04

Lidando com commit e rollback

Transcrição

[00:00] Fala, aluno. Tudo bom? Voltando ao nosso curso de JDBC, agora eu quero voltar na nossa classe `TestaInsercaoComParametro`, onde nós vimos que estava tendo um erro na hora de inserir os produtos. Nós vimos que foi *setado* o `setAutoCommit(false);`, que é exatamente para nós termos o controle da nossa transação, agora eu que quero controlar a transação do início ao fim na hora de adicionar ao banco de dados.

[00:30] Porém não estava sendo inserido. Qual é a ideia da nossa transação? Para mim, ela só vai valer se eu adicionar esses dois produtos ou nenhum deles. Por que eu bato nessa tecla, de adicionar os dois ou nenhum? Vamos pensar em uma conta, uma conta bancária mesmo, onde eu tenho uma conta de origem e tenho uma conta de destino. Vamos supor que na conta de origem nós vamos transferir um dinheiro para a conta de destino, e essa conta destino vai receber.

[01:06] Então, ou seja, eu transfiro o dinheiro, o dinheiro sai da minha conta de origem e chega na conta de destino, então a conta de destino recebe. Se no momento em que eu estiver fazendo essa operação, quando estiver chegando na conta de destino, vamos supor que foi encontrado um erro e a transação, a nossa operação, ela não foi concluída. Ou seja, o dinheiro não chegou na conta de destino.

[01:37] Essa transação, na verdade, ela vai ter que ser desfeita, o dinheiro vai ter que voltar para a conta de origem e vou ter que receber, no meu canal de autoatendimento, uma informação: seguinte, a sua transação não foi realizad

com sucesso, por favor tente novamente mais tarde. Algo nesse nível, porque para mim não faz sentido uma transação sair do meu banco, sair da minha conta, sair um dinheiro da minha conta e não chegar na conta de destino.

[02:12] Mas, imagina, se tiver já saído da minha conta, uma transação para sair e uma outra transação para chegar. Então não fazia sentido, porque saiu da minha conta com sucesso, mas não chegou na outra. O dinheiro ficou onde? Então por isso que eu fico batendo nessa tecla de que as nossas transações, elas têm que ser daquela maneira.

[02:34] Os dois produtos, eles têm que ser inseridos juntos, pelo mesmo motivo de uma conta bancária, que tem que ser na mesma transação, porque se eu tiver qualquer erro em uma das pontas, a transação, ela tem que ser desfeita. Os bancos de dados, hoje em dia, eles já estão trabalhando, eles já estão preparados para essas transações. Eu tenho um encadeamento de comandos SQL, de operações SQL, que se alguma delas der errado, todas as outras desfazem.

[03:00] Eu tenho serviços, hoje em dia, em sistemas, que ou você faz todo aquele procedimento ou não faz nenhum, volta todo mundo. Então por isso que eu venho batendo nessa tecla de nós termos esses dois produtos sendo inseridos juntos, é esse o conceito de transação. Entendemos o conceito, como que agora funciona para resolvermos o nosso problema?

[03:27] Quando estamos trabalhando com controle transacional manual, que é quando botamos o `.setAutoCommit(false);`, o Commit nós temos que explicitar no nosso código. Então em `TestaInsercaoComParametro`, dentro de `Connection`, eu vou ter um método chamado `commit();`.

[03:46] Agora esse `connection.commit();` que vai fazer o seguinte: quando todos os Statements `adicionarVariavel` estiverem preparados, ou seja, quando eu já tiver falado para a minha transação que nós vamos ter um produto que vai ter o seu método `execute`, que vai me retornar as `getGeneratedKeys`, out produto que vai chamar o `execute`, que vai me retornar as `getGeneratedKeys`,

quando tudo estiver ok e não tiver me dado problema, então dá o Commit na minha transação.

[04:15] Agora, como nós já comentamos aqui o `if(nome.equals("Radio"))`, se eu executar o `TestaVariavelComParametro`, ele tem que dar certo. Vamos verificar o resultado. Foram criados dois IDs, o 103 e o 104. Para testarmos se os produtos foram de fato criados, vamos mandar executar a `TestaListagem`. Estamos aqui, nós temos agora o produto com IDs 103, que é uma "SmartTV" e o produto com IDs 104, que é um "Radio de Bateria".

[04:47] Então agora a nossa classe `TestaInsercaoComParametro` está funcionando perfeitamente. Só que veja bem, eu tenho como melhorar esse meu código. Eu posso, na verdade, quando eu realizar o Commit na minha transação, se tiver algum erro, eu posso dar um rollback, ou seja, eu posso explicitar o rollback falando o seguinte: desfaz todo mundo e me dá uma mensagem.

[05:11] Para termos esse controle mais fino da transação, eu vou abrir um `try`, em `TestaInsercaoComParametro`, e vou esse código do `PreparedStatement`, `adicionarVariavel`, `connection.commit`, `stm.close` e `connection.close`, que nós tínhamos do lado de fora para dentro do `try {}`.

[05:25] Agora eu vou pegar uma `catch (Exception)` e vou falar o seguinte: me fala qual foi a exceção. Eu quero que dê uma mensagem, falando que o `System.out.println("ROLLBACK EXECUTADO");` e eu quero fazer de fato o `connection.rollback();`, que é isso que consiste o nosso `.setAutoCommit(false);`, é nós controlarmos a nossa transação, em que momento ela vai dar o rollback, em que momento ela vai dar o Commit.

[05:59] Então é esse tipo de estratégia que vai servir para termos esse controle da nossa transação, quando *setamos* o `setAutoCommit(false);`. Para verificarmos como ficou o nosso código, eu vou tirar o comentário do nosso

`if` porque agora eu quero que dê um erro no momento em que estivermos inserindo, preparando os nossos Statements.

[06:26] Só para ficar mais claro, eu vou apagar esses dois produtos que acabamos de inserir. Duas linhas foram modificadas, duas linhas foram apagadas. Para termos uma confirmação, vamos executar `TestaListagem` de novo, o nosso código. Agora só os dois produtos iniciais. Agora vamos executar o nosso código `TestaInsercaoComParametro`, com o controle mais fino da nossa transação.

[06:56] Executou. Olha lá, ele fala que o ID foi criado, o 105, mas nós já tínhamos visto isso anteriormente. Ele falou qual foi a motivação, qual foi o *print stack trace* da minha sessão e ainda me falou "ROLLBACK EXECUTADO". Com isso eu estou garantido agora que ele vai entrar nesse momento do `catch (Exception)`, e vai fazer o `connection.rollback();`, porque ele executou o `e.printStackTrace();` e executou a nossa mensagem.

[07:35] Então com isso, agora estou garantindo que toda a minha transação, ela está daquela forma que nós queríamos: ou vai adicionar todo mundo ou não vai adicionar ninguém, porque nós vimos, o primeiro produto, ele não passou no `if`, ele não geraria uma exceção. Anteriormente, como estava com o `setAutoCommit` igual a true, ele estava Commitando a cada transação, agora não.

[08:02] Agora eu tenho a minha transação realizando o Commit só depois que der tudo certo, ou então ele vai sair abruptamente no `if`, vai cair no `catch`, que vai capturar essa exceção, vai me mostrar qual foi a motivação da exceção, me mostra a mensagem do `println` e de fato deu o rollback. Agora não tenho nenhum problema com a minha transação.

[08:23] Então, com isso, nós estamos vendo como funciona esse controle transacional já na vida real. Esses aqui vão ser códigos, no dia a dia, no trabalho, que serão feitos dessa forma. Então agora nós já temos um código

com um controle mais fino, um código mais complexo, mas mais bonito também.