**2PL - 2 Phase Lock (Protocolo de alocação de trancas em duas fases)**

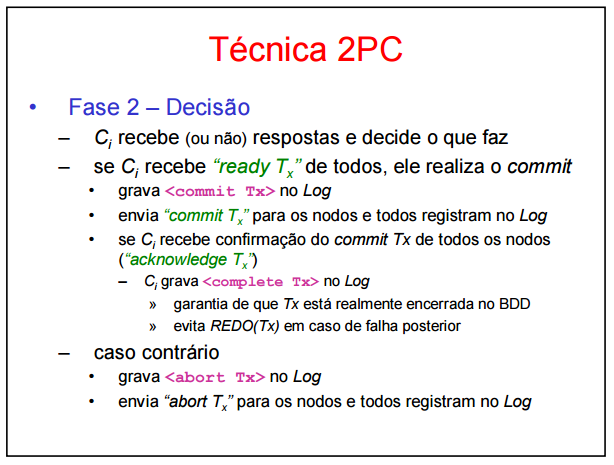
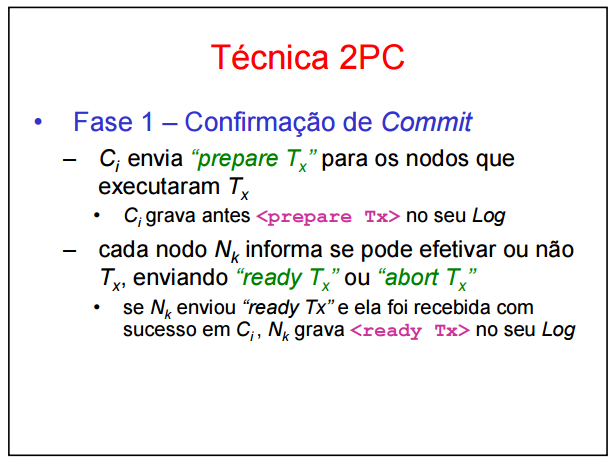
* Escalonamento seriável e recuperável.

Não libera nenhuma tranca até que todas as trancas estejam alocadas.

* **2PL Rigoroso:** Libera todas as trancas somente depois do commit;
* **2PL Estrito:** Libera trancas de escrita somente depois do commit;
* Previne rollback em cascata
* **2PL Conservativo:** Aloca todas as trancas antes de iniciar a execução.

**2PC – 2 Phase Commit (Protocolo de commit em duas fases)**

Caso a operação de uma transação falhe em algum nodo, todas outras operações desta transação nos outros nodos abortam.



**1º (6 ptos) -**

**Seja o BD distribuído de um sistema de aprendizado vi Web, com o esquema global definido abaixo.**

***Benefiniario(idB, nome,uf)***

***Receita(idB, idT, valor, ano)***

***tipoReceita(idT, nome)***

**a) Apresente expressões em álgebra relacional definindo os fragmentos e sua alocação nos nodos SC e RS, considerando:**

***i.*  fragmentação horizontal da tabela *Benefiniario* segundo o valor de *uf* (estado do Brasil);**

***ii.* fragmentação horizontal da tabela *Receita* induzida pela fragmentação da tabela *Benefiniario;***

***iii.* replicação da tabela *tipoReceita* em todos os nodos*..***

b) Considerando as fragmentações descritas no item anterior, forneça expressões SQL a serem processadas nos nodos *SC* e *RS* para solucionar a consulta abaixo, submetida ao nodo *SC*, onde é esperado o resultado. As expressões fornecidas devem minimizar o tráfego em rede.

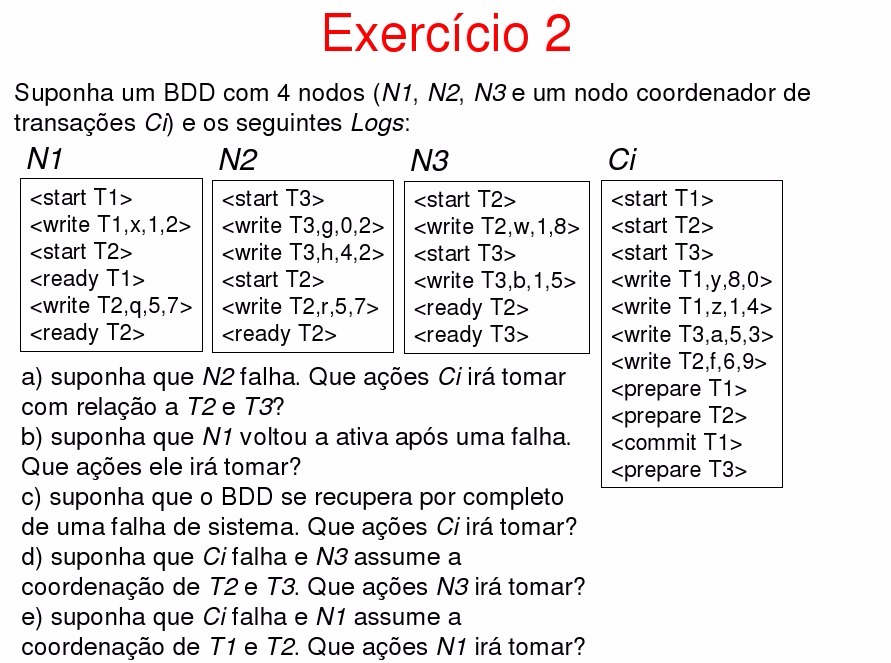
**(Obs: Parece que essa questão é mais complexa na prova, o exemplo abaixo é mais simples)**

**SELECT** D.idD, D.nome, COUNT(A.\*)AS nAlunos, AVG(nota) AS nNotas

**FROM** Aluno A NATURAL JOIN Cursa C NATURAL JOIN Disciplina D

**GROUP BY** D.idD, D.nome

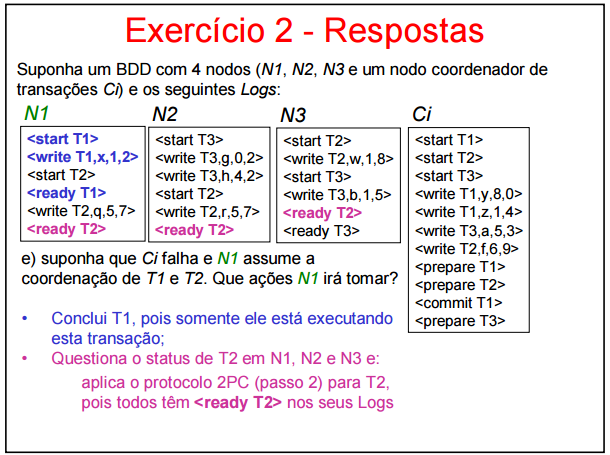
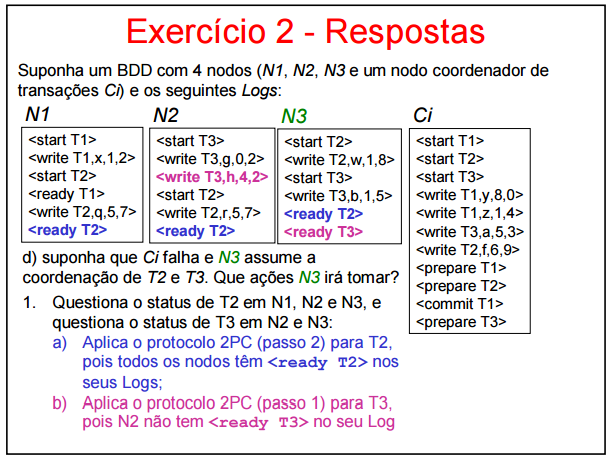
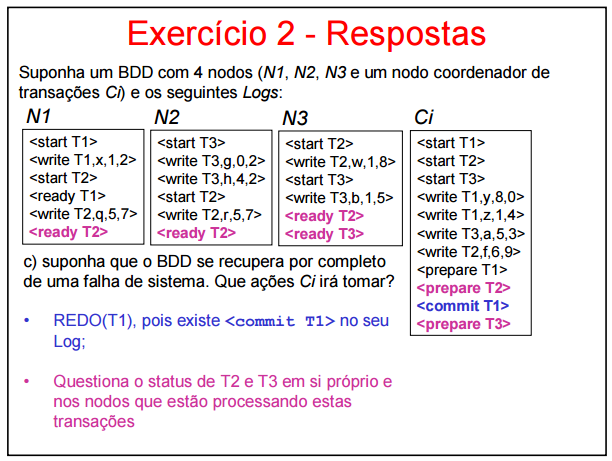
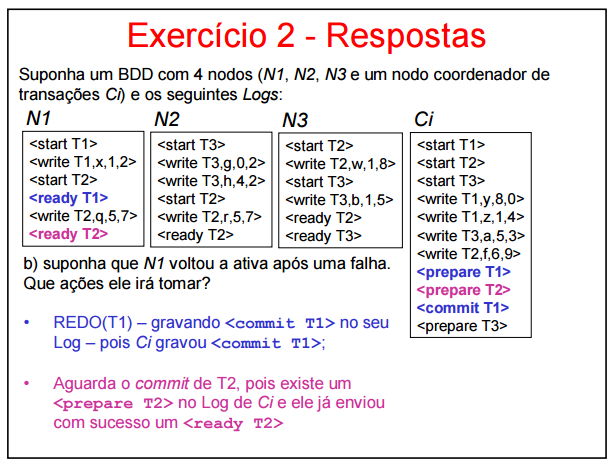
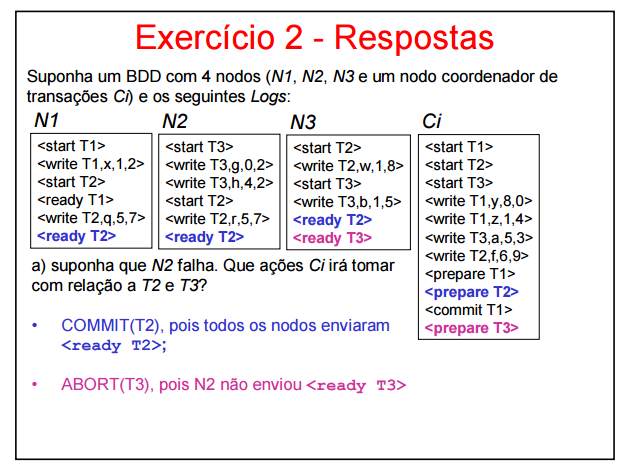
**2ª (4 ptos)**

****

**a)R:** Se N2 falhar, Ci conseguirá commitar T2 (pois N2 já havia feito <ready T2>, e Ci já havia feito <prepare T2>), porém não conseguirá commitar T3 devido ao N2 não ter feito <ready T3>. Considerando 2PC, todos os nodos que estiverem executando T3 farão rollback/abortarão esta transação, garantindo atomicidade.

**b)R:** Ci vai analisar o log de N1, e constatar que N1 havia feito <ready T1> e <ready T2>, assim podendo commitar já que Ci também fez <prepare T1> e <prepare T2>.

**c)R:** Ci continuara o commit de T1 e T2 (tendo em vista que os nodos já deram ready para as transações, e Ci já havia feito prepare), e continuará esperando pelo <ready T3> do nodo N2, até esgotar o time-out e assim fazer rollback/abortar T3 em todos os nodos.

**RESPOSTAS ENCONTRADAS NOS SLIDES DO PROF. RONALDO:**