

29/03/2015 BD2

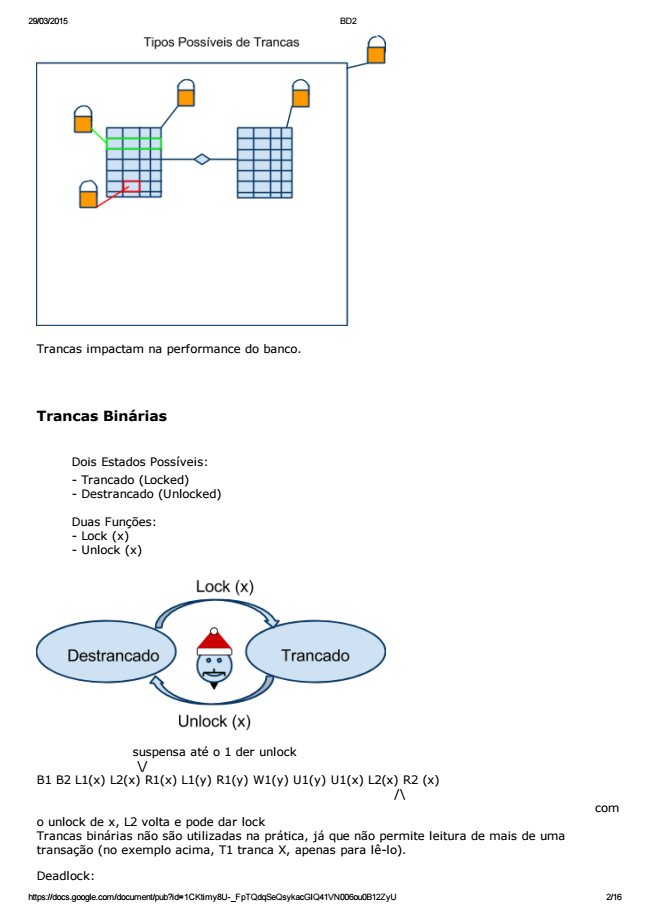
BD2

Parte 2

Controle de Transações Através de Trancas (Lock)

Tranca: Variável associada a um item de dado indicado se alguma transação está trabalhando com tal item.

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 1/16



29/03/2015 BD2

Trancas impactam na performance do banco.

Trancas Binárias

Dois Estados Possíveis: ­ Trancado (Locked) ­ Destrancado (Unlocked)

Duas Funções: ­ Lock (x) ­ Unlock (x)

suspensa até o 1 der unlock

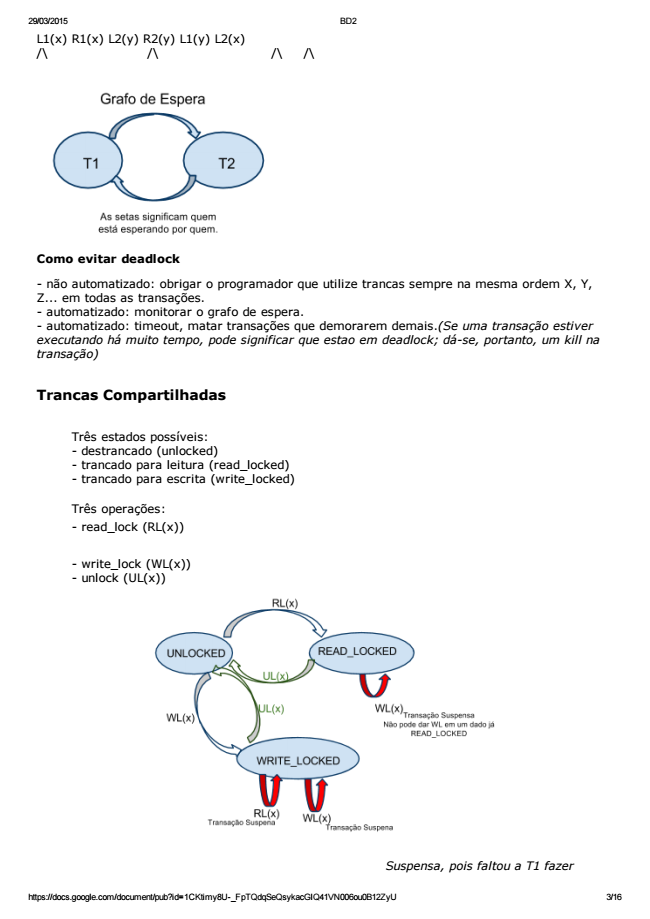
\/ B1 B2 L1(x) L2(x) R1(x) L1(y) R1(y) W1(y) U1(y) U1(x) L2(x) R2 (x)

/\

com o unlock de x, L2 volta e pode dar lock Trancas binárias não são utilizadas na prática, já que não permite leitura de mais de uma transação (no exemplo acima, T1 tranca X, apenas para lê­lo).

Deadlock:

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 2/16



29/03/2015 BD2

L1(x) R1(x) L2(y) R2(y) L1(y) L2(x) /\ /\ /\ /\

Como evitar deadlock

­ não automatizado: obrigar o programador que utilize trancas sempre na mesma ordem X, Y, Z... em todas as transações. ­ automatizado: monitorar o grafo de espera. ­ automatizado: timeout, matar transações que demorarem demais.(Se uma transação estiver executando há muito tempo, pode significar que estao em deadlock; dá­se, portanto, um kill na transação)

Trancas Compartilhadas

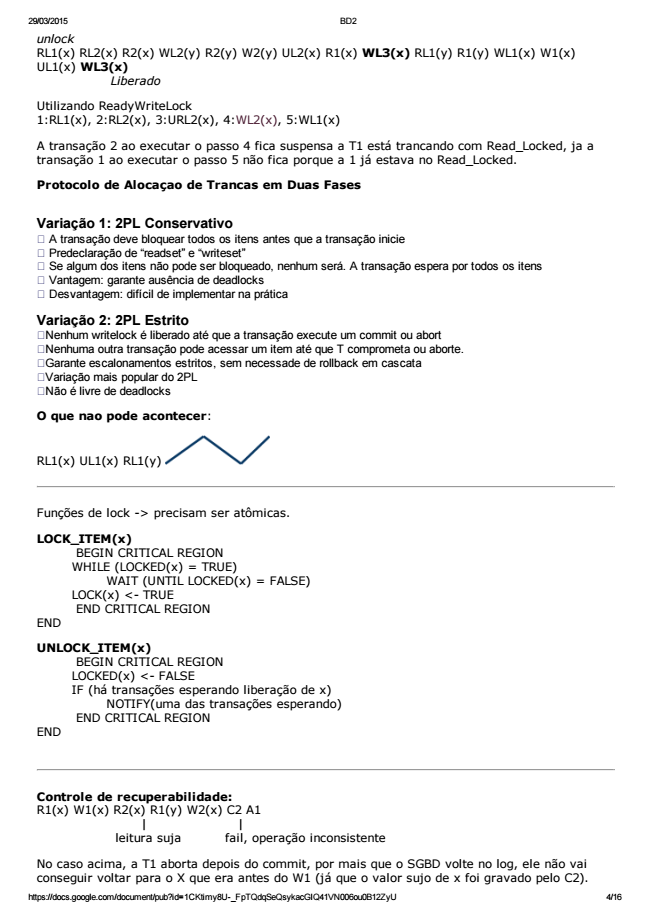
Três estados possíveis: ­ destrancado (unlocked) ­ trancado para leitura (read\_locked) ­ trancado para escrita (write\_locked)

Três operações: ­ read\_lock (RL(x))

­ write\_lock (WL(x)) ­ unlock (UL(x))

*Suspensa, pois faltou a T1 fazer*

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 3/16



29/03/2015 BD2

unlock RL1(x) RL2(x) R2(x) WL2(y) R2(y) W2(y) UL2(x) R1(x) WL3(x) RL1(y) R1(y) WL1(x) W1(x) UL1(x) WL3(x)

*Liberado*

Utilizando ReadyWriteLock 1:RL1(x), 2:RL2(x), 3:URL2(x), 4:WL2(x), 5:WL1(x)

A transação 2 ao executar o passo 4 fica suspensa a T1 está trancando com Read\_Locked, ja a transação 1 ao executar o passo 5 não fica porque a 1 já estava no Read\_Locked.

Protocolo de Alocaçao de Trancas em Duas Fases

Variação 1: 2PL Conservativo

A transação deve bloquear todos os itens antes que a transação inicie Pre​declaração de “read​set” e “write​set” Se algum dos itens não pode ser bloqueado, nenhum será. A transação espera por todos os itens Vantagem: garante ausência de deadlocks Desvantagem: difícil de implementar na prática

Variação 2: 2PL Estrito

Nenhum write​lock é liberado até que a transação execute um commit ou abort Nenhuma outra transação pode acessar um item até que T comprometa ou aborte. Garante escalonamentos estritos, sem necessade de rollback em cascata Variação mais popular do 2PL Não é livre de deadlocks

O que nao pode acontecer:

RL1(x) UL1(x) RL1(y)

Funções de lock ­> precisam ser atômicas.

LOCK\_ITEM(x)

BEGIN CRITICAL REGION WHILE (LOCKED(x) = TRUE)

WAIT (UNTIL LOCKED(x) = FALSE) LOCK(x) <­ TRUE

END CRITICAL REGION END

UNLOCK\_ITEM(x)

BEGIN CRITICAL REGION LOCKED(x) <­ FALSE IF (há transações esperando liberação de x)

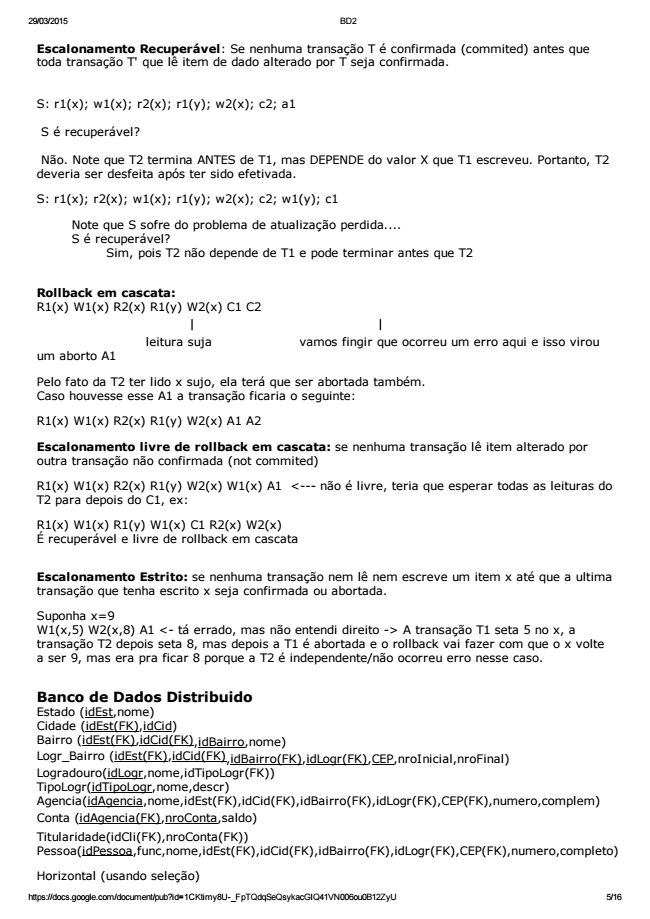
NOTIFY(uma das transações esperando) END CRITICAL REGION END

Controle de recuperabilidade: R1(x) W1(x) R2(x) R1(y) W2(x) C2 A1

| | leitura suja fail, operação inconsistente

No caso acima, a T1 aborta depois do commit, por mais que o SGBD volte no log, ele não vai conseguir voltar para o X que era antes do W1 (já que o valor sujo de x foi gravado pelo C2).

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 4/16



29/03/2015 BD2

Escalonamento Recuperável: Se nenhuma transação T é confirmada (commited) antes que toda transação T' que lê item de dado alterado por T seja confirmada.

S: r1(x); w1(x); r2(x); r1(y); w2(x); c2; a1

S é recuperável?

Não. Note que T2 termina ANTES de T1, mas DEPENDE do valor X que T1 escreveu. Portanto, T2 deveria ser desfeita após ter sido efetivada.

S: r1(x); r2(x); w1(x); r1(y); w2(x); c2; w1(y); c1

Note que S sofre do problema de atualização perdida.... S é recuperável?

Sim, pois T2 não depende de T1 e pode terminar antes que T2

Rollback em cascata: R1(x) W1(x) R2(x) R1(y) W2(x) C1 C2

| | leitura suja vamos fingir que ocorreu um erro aqui e isso virou um aborto A1

Pelo fato da T2 ter lido x sujo, ela terá que ser abortada também. Caso houvesse esse A1 a transação ficaria o seguinte:

R1(x) W1(x) R2(x) R1(y) W2(x) A1 A2

Escalonamento livre de rollback em cascata: se nenhuma transação lê item alterado por outra transação não confirmada (not commited)

R1(x) W1(x) R2(x) R1(y) W2(x) W1(x) A1 <­­­ não é livre, teria que esperar todas as leituras do T2 para depois do C1, ex:

R1(x) W1(x) R1(y) W1(x) C1 R2(x) W2(x) É recuperável e livre de rollback em cascata

Escalonamento Estrito: se nenhuma transação nem lê nem escreve um item x até que a ultima transação que tenha escrito x seja confirmada ou abortada.

Suponha x=9 W1(x,5) W2(x,8) A1 <­ tá errado, mas não entendi direito ­> A transação T1 seta 5 no x, a transação T2 depois seta 8, mas depois a T1 é abortada e o rollback vai fazer com que o x volte a ser 9, mas era pra ficar 8 porque a T2 é independente/não ocorreu erro nesse caso.

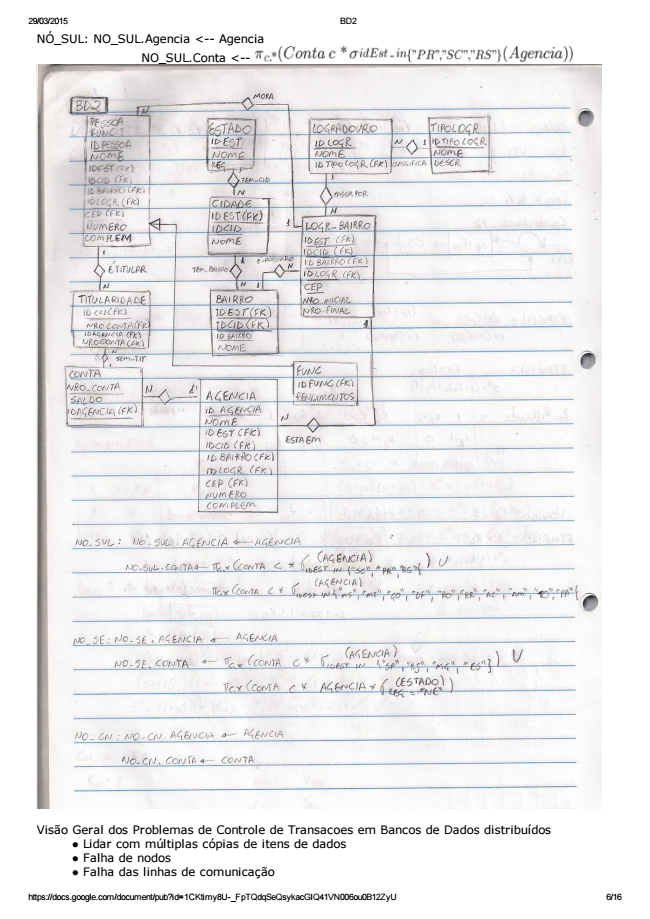
Banco de Dados Distribuido Estado (idEst,nome) Cidade (idEst(FK),idCid) Bairro (idEst(FK),idCid(FK)

,idBairro,nome) Logr\_Bairro (idEst(FK),idCid(FK)

,idBairro(FK),idLogr(FK),CEP,nroInicial,nroFinal) Logradouro(idLogr,nome,idTipoLogr(FK)) TipoLogr(idTipoLogr,nome,descr) Agencia(idAgencia,nome,idEst(FK),idCid(FK),idBairro(FK),idLogr(FK),CEP(FK),numero,complem) Conta (idAgencia(FK),nroConta,saldo) Titularidade(idCli(FK),nroConta(FK)) Pessoa(idPessoa,func,nome,idEst(FK),idCid(FK),idBairro(FK),idLogr(FK),CEP(FK),numero,completo)

Horizontal (usando seleção)

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 5/16



29/03/2015 BD2

NÓ\_SUL: NO\_SUL.Agencia <­­ Agencia

NO\_SUL.Conta <­­

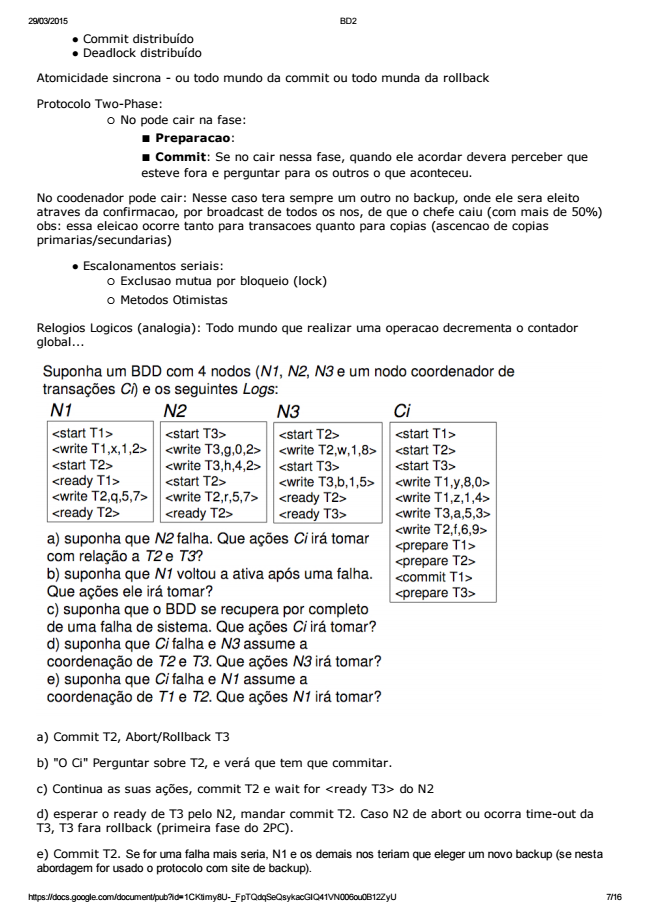
Visão Geral dos Problemas de Controle de Transacoes em Bancos de Dados distribuídos

● Lidar com múltiplas cópias de itens de dados

● Falha de nodos

● Falha das linhas de comunicação

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 6/16



29/03/2015 BD2

● Commit distribuído

● Deadlock distribuído

Atomicidade sincrona ­ ou todo mundo da commit ou todo munda da rollback

Protocolo Two­Phase:

○ No pode cair na fase:

■ Preparacao:

■ Commit: Se no cair nessa fase, quando ele acordar devera perceber que esteve fora e perguntar para os outros o que aconteceu.

No coodenador pode cair: Nesse caso tera sempre um outro no backup, onde ele sera eleito atraves da confirmacao, por broadcast de todos os nos, de que o chefe caiu (com mais de 50%) obs: essa eleicao ocorre tanto para transacoes quanto para copias (ascencao de copias primarias/secundarias)

● Escalonamentos seriais:

○ Exclusao mutua por bloqueio (lock)

○ Metodos Otimistas

Relogios Logicos (analogia): Todo mundo que realizar uma operacao decrementa o contador global...

a) Commit T2, Abort/Rollback T3

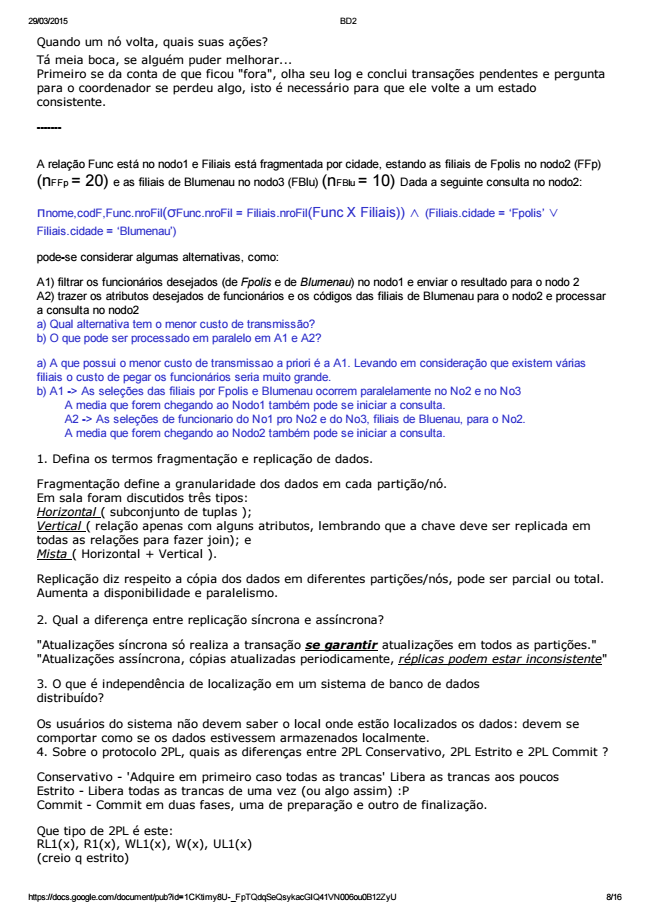
b) "O Ci" Perguntar sobre T2, e verá que tem que commitar.

c) Continua as suas ações, commit T2 e wait for <ready T3> do N2

d) esperar o ready de T3 pelo N2, mandar commit T2. Caso N2 de abort ou ocorra time­out da T3, T3 fara rollback (primeira fase do 2PC).

e) Commit T2. Se for uma falha mais seria, N1 e os demais nos teriam que eleger um novo backup (se nesta abordagem for usado o protocolo com site de backup).

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 7/16



29/03/2015 BD2

Quando um nó volta, quais suas ações? Tá meia boca, se alguém puder melhorar... Primeiro se da conta de que ficou "fora", olha seu log e conclui transações pendentes e pergunta para o coordenador se perdeu algo, isto é necessário para que ele volte a um estado consistente.

­­­­­­­

A relação Func está no nodo1 e Filiais está fragmentada por cidade, estando as filiais de Fpolis no nodo2 (FFp) (n

FFp

= 20)

e as filiais de Blumenau no nodo3 (FBlu)

(n

FBlu

= 10)

Dada a seguinte consulta no nodo2:

πnome,codF,Func.nroFil(σFunc.nroFil = Filiais.nroFil(Func X Filiais)) ∧ (Filiais.cidade = ‘Fpolis’ ∨ Filiais.cidade = ‘Blumenau’)

pode­se considerar algumas alternativas, como:

A1) filtrar os funcionários desejados (de Fpolis e de Blumenau) no nodo1 e enviar o resultado para o nodo 2 A2) trazer os atributos desejados de funcionários e os códigos das filiais de Blumenau para o nodo2 e processar a consulta no nodo2 a) Qual alternativa tem o menor custo de transmissão? b) O que pode ser processado em paralelo em A1 e A2?

a) A que possui o menor custo de transmissao a priori é a A1. Levando em consideração que existem várias filiais o custo de pegar os funcionários seria muito grande. b) A1 ­> As seleções das filiais por Fpolis e Blumenau ocorrem paralelamente no No2 e no No3

A media que forem chegando ao Nodo1 também pode se iniciar a consulta. A2 ­> As seleções de funcionario do No1 pro No2 e do No3, filiais de Bluenau, para o No2. A media que forem chegando ao Nodo2 também pode se iniciar a consulta.

1. Defina os termos fragmentação e replicação de dados.

Fragmentação define a granularidade dos dados em cada partição/nó. Em sala foram discutidos três tipos: Horizontal ( subconjunto de tuplas ); Vertical ( relação apenas com alguns atributos, lembrando que a chave deve ser replicada em todas as relações para fazer join); e Mista ( Horizontal + Vertical ).

Replicação diz respeito a cópia dos dados em diferentes partições/nós, pode ser parcial ou total. Aumenta a disponibilidade e paralelismo.

2. Qual a diferença entre replicação síncrona e assíncrona?

"Atualizações síncrona só realiza a transação se garantir atualizações em todos as partições." "Atualizações assíncrona, cópias atualizadas periodicamente, réplicas podem estar inconsistente"

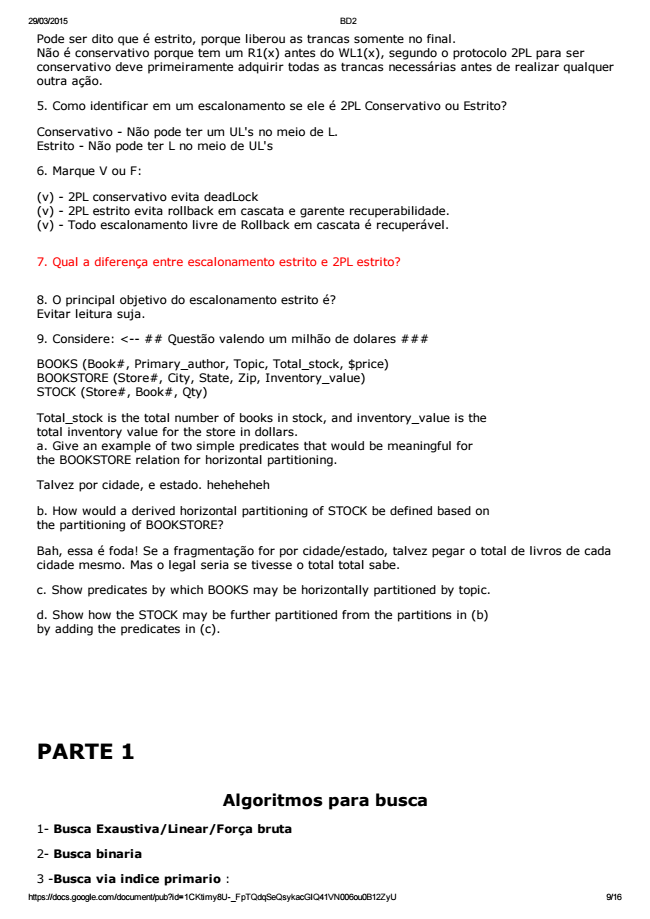
3. O que é independência de localização em um sistema de banco de dados distribuído?

Os usuários do sistema não devem saber o local onde estão localizados os dados: devem se comportar como se os dados estivessem armazenados localmente. 4. Sobre o protocolo 2PL, quais as diferenças entre 2PL Conservativo, 2PL Estrito e 2PL Commit ?

Conservativo ­ 'Adquire em primeiro caso todas as trancas' Libera as trancas aos poucos Estrito ­ Libera todas as trancas de uma vez (ou algo assim) :P Commit ­ Commit em duas fases, uma de preparação e outro de finalização.

Que tipo de 2PL é este: RL1(x), R1(x), WL1(x), W(x), UL1(x) (creio q estrito)

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 8/16



29/03/2015 BD2

Pode ser dito que é estrito, porque liberou as trancas somente no final. Não é conservativo porque tem um R1(x) antes do WL1(x), segundo o protocolo 2PL para ser conservativo deve primeiramente adquirir todas as trancas necessárias antes de realizar qualquer outra ação.

5. Como identificar em um escalonamento se ele é 2PL Conservativo ou Estrito?

Conservativo ­ Não pode ter um UL's no meio de L. Estrito ­ Não pode ter L no meio de UL's

6. Marque V ou F:

(v) ­ 2PL conservativo evita deadLock (v) ­ 2PL estrito evita rollback em cascata e garente recuperabilidade. (v) ­ Todo escalonamento livre de Rollback em cascata é recuperável.

7. Qual a diferença entre escalonamento estrito e 2PL estrito?

8. O principal objetivo do escalonamento estrito é? Evitar leitura suja.

9. Considere: <­­ ## Questão valendo um milhão de dolares ###

BOOKS (Book#, Primary\_author, Topic, Total\_stock, $price) BOOKSTORE (Store#, City, State, Zip, Inventory\_value) STOCK (Store#, Book#, Qty)

Total\_stock is the total number of books in stock, and inventory\_value is the total inventory value for the store in dollars. a. Give an example of two simple predicates that would be meaningful for the BOOKSTORE relation for horizontal partitioning.

Talvez por cidade, e estado. heheheheh

b. How would a derived horizontal partitioning of STOCK be defined based on the partitioning of BOOKSTORE?

Bah, essa é foda! Se a fragmentação for por cidade/estado, talvez pegar o total de livros de cada cidade mesmo. Mas o legal seria se tivesse o total total sabe.

c. Show predicates by which BOOKS may be horizontally partitioned by topic.

d. Show how the STOCK may be further partitioned from the partitions in (b) by adding the predicates in (c).

PARTE 1

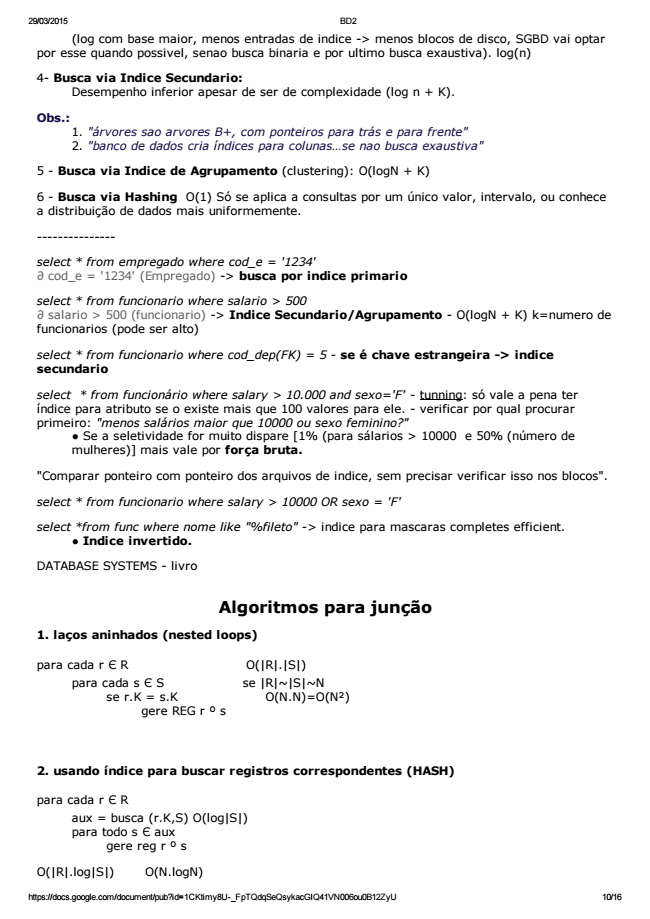
Algoritmos para busca

1­ Busca Exaustiva/Linear/Força bruta

2­ Busca binaria

3 ­Busca via indice primario :

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 9/16



29/03/2015 BD2

(log com base maior, menos entradas de indice ­> menos blocos de disco, SGBD vai optar por esse quando possivel, senao busca binaria e por ultimo busca exaustiva). log(n)

4­ Busca via Indice Secundario:

Desempenho inferior apesar de ser de complexidade (log n + K).

Obs.:

*1. "árvores sao arvores B+, com ponteiros para trás e para frente" 2. "banco de dados cria índices para colunas...se nao busca exaustiva"*

5 ­ Busca via Indice de Agrupamento (clustering): O(logN + K)

6 ­ Busca via Hashing O(1) Só se aplica a consultas por um único valor, intervalo, ou conhece a distribuição de dados mais uniformemente.

­­­­­­­­­­­­­­­

*select \* from empregado where cod\_e = '1234' ∂ cod\_e = '1234' (Empregado) ­> busca por indice primario*

select \* from funcionario where salario > 500 ∂ salario > 500 (funcionario) ­> Indice Secundario/Agrupamento ­ O(logN + K) k=numero de funcionarios (pode ser alto)

*select \* from funcionario where cod\_dep(FK) = 5 ­ se é chave estrangeira ­> indice secundario*

select \* from funcionário where salary > 10.000 and sexo='F' ­ tunning: só vale a pena ter índice para atributo se o existe mais que 100 valores para ele. ­ verificar por qual procurar primeiro: "menos salários maior que 10000 ou sexo feminino?"

● Se a seletividade for muito dispare [1% (para sálarios > 10000 e 50% (número de mulheres)] mais vale por força bruta.

"Comparar ponteiro com ponteiro dos arquivos de indice, sem precisar verificar isso nos blocos".

*select \* from funcionario where salary > 10000 OR sexo = 'F'*

*select \*from func where nome like "%fileto" ­> indice para mascaras completes efficient.*

● Indice invertido.

DATABASE SYSTEMS ­ livro

Algoritmos para junção

1. laços aninhados (nested loops)

para cada r Є R O(|R|.|S|)

para cada s Є S se |R|~|S|~N

se r.K = s.K O(N.N)=O(N2)

gere REG r o s

2. usando índice para buscar registros correspondentes (HASH)

para cada r Є R

aux = busca (r.K,S) O(log|S|) para todo s Є aux

gere reg r o s

O(|R|.log|S|) O(N.logN)

https://docs.google.com/document/pub?id=1CKtimy8U­\_FpTQdqSeQsykacGlQ41VN006ou0B12ZyU 10/16