

Aspectos de Implementação de Banco de Dados

Profa. Daniela Leal Musa

Transação

Unidade de execução indivisível que acessa e pode atualizar itens de dados.

Propriedades: (ACID)

Atomicidade

Consistência

Isolamento

Durabilidade

Execução simultânea de transações

Vantagens:

- Melhora o throughput* de transações
- Melhora utilização dos recursos
- Reduz tempo de espera das transações

• Problemas:

- Garantir a consistência
- Lost Update, Inconsistent Retrieval, Phantom

* número de transações executadas em determinado período de tempo

Como identificar execuções que garantem a consistência?

- Execuções SERIALS:
 - Execuções corretas
 - Executadas individualmente
 - A execução de uma transação não interfere na execução da outra

MAU USO DOS RECURSOS

Como identificar execuções que garantem a consistência?

- Execuções SERIALIZÁVEL:
 - Execuções concorrentes
 - Equivalente a uma execução serial
 - Execuções corretas

Execução Serial

t1	t2
read(a) a=a-50 write(a) read(b) b=b+50 write(b) commit	read(a) temp=a*0,1 a=a-temp write(a) read(b) b=b+temp write(b) commit

Início:

a-1000

b-2000

Fim:

a-855

b-2145

Execução Concorrente e correta

t1	t2
read(a) a=a-50 write(a)	read(a) temp=a*0,1 a=a-temp write(a)
read(b) b=b+50 write(b)	read(b) b=b+temp write(b)

Início: a:1000

b:2000

Fim: a:855

b:2145

Histórias de Transações

- Mecanismo que representa a execução concorrente de um conjunto de transações
- Operações consideradas:
 - ri – read - wi – write - ci- commit - ai- abort
 - setas indicam a ordem das operações
 - as letras do alfabeto representam os itens de dados usados na transação
 - somente os nomes são considerados (valores não)

Histórias de Transações

- Exemplo: $ri[x]$ – representa a leitura feita pela transação i no dado x .
- Modelando transações na História:

Procedure p	T1
START	
Temp= read(x)	$r1[x]$
temp=temp+1	
write(x, temp)	$w1[x]$
Commit	$c1$
END	

História de T1 = $r1[x] \rightarrow w1[x] \rightarrow c1$

Histórias

- Histórias Completas: modela a execução completa de cada T_i em T . Não possui transações ativas, somente validadas ou anuladas
 - $r1[x] \rightarrow r2[y] \rightarrow w1[x] \rightarrow w2[y] \rightarrow c1 \rightarrow a2$
- Histórias Incompletas: possui transações ativas
 - $r1[x] \rightarrow r2[y] \rightarrow w1[x] \rightarrow w2[y]$
- Projeção committed: parte de H formada por operações de T_i que já terminaram com sucesso
- $r1[x] \rightarrow w1[x] \rightarrow c1$

Conflitos de Transações

- Uma operação p conflita com outra operação q na história H , se as duas fazem acesso ao mesmo item de dado e uma delas é uma operação de escrita.

	$ri[x]$	$wi[x]$
$rj[x]$	Sem conflito	conflito
$wj[x]$	conflito	conflito

Conflitos de Transações

- Se duas operações p e q conflitam então sua ordem importa
- Numa História, as operações em conflito devem ser ordenadas:
 - Numa mesma transação
 - Transações diferentes

Testando a serializabilidade

- Uma História H é serializável (SR) se sua projeção committed é equivalente a uma história serial.
- Como saber se uma H é serializável (equivalente a uma execução serial) ?
 - Equivalência entre histórias
 - Grafo de Serialização

Equivalência

- Considerando duas histórias H e H':
 - H e H' são definidas sobre o mesmo conjunto de transações e modelam as mesmas operações
 - H e H' ordenam operações conflitantes de transações ativas e committed da mesma forma

- Exemplo:

H1=r1(a)w1(a)r2(a)w2(a)r1(c)w1(c)r3(d)w3(d)r2(d)w2(d)c1 c2 c3

H2=r3(d)w3(d)r1(a)w1(a)r2(a)w2(a)r2(d)w2(d)r1(c)w1(c)c1 c2 c3

Grafo de Serialização

- Seja H uma História sobre um conjunto $T = \{t_1, t_2, \dots, t_n\}$ de transações.
- O grafo de serialização de H , $SG(H)$
 - Grafo direcionado
 - nodos são transações committed em $C(H)$
 - Arcos : indicam a ordem em que foram executadas as operações conflitantes.

Grafo de Serialização

- Dada uma história
 - Analisar os conflitos
 - Montar o grafo
 - se grafo Acíclico \rightarrow então H é serializável

Verificação de Serializabilidade

T1	T2
read(X)	
X = X - 20	
write(X)	
	read(X)
	X = X + 10
	write(X)
read(Y)	
Y = Y + 20	
write(Y)	



serializável

T1	T2
read(X)	
X = X - 20	
	read(X)
	X = X + 10
write(X)	
read(Y)	
	write(X)
Y = Y + 20	
write(Y)	



Não serializável

Exemplo

- $H = r2(x) \rightarrow r1(x) \rightarrow w1(x) \rightarrow r3(x) \rightarrow w3(x) \rightarrow$
 $r1(x) \rightarrow w1(x) \rightarrow r3(x) \rightarrow w3(x) \rightarrow$
 $w2(y) \rightarrow w1(y) \quad c1 \quad c2 \quad c3$

t1	t2	t3
	r(x)	
r(x)		
w(x)		
		r(x)
		w(x)
r(x)		
w(x)		
		r(x)
		w(x)
	w(y)	
w(y)		
c		
	c	
		c

Caracterizando histórias

- Serializável (SR)
- Recuperável (RC)
- Evitam aborto em cascata (ACA)
- Strict (ST)

Recuperável (RC)

- Base para a propriedade de Isolamento.
- Uma execução é dita recuperável se:
 - para cada transação T_i que termina com sucesso
 - T_i só termina após
 - todas as transações T_j , T_k , etc, das quais T_i leu resultados, terem terminado com sucesso.

Recuperável

- Exemplo

$r1(x) \rightarrow w1(x) \rightarrow r2(x) \rightarrow r1(y) \rightarrow w2(x) \rightarrow$
 $w1(y) \rightarrow c1 \rightarrow c2$

T2 lê de T1 e só termina depois de T1

r1(x)	
w1(x)	
	r2(x)
r1(y)	
	w2(x)
w1(y)	
c	
	c

Recuperável

- Exemplo de Não Recuperável:

$r1(x) \rightarrow w1(x) \rightarrow r2(x) \rightarrow r1(y) \rightarrow w2(x) \rightarrow c2 \rightarrow$
 $w1(y) \rightarrow c1$

Não recuperável, pois T2 lê e termina antes de T1

r1(x)	
w1(x)	
	r2(x)
r1(y)	
	w2(x)
	c
w1(y)	
c	

Evita Aborto em Cascata (ACA)

- Base para uma recuperação eficiente
- SGBD evita abortos em cascata se:
 - Cada Ti só lê dados escritos por Transações que já terminaram com sucesso.

Evita Aborto em Cascata (ACA)

- Exemplo 1

$r1(x) \rightarrow w1(x) \rightarrow r2(x) \rightarrow r1(y) \rightarrow w2(x) \rightarrow w1(y) \rightarrow c1 \rightarrow c2$

- Não evita aborta em cascata
- Pois T2 lê antes de T1 dar commit

r1(x)	
w1(x)	
	r2(x)
r1(y)	
	w2(x)
w1(y)	
c	
	c

Evita Aborto em Cascata (ACA)

- Exemplo 2

$w1(x) \rightarrow w1(y) \rightarrow r2(u) \rightarrow w2(u) \rightarrow w1(x) \rightarrow c1$
 $\rightarrow r2(y) \rightarrow w2(y) \rightarrow c2$

Evita aborto em cascata

Pois T2 lê depois de T1 dar commit

w1(x)	
w1(y)	
	r2(u)
	w2(u)
w1(x)	
c	
	r2(y)
	w2(y)
	c

Execuções STRICT (ST)

- Base para a recuperação
 - Uma execução concorrente é ST se:
 - A execução de operações de $ri(x)$ ou $wi(x)$ é postergada até que todas as transações t_j, t_k que executaram $w(x)$ antes de T_i , tenham terminado.

Strict (ST)

- Exemplo:

$w1(x) \rightarrow w1(y) \rightarrow r2(u) \rightarrow w1(z) \rightarrow c1 \rightarrow r2(x) \rightarrow w2(x)w2(y) \rightarrow c2$

STRICT, pois T2 espera o commit de t1 para executar $w(x)$ e $r(y)$ $w(y)$

w1(x)	
w1(y)	
	r2(u)
w1(z)	
c	
	r2(x)
	w2(x)
	w2(y)
	c

Strict (ST)

- Exemplo:
- $w1(x) \rightarrow w1(y) \rightarrow r2(u) \rightarrow w2(x) \rightarrow r2(y) \rightarrow w2(y) \rightarrow$
- $w1(z) \rightarrow c1 \rightarrow c2$
- Não STRICT, pois T2 não espera o commit de t1 para executar $w(x)$ e $r(y)$ $w(y)$

w1(x)	
w1(y)	
	r2(u)
	w2(x)
	r2(y)
	w2(y)
w1(z)	
c	
	c



Uma História SR pode ser RC, ACA e ST,
Uma história RC, ACA ou ST pode não ser SR.

Exercício 1

- Dada as transações abaixo, ordene as operações de modo que a execução concorrente seja RC.
 - T1: $r(x) \rightarrow w(x) \rightarrow r(y) \rightarrow w(y) \rightarrow c$
 - T2: $r(z) \rightarrow r(x) \rightarrow w(x) \rightarrow c$
- Apresente uma execução ACA e uma ST

Exercício 1 - Resposta

	$r(z)$
$r(x)$	
$w(x)$	
	$r(x)$
	$w(x)$
$r(y)$	
$w(y)$	
c	
	c

RC

	$r(z)$
$r(x)$	
$w(x)$	
$r(y)$	
$w(y)$	
c	
	$r(x)$
	$w(x)$
	c

ACA e ST

Exercício 2

Dadas as transações abaixo, associe corretamente a história com o tipo de escalonamento :

• SR = escalonamento serializável • R = escalonamento recuperável • ACA = evita aborto em cascata • ST = escalonamento strict • S = escalonamento serial

T1 = w(x) w(y) w(z) c1

T2 = r(u) w(x) r(y) w(y) c2

HE1 = w1(x) w1(y) r2(u) w2(x) r2(y) w2(y) c2 w1(z) c1

HE2 = w1(x) w1(y) w1(z) c1 r2(u) w2(x) r2(y) w2(y) c2

HE3 = w1(x) w1(y) r2(u) w2(x) w1(z) c1 r2(y) w2(y) c2

HE4 = w1(x) w1(y) r2(u) w1(z) c1 w2(x) r2(y) w2(y) c2

HE5 = w1(x) w1(y) r2(u) w2(x) r2(y) w2(y) w1(z) c1 c2

2. Dadas as transações, dê um exemplo, envolvendo todas elas, de uma história não-serial:

a) não-serializável

b) serializável e não-recuperável

c) evita aborto em cascata

T1: read(X);write(X); write(Y)

T2: read(X); write(Y)

T3: read(X); write(X)

Exercício 2

$w(x)$	
$w(y)$	
	$r(u)$
	$w(x)$
	$r(y)$
	$w(y)$
	c
$w(z)$	
c	