Tem o Playstation 5, foram vendidos 40, 9 de fevereiro última data de venda, o celular, foram vendidos 10 e o notebook, foram vendidos 2. Só para ter mais registros, para ficar mais próximo da realidade. Voltando para o nosso código de consulta, para o nosso problema, como fazemos para resolver isso? Tirar esse `<Object[]>`?

[01:48] A ideia é tentarmos deixar esse código mais próximo da orientação a objetos, porque aqui está muito genérico, está muito vago, um *array* de *object*. Então, o que podemos fazer? Podemos transformar esse *array* de *object* em uma classe, uma classe que represente esse relatório. Na classe, eu terei os atributos: nome do produto, quantidade vendida e última venda.

[02:09] Então, ao invés de trabalhar com o *array* de *object*, eu vou trabalhar com uma classe. Aqui, o meu retorno não será o `public List<Object[]>`, será o nome da minha classe. Por exemplo, poderia ser `List<RelatorioDeVendasVo>`. Geralmente, nessa classe, o pessoal usa esse padrão *value object*.

[02:30] É uma classe que só tem atributos, *getters* e *setters*, não tem nenhum comportamento, só tem o estado, só tem os atributos, então é comum o pessoal usar esse padrão Vo. Essa será a classe que eu vou retornar, não será mais um *array* de *object*. Porém, vai dar erro de compilação, porque não existe essa classe. Vamos criar essa classe.

[02:48] "Ctrl + 1", "Create Class". Só vou trocar o pacote, vou jogar dentro do pacote "modelo". Na verdade, vou criar um pacote chamado "vo", que ficam os VOs aqui dentro. Qual é a ideia? Dentro dessa classe, eu terei apenas essas três informações da tabela de relatório de vendas.

[03:04] São esses três atributos. O primeiro é o nome do produto, que é uma *string*, `private String nomeProduto;`, só para ficar mais claro, porque aqui eu não estou

dentro da classe "Produto", então nome do que? Vou colocar `nomeProduto`. `private` - vai ter um, eu acho que precisa ser um `Long`, se não me engano, não pode ser um *int*, porque estamos fazendo um *sum*, e ele devolve um *long*, `private Long quantidadeVendida`.

[03:34] Por fim, `private LocalDate dataUltimaVenda`. São essas as três informações que eu preciso no meu VO. Esse meu VO, essa minha classe, representa o meu relatório. Cada coluna do relatório vira um atributo nessa classe. A ideia é que eu vou ter um construtor, que recebe essas três informações. Vou gerar o construtor.

[03:59] E, no caso, vou gerar os *getters* e *setters*. Nesse caso, só vou colocar os *getters* mesmo, porque o construtor já está inicializando todos os atributos. "Source > Generate Getters and Setters...", mas vou marcar apenas os *getters*. Está aqui, essa é a classe que representa o relatório. É uma classe, não é uma entidade, por isso a ideia do *value object*, é só uma classe de valor, um objeto de valor.

[04:27] Ele só tem atributos, *getters* e o construtor para receber esses atributos, não tem nenhuma anotação. Porém, se não é uma entidade, eu não consigo fazer um `SELECT RelatorioDeVendas` aqui, não dá para fazer isso dessa maneira `SELECT RelatorioDeVendasVo r`, porque não é uma entidade. No *select* só pode ter uma entidade.

[04:50] Porém, a JPA, ela tem um recurso. Quando você quer fazer um relatório, você quer fazer uma consulta, um *select* de um relatório e não quer devolver um *array* de *object*, tem um recurso, o *select new*. Então você pode fazer `String jpql = "SELECT new "`. É como se eu estivesse dando *new* em uma classe. Qual classe? `RelatorioDeVendasVo`.

[05:07] Continuando, `"SELECT new RelatorioDeVendasVo ()"`. É como se fosse um *new*, passa as informações, separadas por vírgula, como já estão no código, que serão os parâmetros. Então tem uma vírgula, tem outra vírgula. Tem uma vírgula, esse primeiro o campo, e eu acabei apagando o nome do produto, então vou jogar mais uma linha.

[05:33] Na primeira linha do *select* estava o nome, então vou voltar ela aqui. `"produto.nome, sum(item.quantidade), max(pedido.data))"` - apaguei os parênteses aqui, já coloquei. Nesse não tem vírgula depois, só que eu preciso fechar os parênteses do `Select new`, porque esse parêntese está fechando o `MAX`, então preciso de mais um parênteses para fechar esse `SELECT new`. E aí vai. `FROM Pedido pedido JOIN`, etc.

[06:00] Tudo igual, não muda nada. A JPA, ela tem esse suporte, é o *select new*. Quando tem um *select new*, ela sabe que é para criar uma instância da classe passada, que não é uma entidade, e ela vai passar os valores para o construtor. Ela dá um *new* passando essas informações para um construtor. Para cada registro, ela vai criar um objeto e jogar em uma lista. Bem mais fácil de ler, bem mais orientado a objetos.

[06:26] Agora vai dar erro de compilação no `List<Object[]> relatorio = pedidoDao.relatorioDeVendas();`, porque, no caso, esse relatório não traz mais um `List<Object[]>`. É um `List<RelatorioDeVendasVo>`. Agora o *for each* vai mudar, `relatorio.forEach()`, vou fazer um *for each* tradicional aqui.

[06:47] Para facilitar, o que eu posso fazer? `relatorio.forEach(System.out::println);`. Vou usar aqui essa expressão mais curta. Ele vai percorrer cada um dos elementos e dar um *system out* neles. Porém, a nossa classe "RelatorioDeVendasVo" não tem um `toString`, então ele vai imprimir aquela impressão padrão de objeto.

[07:13] Vou gerar um método `toString` na classe "RelatorioDeVendasVo", só para facilitar. Vou clicar com o botão direito, "Source > Generate toString". Vou marcar os três atributos. Vai gerar os três atributos. Agora eu consigo fazer o *system out* no objeto, que ele vai chamar o `toString`. A princípio é isso, vamos rodar esse *main*, vamos ver no console o que vai rolar.

[07:43] Deu um problema na minha *query*, vamos dar uma olhada. Ele disse que não conseguiu localizar a classe "RelatorioDeVendas". Porque eu esqueci de um

detalhe importante. No *select new* - cadê a classe "PedidoDao"? Tem que passar o caminho completo da classe, então tem que passar o pacote dela.

[07:59] `package br.com.alura.loja.vo;`. Vou dar um "Ctrl + C", vou colar aqui `SELECT new br.com.alura.loja.vo.RelatorioDeVendasVo`. Tem que ser o nome completo da classe, incluindo o pacote. Vamos rodar novamente o *main*. Vamos dar uma olhada.

[08:21] Ele trouxe. É o mesmo *select*, então perceba: "select produto2_.nome", "sum(itens1_.quantidade)", "max(pedido0_.data)", "from pedido", fez os *joins*, "group by" nome, "order by", igual. Só que agora ele trouxe um objeto invés de trazer um *array* de *object*, trouxe objetos do tipo relatório de vendas.

[08:42] E trouxe certo, PS5, foram vendidos 40, o celular, foram vendidos 10, o Macbook, foram vendidos 2. E a data, como estou pegando a data atual, vão vir todos com a data atual, de hoje, que é 9 de fevereiro de 2021, que eu estou gravando esse vídeo. Perceba: funcionou da mesma maneira, não mudou nada, é o resultado, só mudou o código.

[09:03] Isso é tipo um *refactory*, uma refatoração que nós fizemos uma melhoria no código. Ao invés de trabalhar com um *array* de *object*, criamos uma classe que representa esse *array* de *object*, para deixar o código um pouco mais simples. Mais um recurso da JPA, no caso do JPQL, da consulta, que é o *select new*.

[09:20] Isso é muito utilizado quando você precisa fazer relatórios, como esse relatório de vendas. Você tem que trazer informações fazendo *group by*, usando funções, e trazer várias colunas de várias tabelas misturadas. *Select new* é a solução para essa situação. Espero que vocês tenham gostado. No próximo vídeo vamos discutir mais um recurso de consultas. Vejo vocês lá.