Lista de Exercícios (Respostas)

MC536 – Bancos de Dados: Teoria e Prática Instituto de Computação Universidade Estadual de Campinas		Transações, Concorrência e Recuperação 2014 André Santanchè

Para cada uma das questões a seguir que solicitar um plano de execução, utilize o identificador da transação (exemplo, T1: ou T2:) antes de cada instrução no plano. As instruções que podem ser usadas no plano (quando necessário) são: read (leitura), write (gravação), lock/unlock (bloqueio binário), rlock/wlock/unlock (bloqueio compartilhado exclusivo), start (início transação), commit (transação recebe commit), abort (transação abortada). Cada instrução que envolver uma tupla da tabela deve indicar a tupla pela sua chave primária. Exemplos de operações em um plano (considerando que X e Y são valores de chaves primárias):

```
T1: read(X)
T2: write(Y)
```

Questão 1 (questão de prova)

Considere a seguinte tabela que registra saldos de conta corrente, em que é registrado o id da conta e seu saldo. A tabela só tem três registros apresentados abaixo:

ContaCorrente	
contaid	**saldo**
X	500
Y	200
Z	350

```
CREATE TABLE ContaCorrente (
  contaid VARCHAR(5) NOT NULL ,
  saldo FLOAT,
  PRIMARY KEY (contaid) );

INSERT INTO ContaCorrente VALUES ("X", 500);
INSERT INTO ContaCorrente VALUES ("Y", 200);
INSERT INTO ContaCorrente VALUES ("Z", 350);
```

Considere as duas transações sendo executadas de forma concorrente sobre estas tabelas em que podem ser intercaladas operações de leitura e gravação.

T1: Transferência da conta Y para a conta X

```
UPDATE ContaCorrente
       SET saldo = saldo - 100
       WHERE contaid = "Y";
UPDATE ContaCorrente
       SET saldo = saldo + 100
       WHERE contaid = "X";
```

T2: Somatório de todos os saldos de conta

```
CREATE VIEW SaldoTotal AS
    SELECT SUM(saldo) FROM ContaCorrente;
```

Responda as seguintes questões a seguir (na prova o aluno escolheu duas entre três para responder):

a) Monte um plano de execução que apresente problema de isolamento. Indique de forma sintética qual o problema. Indique que característica deve ter um plano para evitar este problema e apresente o plano corrigido.

b) Apresente um plano com problemas de deadlock (explicite os bloqueios e desbloqueios no plano), indique onde será o deadlock e indique que tipo de plano 2PL pode ser adotado para evitá-lo e porquê.

Para as Questões 2 e 3

A seguinte sequência `Transferência` é executada inteira por uma transação:

```
UPDATE ContaCorrente
       SET Saldo = Saldo - valor
       WHERE contaid = contaOrigem;
UPDATE ContaCorrente
       SET Saldo = Saldo + valor
       WHERE contaid = contaDestino;
INSERT INTO Transferencia
       VALUES (transferId, contaOrigem, contaDestino, valor);
```

Questão 2

Considere que duas transações estão executando a sequência de `Transferência` de forma concorrente com os seguintes valores:

```
T1 (transferId:'1122', valor:50, contaOrigem:'12345', contaDestino:'54321')

T2 (transferId:'7070', valor:30, contaOrigem:'54321', contaDestino:'12345')
```

Considerando que cada tupla da tabela é um item de dados a ser controlado independentemente:

a) Escreva um plano de execução serial relacionado a estas duas chamadas.

b) Há algum outro plano execução para estas chamadas que seja serializável? Se existir escreva os possíveis planos usando técnicas de equivalência baseadas em conflito e em visão. Justifique como pode comprovar que são equivalentes.

c) Escreva um possível plano que gere problemas de Isolamento e justifique como isso pode acontecer.

d) Escreva um possível plano que gere deadlock e justifique como isso pode acontecer.

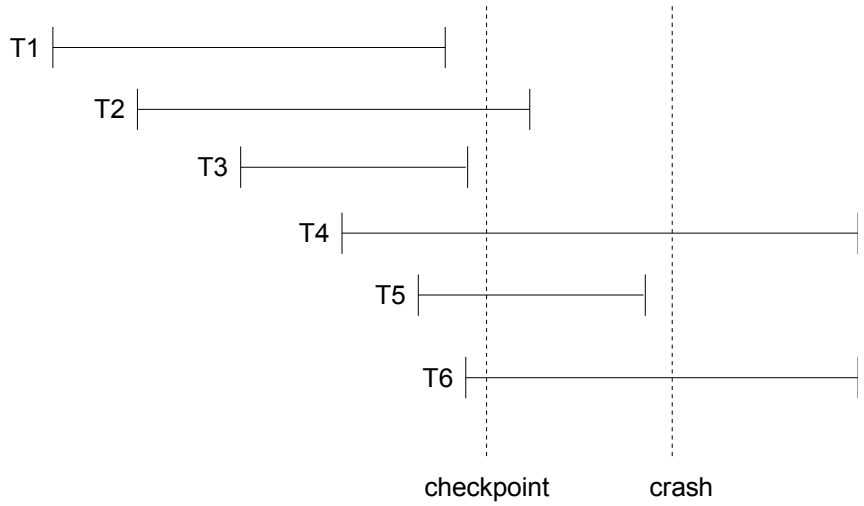
Questão 3

Considere as seguintes transações executando a sequência `Transferência`:

```
T1 (transferId:'1122', valor:50, contaOrigem:'12345', contaDestino:'54321')

T2 (transferId:'7070', valor:30, contaOrigem:'54321', contaDestino:'12345')

T3 (transferId:'1756', valor:70, contaOrigem:'54321', contaDestino:'22222')

T4 (transferId:'2198', valor:85, contaOrigem:'87654', contaDestino:'54637')

T5 (transferId:'4563', valor:90, contaOrigem:'17322', contaDestino:'54321')

T6 (transferId:'5968', valor:10, contaOrigem:'17322', contaDestino:'33333')
```

Os saldos das contas antes de sua aplicação são: (12345 → 700); (54321 → 450); (22222 → 1.500); (87654 → 200); (54637 → 820); (17322 → 330); (33333 → 100)

Analise o diagrama abaixo representando a execução concorrente de transações e responda:



a) Qual será o saldo das contas após a restauração do banco.

As operações podem ter sido executadas em ordem diferente, mas no caso específico destas operações de transferência, no final de tudo não importará a ordem para obtermos os mesmos resultados. Se não houvesse crash, os resultados seriam:

Conta	Antes	Operações	Depois
12345	700	(T1,'1122',50->,650);(T2,'7070',30<-,680)	680
54321	450	(T1,'1122',50<-,500);(T2,'7070',30->,470); (T3,'1756',70->,400);(T5,'4563',90<-,490)	490
22222	1500	(T3,'1756',70<-,1570)	1570
87654	200	(T4,'2198',85->,115)	115
54637	820	(T4,'2198',85<-,905)	905
17322	330	(T5,'4563',90->,240);(T6,'5968',10->,230)	230
33333	100	(T6,'5968',10<-,110)	110

Se houver crash e for possível recuperar as transações, os efeitos de T1, T2, T3 e T5 têm que se tornar permanentes, já que sofreram COMMIT antes do crash e os efeitos de T4 e T6 têm que ser desfeitos. A planilha a seguir mostra em vermelho as operações que teriam que ser desfeitas e qual seria o saldo das contas após a restauração do banco.

Conta	Antes	Operações	Depois
12345	700	(T1,'1122',50->,650);(T2,'7070',30<-,680)	680
54321	450	(T1,'1122',50<-,500);(T2,'7070',30->,470); (T3,'1756',70->,400);(T5,'4563',90<-,490)	490
22222	1500	(T3,'1756',70<-,1570)	1570
87654	200	(T4,'2198',85->,115)	200
54637	820	(T4,'2198',85<-,905)	820
17322	330	(T5,'4563',90->,240);(T6,'5968',10->,230)	240
33333	100	(T6,'5968',10<-,110)	100

b) Considerando que este é um plano serializável ele é: Restaurável? Livre de cascata? Estrito? Justifique.

Há várias possibilidades e cenários a serem analisados aqui. A seguir apresento um exemplo de cada.

Para que o plano seja restaurável:

- T realiza commit somente depois que todas as transações cujos valores T leu realizam commit

A seguir uma situação em que este plano não seria restaurável:

(i) Em um plano serial T1 seria completamente executado antes de T2 ou vice-versa. Se a serialização for equivalente a T2 sendo executado antes de T1, então T1 leu valores modificados por T2 e realizou o COMMIT antes de T2, o que o tornaria não restaurável.

Para que o plano seja livre de cascata:

- T só lê valores que foram alterados por transações que já realizaram COMMIT

Neste caso, uma situação em que o plano não seria livre de cascata:

(ii) Se T2 executar a modificação na conta '54321' antes de T3 lê-la, então T3 necessariamente leria dados não commitados de T2 (já que T2 executa o COMMIT depois). Outra possibilidade é se T3 executar a modificação na conta '54321' antes de T2 e T2 ler este valor antes do COMMIT de T3.

Para que o plano seja estrito:

- T só lê e/ou grava valores que foram alterados por transações que já realizaram commit

Então valem as mesmas observações feitas para livre de cascata mas considerando a leitura e a gravação (não apenas a leitura) de quem leu o dado depois de modificado.

c) Se não for um dos planos de (b) que modificações teriam que ser feitas para alcançá-lo?

Em relação aos casos observados na letra (b). Para o plano restaurável:

(i) Em relação a T1 x T2, há duas modificações possíveis para tornar o plano restaurável, caso ele não esteja: ao produzir uma serialização equivalente T1 teria que realizar suas operações sobre as contas 12345 e 54321 antes de T2, caso contrário, T2 teria que realizar COMMIT antes de T1.

Para o plano livre de cascata:

(ii) Em relação a T2xT3, T3 teria que executar a modificação na conta '54321' antes de T2 e T2 só leria os dados desta conta depois que T3 executasse o COMMIT.

Para que o plano se torne estrito vale a mesma observação, já que a gravação de T2 só acontece depois da leitura.

d) Quais os valores após a restauração do banco nos planos que você fez em (c).