
	<p align="center"><b>MINISTÉRIO DA EDUCAÇÃO</b></p> <p align="center"><b>UNIVERSIDADE FEDERAL DO PIAUÍ</b></p> <p align="center"><b>CAMPUS SENADOR HELVÍDIO NUNES DE BARROS</b></p> <p align="center">Curso: Sistemas de Informação</p> <p align="center">Disciplina: Banco de Dados II</p> <p align="center">Aluno: Hector José Rodrigues Salgueiros</p>	
---	---	---

### Atividade

- 1. O que significa a execução concorrente de transações de banco de dados em um sistema multiusuário? Discuta porque o controle de concorrência é necessário e dê exemplos informais**

Significa que várias transações ocorrem ao mesmo tempo, por diferentes usuários. O controle de concorrência é necessário para que não haja conflitos de leitura e gravação entre as transações.

- 2. Discuta os diferentes tipos de falhas. O que significa uma falha catastrófica?**

Falha por hardware, quando os componentes da máquina são a razão do problema; Falha por software, quando o programa utilizado para funcionar corretamente; Falha por rede, quando a conexão com a internet é perdida; Falha de energia, quando ocorre alguma oscilação ou queda da energia elétrica; Falha humana, quando o desenvolvedor o responsável pela atividade comete algum erro acidentalmente. Falhas catastróficas são falhas que não podem ser prevenidas ou imaginadas que geram graves danos ou irreparáveis ao banco de dados, como terremotos, incêndios, tsunamis, entre outros casos naturais.

- 3. Para que é usado o log do sistema? Quais são os tipos característicos de registros em um log do sistema? O que são pontos de confirmação da transação e por que eles são importantes?**

Para recuperar um antigo estado do banco de dados. Os tipos característicos são Read, Write, O dado utilizado para escrita, O antigo dado utilizado na escrita, Commit e Abort. Os pontos de confirmação ou commit são pontos no arquivos onde está decretado que todas as transações ali feitas estão salvas e são importantes em casos de falhas, para quando for necessário recuperar os dados, transações que sofreram commit já estão salvas no disco.

**4. Discuta as propriedades de atomicidade, durabilidade, isolamento e preservação da consistência de uma transação de banco de dados**

Atomicidade: Uma transação é entregue completamente ou nada dela é entregue, ou seja, se uma parte da transação falhar, a transação inteira é revertida para o estado anterior.

Durabilidade: Uma transação esteja permanentemente salva no banco de dados e não seja perdida mesmo em caso de falha.

Isolamento: Permite que várias transações executem simultaneamente sem ocorrer interferência uma com a outra no resultado final.

Preservação da consistência: Garante que o banco de dados esteja em um estado consistente antes e depois da execução de uma transação.

**5. O que é um schedule (plano de execução)? Defina os conceitos de schedules recuperáveis, sem cascara e estritos, e compare-os em matéria de sua facilidade de recuperação.**

É uma forma de representar uma ordem de transações que permita o banco de dados se manter funcionando corretamente. Schedules recuperáveis ocorrem em situações que uma transação dependente de outra, só realize o commit, após o commit da qual a transação está dependente; Sem cascata, não há dependência entre as transações, com isso, mesmo que uma transação falhe, nada alterará nas outras; Estrito, define que nenhuma outra transação consiga ler ou escrever um atributo que já foi gravado em alguma outra transação. O Sem cascata é mais fácil, pois caso uma transação venha a falhar, não afetará as outras.

**6. O que é um schedule serial? O que é um schedule serializável? Por que um schedule serial é considerado correto? Por que um schedule serializável é considerado correto?**

É um esquema onde ocorrem uma transação por vez. É um esquema onde ocorrem transações simultaneamente sem alterar o resultado final. Ambas são métodos corretos pois entregam o mesmo resultado, porém de forma diferente.

**7. Observe as transações T1 e T2 no Slide XX e defina um schedule que é serializável com intercalação e outro que não seja.**

Com intercalação:

T1: Ler\_item(X);

T1:  $X := X - N$ ;

T1: Escrever\_item(X);

T2: Ler\_item(X);

T2:  $X := X + M$ ;

T2: Escrever\_item(X);

T1: Ler\_item(Y);

T1:  $Y := Y + N$ ;

T1: Escrever\_item(Y);

Sem intercalação:

T1: Ler\_item(X);  
T1:  $X := X - N$ ;  
T1: Escrever\_item(X);  
T1: Ler\_item(Y);  
T1:  $Y := Y + N$ ;  
T1: Escrever\_item(Y);  
T2: Ler\_item(X);  
T2:  $X := X + M$ ;  
T2: Escrever\_item(X);

**8. Qual dos seguintes schedules é serializável (com intercalação)? Para cada schedule serializável, determine os schedules seriais equivalentes.**

**a.  $r1(X)$ ;  $r3(X)$ ;  $w1(X)$ ;  $r2(X)$ ;  $w3(X)$ ;**

Não é serializável devido conflito entre T1 e T3. Nota-se ciclo entre  $T3 \rightarrow T1 \rightarrow T2 \rightarrow T3$

**b.  $r1(X)$ ;  $r3(X)$ ;  $w3(X)$ ;  $w1(X)$ ;  $r2(X)$ ;**

Não é serializável devido conflito entre T3 e T1. Nota-se ciclo entre  $T1 \rightarrow T3 \rightarrow T1$

**c.  $r3(X)$ ;  $r2(X)$ ;  $w3(X)$ ;  $r1(X)$ ;  $w1(X)$ ;**

É serializável pois T1 leu e atualizou item x após operações T3. Não há ciclo  $T2 \rightarrow T3 \rightarrow T1$

**d.  $r3(X)$ ;  $r2(X)$ ;  $r1(X)$ ;  $w3(X)$ ;  $w1(X)$ ;**

Não é serializável devido conflito entre T3 e T1. Nota-se ciclo  $T1 \rightarrow T2 \rightarrow T1$

**9. Considere as três transações T1, T2 e T3, e os schedules S1 e S2 a seguir. Determine se cada schedule é estrito, sem cascata, recuperável ou não recuperável. A seguir, determine a condição de facilidade de recuperação mais estrita que cada schedule satisfaz.**

**S1:  $r1(X)$ ;  $r2(Z)$ ;  $r1(Z)$ ;  $r3(X)$ ;  $r3(Y)$ ;  $w1(X)$ ;  $c1$ ;  $w3(Y)$ ;  $c3$ ;  $r2(Y)$ ;  $w2(Z)$ ;  $w2(Y)$ ;  $c2$ ;**

Recuperável: não há *commit* de transação que depende de item modificado por outras transações;

Livre de cascata: no item onde há concorrência (item Y), ocorre primeiro o *commit* da transação que o escreve, posteriormente leituras e escritas desse item por outra transação;

Estrita: depois de um item escrito por uma transação nenhuma outra transação lê ou escreve esse item. No item Y, houve o *commit* da transação que o escreve antes de posteriores leituras ou escritas desse item.

**S2: r1 (X); r2 (Z); r1 (Z); r3 (X); r3 (Y); w1 (X); w3 (Y); r2 (Y); w2 (Z); w2 (Y); c1; c2; c3;**

Não recuperável porque ocorre o commit de T2 ocorre antes de T3, se T3 aborta T2 não pode mais ser revertido. Tornamos o plano recuperável fazendo:

r1 (X); r2 (Z); r1 (Z); r3 (X); r3 (Y); w1 (X); w3 (Y); r2 (Y); w2 (Z); w2 (Y); c1; c3; c2;

O plano acima não é livre de cascata pois ao abortar T3 tem de abortar T2, então;

r1 (X); r2 (Z); r1 (Z); r3 (X); r3 (Y); w1 (X); w3 (Y); c3; r2 (Y); w2 (Z); w2 (Y); c1; c2;

O plano acima também se torna restrito pelos mesmos motivos do plano S1.