SQL Transactions - Aula 08

Gabriel de Paula Gaspar Pinto

Exercício 1

Transação, em um contexto de banco de dados, é um conjunto de comandos SQL individuais que são agrupados e executados juntos de maneira sequencial. Logo, ou tudo acontece ou nada acontece. Por exemplo, caso haja um erro no banco de dados, a luz caia ou tenha um problema deste tipo, no meio de uma transação, nenhum comando será guardado e o banco voltará estar do jeito que estava antes desta transação. Mas, caso não aconteça nenhum problema durante a transação, ela é concluída e suas alterações são efetivadas no banco.

Um exemplo de transações da vida real, é um banco (do setor financeiro). Quando se vai sacar dinheiro, a parte do banco de dados faz um SELECT para mostrar a quantidade de dinheiro na conta e um UPDATE para subtrair o valor sacado e logo após o dinheiro sai do caixa eletrônico. Caso haja um problema entre o processo de atualizar o valor da conta e o saque do dinheiro, se não fosse usado transações, o cliente simplesmente perderia o dinheiro, mas como as transações são usadas, o UPDATE é revertido para o valor anterior, como se nada tivesse acontecido.

Exercício 2

a) Em um sistema bancário, ao tentar transferir R$100,00 de uma conta A para B. O dinheiro pode ser removido da conta A mas não ser creditado na conta B, resultando no dinheiro sumindo.

b) Em uma conta poupança, o saldo nunca pode ser negativo, mas sem consistência, é possível que o saldo seja atualizado para um número negativo, resultando em problemas.

c) Em um sistema sem isolamento, de um show, por exemplo, quando restar somente 1 ingresso, duas pessoas, simultaneamente, podem ver que há somente 1 ingresso restante. As duas conseguem comprar, resultando em um estoque negativo.

d) Um pedido pode ser finalizado e confirmado ao cliente, mas antes das mudanças serem gravadas, ocorre uma queda de energia, resultando em um pedido perdido, mesmo após a confirmação.

Exercício 3

a) Para inserir o usuário, um simples INSERT INTO foi suficiente, mas para inserir os conteúdos da conta corrente e da conta poupança, foi necessário usar LAST_INSERT_ID(), que retorna a última chave primária inserida, já que a chave primária do usuário é auto incremental, o que torna a chave primária desconhecida, já que não é algo imutável, como o CPF de uma pessoa. Em relação à transações, tudo ocorreu de acordo com o esperado, sem nenhum erro.

b) Como esperado, o rollback fez com que o banco de dados retornasse ao estado anterior ao início da transação.

   Isso também aconteceria em situações como ao inserir dados em várias tabelas e um dos inserts falhar, quando uma condição de integridade for violada durante uma atualização em lote ou quando há bloqueios ou deadlocks durante a transação são exemplos que podem ser dados.

   O rollback é útil para proteger o banco de dados em caso de falha humana. Por exemplo, se alguma pessoa remover dados sensíveis, sem querer, ainda é possível de usar o rollback para que o sistema consiga recuperar os dados apagados por erro. Outro caso em que o rollback é útil, é em falhas não humanas, como por exemplo, quedas de energia. Se o banco de dados ficar sem energia no meio de uma transação, a transação não é concluída e o banco retorna ao estado anterior ao início da transação.

c) Após iniciar a transação na primeira conexão e subtrair R$500,00 da conta poupança, mas sem rodar o commit, a segunda conexão só conseguia rodar selects na tabela, se rodasse um update, a segunda aba perdia conexão com o banco de dados. Mas, após dar o commit na primeira transação, foi possível remover mais R$400,00 da conta, mesmo que ela só tenha R$300,00, restando em um saldo de R$100,00 negativos em uma conta poupança. Rodando o update da segunda conexão e, antes de 30 segundos dar commit na primeira transação, o erro de desconexão é não acontece. Isso acontece porque o padrão de nível de isolamento do MySQL é "repeatable read". Para previnir o erro de ter uma conta poupança negativa, é possível atualizar a query de update, para algo como mostrado abaixo ou simplesmente mudar o nível de isolamento do banco para "serializable", que não permite conexões concorrentes.

```
        UPDATE conta_poupanca SET saldo = saldo - 400 WHERE id_conta = 1 AND saldo >= 400;

```

d) Após iniciar a transação na primeira conexão, não é possível atualizar nenhum dado na segunda conexão até a transação da primeira conexão ser finalizada. Após 30 segundos, caso a transação da primeira conexão não tenha sido finalizada, a segunda conexão perde a conexão com o banco de dados.

Exercício 4

a)

```
        SET @num1 = 20;
        SET @num2 = 30;
        SET @num3 = @num1 + @num2;
        SELECT @num3;

```

b) Com a seguinte query, é possível fazer o pedido:

```
        SET @a = 5;
        SET @b = 2;
        SET @aux = 0;

        SET @aux = @a;
        SET @a = @b;
        SET @b = @aux;

        SELECT @a AS a, @b AS b;

```

Exercício 5

a)

```
        UPDATE conta_corrente SET saldo = 400 WHERE id_conta = 1;
        UPDATE conta_poupanca SET saldo = 300 WHERE id_conta = 1;

```

b)

```
        SET @transferencia = 300;

```

c)

```
        UPDATE conta_corrente SET saldo = saldo - @transferencia WHERE id_conta = 1;
        UPDATE conta_poupanca SET saldo = saldo + @transferencia WHERE id_conta = 1;

```

d) As contas estavam com os valores exatos.

Questões de concurso

1. a) Certo
2. a) As transações controlam melhor apenas a concorrência.
3. a) Certo
4. A questão não possui enunciado completo, mas considerando o que foi dado, está incorreto.
5. c) uma transação lê dados gravados por outra transação que ainda não foi confirmada (committed).

6. c) A atomicidade é a propriedade que assegura que as atualizações relacionadas e dependentes ocorram dentro dos limites da transação ou nenhuma atualização será efetivada no banco de dados.
7. b) Errado
8. d) II e III.
9. c) Propriedade de durabilidade, que garante que, após uma transação ser concluída com êxito, as alterações feitas no banco de dados persistem, mesmo se houver falhas do sistema.
10. c) Isolação e Atomicidade.
11. e) Isolamento
12. b) mesmo na ocorrência de falhas no sistema de banco de dados, após o término da transação.
13. d) Atomicidade, Consistência, Durabilidade e Isolamento.
14. c) durabilidade.
15. d) O isolamento resolve os efeitos decorrentes da execução de transações concorrentes, em que cada transação é executada de forma que as operações parciais das demais transações não afetem a transação atual.
16. e) transações.
17. a) Certo
18. b) Errado
19. b) Errado