

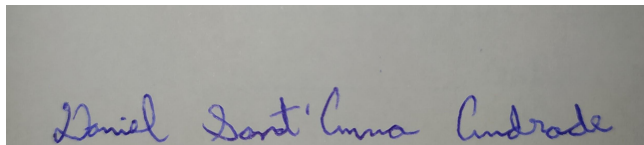
2ª. Prova BD II 2021-2 Prof. Sergio Serra - SI - UFRRJ

Nome: Daniel Sant' Anna Andrade

Nota: _____

Matrícula: 20200036904

Assinatura:



Prova somente via SIGAA. Assine esse documento. Prova Com consulta e individual. Provas corrigidas por comparação não copie do colega

1ª Questão – (4 PONTOS) O QUE SÃO TRANSAÇÕES EM BANCOS DE DADOS RELACIONAIS? QUAIS SUAS PRINCIPAIS PROPRIEDADES. Quais os tipos de problemas podem ocorrer se não ocorrer uma gerência eficiente de transações? Explique detalhadamente cada uma delas, descreva tecnicamente um exemplo

R= Transações são um conjunto de operações que compõem uma tarefa ou unidade lógica de trabalho que será executada, com elas podendo realizar as instruções CRUD.

O SGBD precisa garantir que uma transação seja executada corretamente, ou seja, ser executada por completo sem que ocorra falhas. No caso da ocorrência de falhas a transação precisa ser abortada.

Outra parte importante no assunto de transações, são as propriedades ACIDs, elas garantem a integridade dos dados, permitindo a execução correta das transações. Essas propriedades são:

- Atomicidade: garante que cada transação seja tratada como uma entidade única, onde ou ela deve ser executada por completo (realizando commit no final) ou falhar completamente (realizando rollback após a falha).
- Consistência: assegura que uma transação somente leve o banco de dados de um estado válido a outro, os dados que são gravados devem sempre ser válidos, de acordo com regras definidas.
- Isolamento: é comum que transações sejam executadas concorrentemente por vários usuários ao mesmo tempo. O isolamento permite deixar o banco de dados no mesmo estado em que ele estaria caso as transações fossem executadas em sequência.
- Durabilidade: garante que uma transação, quando executada, permanecerá assim mesmo que haja um problema no sistema. As transações já finalizadas são gravadas em dispositivos de memória permanente, para que os dados estejam sempre disponíveis, mesmo que a instância do banco de dados seja reiniciada.

Uma má gerência dos bancos de dados podem causar uma falha de sistema, um erro na execução das transações, uma condição de exceção (aviso de que foi detectado um problema), uma imposição de controle de concorrência, uma falha de disco, entre outros.

Esses problemas podem acarretar em:

- O problema da atualização perdida, onde uma transação não é executada corretamente;

T1	T2
Ler itens no estoque (Total = 12)	
Vender 3 Items	
	Ler itens no estoque (Total = 12)
	Vender 2 Items
Atualizar itens no estoque para 9	
	Atualizar itens no estoque para 10

- O problema de dirty read, a transação lê dados escritos por uma transação simultânea não efetivada (não foi realizado o commit);

T1	T2
read(x)	
X=X-5	
write(x)	
	read(x)
	x=x+5
	write(x)
ROLLBACK	
	commit

- O problema do resumo incorreto (a consulta retorna dados para desatualizados);

T1	T2
	sum = 0
	read(A)
	sum = sum + A

read(X)	
$X = X - N$	
write(X)	
	read(X)
	sum = sum + X
	read(Y)
	sum = sum + Y
read(Y)	
$Y = Y + N$	
write(Y)	

- O problema de unrepeatable read, onde a transação lê novamente os dados lidos e percebe que eles foram alterados por outra transação (que os efetivou após ter sido feita a leitura anterior).

T1	T2
read(X)	
	read(X)
write(X)	
	read(X)

- O problema de phantom read, onde a transação executa uma segunda vez uma consulta para retornar um conjunto de linhas que satisfazem uma condição mas, descobre que o conjunto de linhas que satisfazem a condição é diferente por causa de uma outra transação efetuada recentemente.

T1	T2
read(X)	
	read(X)
delete(X)	
	read(X)

2ª. Questão - (4 PONTOS) O que são deadlocks e impasses? Quais as diferenças? Descreva, em detalhes, o funcionamento dos algoritmos de prevenção e

eliminação de deadlocks. Explique e compare as diferenças de como eles são materializados nos SGBDs MySQL e Postgresql.

R= Deadlocks são impasses, eles impossibilitam a execução de transações as mantendo em um estado na qual nenhuma das transações nesse ciclo conseguem terminar de serem executadas. Para se entender deadlocks, é necessário também saber de locks e blocks.

Os locks ocorrem quando uma transação acessa uma parte dos dados onde outra transação também esteja precisando desses dados ao mesmo tempo. Ao realizar o lock dos dados, é garantido que, no momento que a query entrar em execução, nenhuma informação sua será alterada por outra transação. Eles são normais e é um mecanismo usado pelos SGBDs para protegerem a integridade dos dados durante as transações.

Os blocks (bloqueio) ocorrem quando duas transações precisam acessar de forma simultaneamente o mesmo fragmento de dados. Assim, uma transação dá um lock dos dados que estão sendo acessados, e a próxima transação que os quiser utilizar fica bloqueada (blocked) esperando que o outro conclua e termine o lock. Quando a primeira transação for concluída, a transação que a bloqueia é desbloqueada seguindo sua operação. A cadeia de blocks é parecida com o funcionamento de uma fila. Uma situação de block NÃO pode ser resolvida por si só, ou seja, ela depende de quem realizou o lock retirá-lo. Se o processo de lock não concluiu a transação corretamente, ela mantém todas as outras transações em blocks, às vezes podendo levar muito tempo para tudo ser concluído.

O deadlock ocorre quando uma transação entra em block e espera por uma segunda transação para concluir seu trabalho e liberar os locks. Enquanto a segunda transação ao mesmo tempo entra em block e espera que a primeira libere seu lock, ou seja, existe uma dependência cruzada entre elas, que nunca será resolvida.

Em um deadlock, as transações já estão bloqueando uma a outra, portanto, é necessário que haja uma intervenção externa para resolver o impasse. Por isso, os SGBDs possuem mecanismos de detecção e resolução de conflitos, em que uma transação precisa ser escolhida e eliminada para que a outra possa continuar funcionando. A transação escolhida como vítima recebe uma mensagem de erro muito específica indicando que ele foi escolhido como uma vítima de deadlock, podendo a transação ser reiniciada por meio do código com base nessa mensagem de erro. Os deadlocks são considerados uma situação crítica no mundo do banco de dados, pois as transações são terminadas automaticamente sem sua conclusão.

Existem dois métodos para tratamento de deadlock:

Prevenção: garante que o sistema nunca entrará em deadlock. É o mais utilizado se a probabilidade do sistema entrar em deadlock for alta.

Detecção e recuperação: permite que o sistema entre em deadlock para então removê-lo e recuperá-lo.

Métodos de prevenção:

1. Obriga que cada transação bloqueie todos os itens de dados antes de sua execução. A transação só poderá funcionar caso todos os itens estejam bloqueados.
2. Quando uma transação T2 solicita o bloqueio que está sendo mantido pela transação T1, o bloqueio concedido a T1 pode ser retirado utilizando rollback de T1, concedendo a T2. Essa execução pode ser feita de duas formas:
 - 2.1. Esperar-morrer: caso a transação Ti solicite um item de Tj podem ser realizada uma de duas coisas: se Ti é mais velha que Tj, Ti espera, se Ti é mais nova do que Tj, ti é desfeita.
 - 2.2. Ferir-esperar: caso a transação Ti solicite um item de Tj podem ser realizada uma de duas coisas: se Ti é mais nova do que Tj, Ti espera, se Ti é mais velha do que Tj, Tj é desfeita.

Deadlocks no PostgreSQL:

A utilização de bloqueios explícitos pode causar impasses (deadlocks), especialmente quando duas (ou mais) transações mantiverem bloqueios desejados pelas demais. O PostgreSQL detecta automaticamente as situações de impasse e, para solucioná-las, aborta uma das transações envolvidas e permite que a(s) outra(s) prossiga(m).

Um deadlock ocorre quando dois ou mais processos possuem bloqueios em objetos separados, e cada um dos processos está tentando obter um bloqueio no objeto que o outro processo bloqueou. Assim, o SQL Server, resolve abortando um dos processos permitindo que o outro processo continue a sua transação no banco de dados. O processo abortado envia uma mensagem para o arquivo de error log do SQL Server informando sobre ela. Deadlocks podem causar uma pressão sobre os recursos do servidor SQL Server, principalmente CPU.

Dicas para evitar deadlocks:

- Verifique se o banco de dados está desenhado corretamente.
- Não permita que usuários interfiram durante as transações.
- Procure ter transações no SQL o mais curto possível.
- Reduza a quantidade de vezes que sua aplicação conversa com o SQL Server usando procedures para manter as transações dentro de um único lote.
- Reduza a quantidade de leituras, se você precisa fazer leituras constantes da mesma informação, coloque essa informação em uma variável ou matriz para você consultar por lá.
- Diminua o máximo possível o tempo de bloqueio. Desenvolva aplicativos que fazem bloqueios em último caso e libere-os o mais rápido possível.

Deadlocks no MySQL:

O MySQL trabalha com o autocommit ativado, por conta disso, os bloqueios neste SGBD são utilizados para serializar os acessos simultâneos aos dados, impedindo que as operações de leitura e escrita ocorram ao mesmo tempo numa tabela.

Para lidar com os deadlocks o MySQL possui dois métodos:

- Esperar o tempo limite de execução de uma transação.
- Realizar a detecção de deadlock, realizando o rollback das transações para liberar as outras transações.

Dicas para evitar deadlocks:

- Tornar as transações pequenas e de curta duração;
- Melhorar a qualidade de busca;
- Adicionar e utilizar índices;
- Evitar ao máximo possível o uso de bloqueios;
- Fazer alterações nas tabelas, removendo a restrição de chaves estrangeiras, evitando linkar tabelas.
- Criar uma tabela auxiliar para serializar as transações antes de acessar outras tabelas.

Até 2 pontos serão inseridos na AV2 apenas para aqueles que apresentaram o trabalho prático em sala de aula.