

Escalonamento

Elmasri – Capítulos 21 e 22

Ramakrishnan – Capítulos 16 e 17

Silberchatz – Capítulos 15 e 16

Complete Book – Chapter 18



Introdução

- Controle de concorrência assegura o *isolamento* das transações
- Garantem a serialização dos escalonamentos das transações
 - Uso de protocolos (conjunto de regras)

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

Escalonamento

- Um escalonador é uma sequência de operações realizadas por uma ou mais transações ordenadas em relação ao tempo

Definição: escalonador é dito serial se **não** existe intercalação de operações

Escalonamento serial

Considere as seguintes transações e as possíveis execuções **sequenciais** :

T_0 : Read (A)

A: A-100

Write (A) 800

Read (B)

B: B + 100

Write (B) 2200

T_1 : Read (A)

x: A * 0.10 100

A: A - x (redução 10%) **900**

Write (A)

Read (B) 2000

B: B + x (incremento 10% de A) **2100**

Write (B)

Valores	A	B
Iniciais	1000	2000
$T_0 \rightarrow T_1$	810	2190
$T_1 \rightarrow T_0$	800	2200

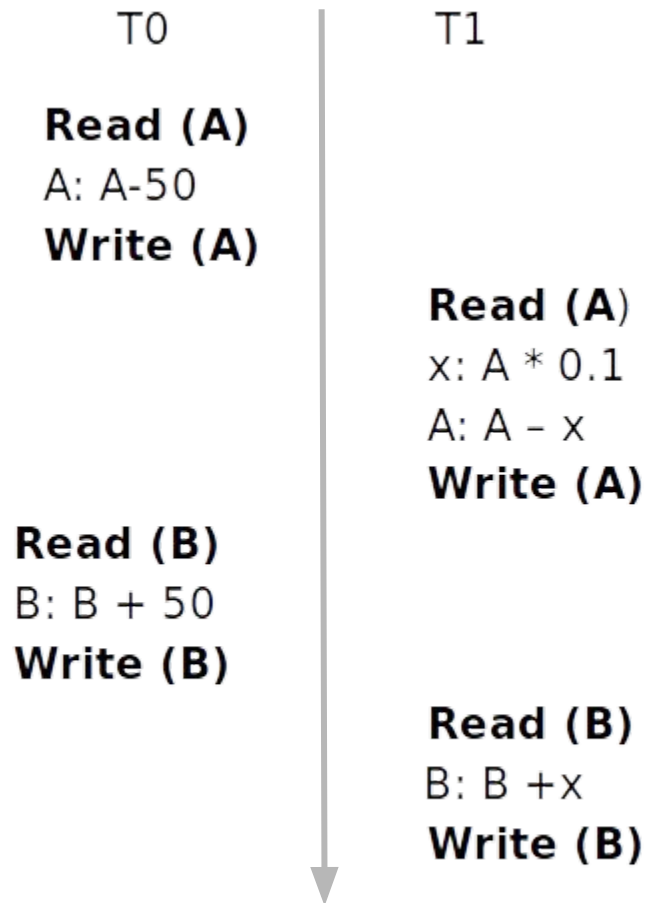
Escalonamento não-serial



O que é: Um escalonamento com operações concorrentes com o mesmo efeito de transações seriais.

Um resultado correto é obtido por escalonadores concorrentes sempre que o resultado obtido seja **igual** ao produzido por um escalonador serial.

Escalonamento não-serial



Valores	A	B
Iniciais	1000	2000
T0 -> T1		

Escalonamento não-serial?

T_0

Read (A)

A: A-50

Write (A)

Read (B)

B: B + 50

Write (B)

T_1

Read (A)

x: A * 0.1

A: A - x

Write (A)

Read (B)

B: B + x

Write (B)

Valores A B
Iniciais 1000 2000
 $T_0 \rightarrow T_1$

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

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

Escalonamento com conflito de serialibidade

Diz-se que duas operações são **conflitantes** se elas operam sobre o mesmo item de dados, sendo que no mínimo uma delas é uma gravação, e são emitidas por **diferentes transações**.

Usaremos a seguinte notação :

- $R_i(x)$ - para operação de leitura do item x realizada pela transação i .
- $W_i(x)$ - para operação de gravação do item x realizada pela transação i .

Escalonamento com conflito de serialidade

Diz-se que duas operações são **conflitantes** se elas operam sobre o mesmo item de dados, sendo que no mínimo uma delas é uma gravação, e são emitidas por **diferentes transações**.

E1 : R1(x), R2(x), W2(x), W1(x)

T1	T2
R(X)	
	R(X)
w(X)	

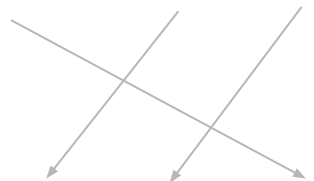
Escalonamento com conflito de serialibidade

Se **A** precede **B** e são instruções diferentes e *não conflitantes* então pode-se trocar a ordem entre elas gerando-se assim um novo escalonador que diferem apenas na ordem destas operações.

Exemplo :

E1 : R1(x), R2(x), W2(y), W1(x)

E2 : R2(x), W2(y), R1(x), W1(x)



Escalonamento com conflito de serialibidade

Se I_i precede I_j e são instruções diferentes e *não conflitantes* então pode-se trocar a ordem entre elas gerando-se assim um novo escalonador que diferem apenas na ordem destas operações.

Exemplo :

E1 : **R1**(x), **R2**(x), W2(y), **W1**(x) (não-serial)

E2 : **R2**(x), W2(y), R1(x), **W1**(x) (serial)

Escalonadores serializáveis em conflito

“dado um escalonamento não-serial E1 para um conjunto de Transações T, **E1 é serializável** se a ordem de quaisquer 2 operações em conflito é a mesma em E1 e em algum escalonamento **serial** E.”

Escalonamento não-serial serializável

S- serial

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

S' - não serial

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

Exemplo

E1 é serializáveis!
E2 é não serializáveis?

escalonamento serial *E*

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

escalonamento não-serial *E1*

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

escalonamento não-serial *E2*

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

Protocolos Baseados em Bloqueio

- Idéia Básica

Quando uma transação acessa um item de dados deve antes bloqueá-lo, caso este já esteja bloqueado por outra transação deve esperar até que o item seja liberado

- Modos de Bloqueio

- Compartilhado **LS**
- Exclusivo **LX**



Protocolos Baseados em Bloqueio

Compartilhado - LS

Quando o item desejado não está bloqueado por nenhuma transação ou está bloqueado em modo compartilhado.

Exclusivo - LX

Somente quando o item desejado não está bloqueado

Transações Bem Formadas

- Aquelas que sempre bloqueiam o item de dados em modo compartilhado antes de lê-lo e sempre bloqueiam em modo exclusivo antes de gravá-lo
- Duas transações estão em conflito se elas desejam bloquear o mesmo item de dados em modos incompatíveis.

Exemplo 1

T1: LX1 (B)
R1 (B)
B: B-50
W1 (B)
UL1 (B) 150
LX1 (A)
R1 (A)
A: A + 50 **150**
W1 (A)
UL1 (A)

T2: LS2 (A) 150
R2 (A)
UL2 (A)
LS2 (B)
R2 (B)
UL2 (B)
Display (A + B)
300

Seja E1:

Valores Iniciais: A=100 B=200

LX1 (B) R1(B) W1 (B) UL1 (B) LX1 (A) R1 (A) W1 (A) UL1 (A) LS2 (A) R2 (A)
UL2 (A) LS2 (B) R2 (B) UL2 (B) Display (A + B)

O valor da linha "Display (A+B)" está correto?

Exemplo 2

T1: LX1 (B)
LX1 (A)
R1 (B)
B: B-50
W1 (B)
UL1 (B)
R1 (A)
A: A + 50
W1 (A)
UL1 (A)

T2: LS2 (A) 100
R2 (A)
UL2 (A)
LS2 (B)
R2 (B) 150
UL2 (B)
Display (A + B)
100+150=250

Seja E2:

Valores Iniciais: A=100 B=200

LX1 (B) R1(B) W1 (B) UL1 (B) LS2 (A) R2 (A) UL2 (A) LS2 (B) R2 (B) UL2 (B)
LX1 (A) R1 (A) W1 (A) UL1 (A) Display (A + B)

O valor da linha "Display (A+B)" está correto?

Exemplo 2

T1: LX1 (B)
R1 (B)
B: B-50
W1 (B)
LX1 (A)
R1 (A)
A: A + 50
W1 (A)
UL1 (B)
UL1 (A)

T2: LS2 (A) 100
R2 (A)
UL2 (A)
LS2 (B)
R2 (B) 150
UL2 (B)
Display (A + B)
100+150=250

Seja E2:


Valores Iniciais: A=100 B=200

LX1 (B) R1(B) W1 (B) UL1 (B) LS2 (A) R2 (A) UL2 (A) LS2 (B) R2 (B) UL2 (B)
LX1 (A) R1 (A) W1 (A) UL1 (A) Display (A + B)

O valor da linha "Display (A+B)" está correto?

Exemplo 2

Valores Iniciais: A=100 B=200



LX1 (B)
R1 (B)
B: B-50
W1 (B)
UL1 (B)
LS2 (A)
R2 (A)
UL2 (A)
LS2 (B)
R2 (B)
UL2 (B)
Display (A + B)
LX1 (A)
R1 (A)
A: A + 50
W1 (A)
UL1 (A)

Protocolo de Bloqueio Bifásico (2PL)

A execução concorrente de transações é correta se observada as seguintes regras:

1. Transações bem formadas
2. Regras de compatibilidade de bloqueio são obedecidas
3. Cada transação após liberar um bloqueio não solicita um novo bloqueio

A condição 3 pode ser expressa dizendo que as transações são *Bifasicamente Bloqueadas*

Protocolo de Bloqueio Bifásico (2PL)

Todas as transações devem obedecer as seguintes fases:

Primeira Fase – *Crescimento*

Durante a qual obtém seus bloqueios, mas não libera bloqueio algum.

Segunda Fase – *Retração*

Na qual os bloqueios são liberados mas nenhum bloqueio pode ser requerido

Exemplo

T1	T2	A	B
LX(A)		25	25
Read(A)			
A=A+100	LX(A)	125	
W(A)	LX(A)	125	
UL(A)	LX(A)		
	LX(A)		
	read(A)		
	A=A*2		50
	write(A)		
	UL (A)		

2PL

- Quais transações obedecem 2PL?

T_1	T_2	T_3	T_4
LS(Y) Read(Y) Unlock(Y) LX(X) Read(X) $X := X + Y$ Write(X) Unlock(X)	LS(X) Read(X) UL(X) LX(Y) Read(Y) $Y := X + Y$ Write(Y) Unlock(Y)	LS(Y) Read(Y) LX(X) UL(Y) Read(X) $X := X + Y$ Write(X) UL(X)	LS(X) LX(Y) Read(X) Read(Y) $Y := X + Y$ Write(Y) Unlock(X) Unlock(Y)

Atividade A

Os escalonadores abaixo seguem o 2PL?

A)	LS1(A)	B)	LS1(A)	C)	LS1(A)
	R1(A)		R1(A)		R1(A)
	LS2(A)		LX2(A)		LX2(A)
	LX1(B)		LX1(B)		UL1(A)
	UL1(A)		R1(B)		LX1(B)
	R1(B)		W1(B)		R1(B)
	W1(B)		UL1(A)		W1(B)
	R2(A)		UL1(B)		UL1(B)
	UL2(A)		R2(A)		R2(A)
	UL1(B)		W2(A)		W2(A)
			UL2(A)		UL2(B)

Atividade B

S1: r1(A), r2(D), w1(A), r2(C), r2(B), w2(B), w1(C)

1- O escalonador S1 respeita o protocolo 2PL usando somente bloqueio exclusivos?

2- Com bloqueios exclusivos e compartilhados, respeita o 2PL?

Resolução B-1

S1: r1(A), r2(D), w1(A), r2(C), r2(B), w2(B), w1(C)

1	2
LX(A)	
R(a)	
	LX(D)
	R(D)
W(A)	
	LX(B)

	R(B)
	W(B)
LX(C)	
w(C)	
UL(A)	
UL(C)	
	UL(D)
	UL(B)

Resolução B

S1: r1(A), r2(D), r3(B), w1(A), r2(C), r2(B), w2(B), w1(C)

1	2
---	---

Atividade C

S3: r²(A), r³(B), w¹(A), r²(C), r²(D), w¹(D)

3A- O escalonador S3 respeita o **protocolo 2PL** usando bloqueio **exclusivo**?

t1	t2	t3
	LX(A)	
	R(A)	
		LX(B)
		r(B)
	LX(C)	
	LX(D)	
	UL(A)	

t1	t2	t3
LX(A)		
w(A)		
	r(C)	
	r(D)	
	UL(C)	
	UL(D)	
LX(D)		
w(D)		

t1	t2	t3
UL(A)		
UL(D)		
		UL(B)

Atividade C'

S3: r²(A), r³(B), w¹(A), r²(C), r²(D), w¹(D), w²(A)

3A- O escalonador S3 respeita o **protocolo 2PL** usando bloqueio **exclusivo**?

t1	t2	t3
	LX(A)	
	R(A)	
		LX(B)
		r(B)
	LX(C)	
	LX(D)	
LX(A)		

t1	t2	t3

t1	t2	t3

Atividade C

S3: r2(A), r3(B), w1(A), r2(C), r2(D), w1(D)

3B- O escalonador S3 respeita o protocolo 2PL usando bloqueio exclusivo e compartilhado?

Seriabilidade e Isolamento

- O 2PL garante a **seriabilidade** de transações
- A propriedade de **isolamento** só é alcançada caso todos os bloqueios exclusivos sejam mantidos até a confirmação (commit). Tal como o esquema abaixo:

Begin Transaction

Aquisição de bloqueios antes de ler ou gravar

Commit

Libera bloqueios

OBS: A vulnerabilidade do 2PL a impasses continua

Impasse (Deadlock)



Considere que a transação T_i tenta bloquear X , mas X já está bloqueado por T_j

Esperar-morrer: transações mais antigas esperam, as mais novas são abortadas

- Se $TS(T_i) < TS(T_j)$ (T_i é mais velha que T_j) então
 T_i pode esperar

Senão

 aborta T_i (T_i recomeça mais tarde com o mesmo
 TS)

Impasse (Deadlock)



Considere que a transação T_i tenta bloquear X , mas X já está bloqueado por T_j

Ferir-esperar: transações mais novas esperam pelas antigas e as mais antigas abortam as mais novas (voltam com o mesmo TS)

IF $TS(T_i) < TS(T_j)$ (T_i é mais velha que T_j) então
aborta T_j (T_j recomeça mais tarde com o mesmo TS)

Senão

T_i pode esperar

Atividade

- 1- Crie uma situação de deadlock no postgres.
- 2- Descreva o que aconteceu. Qual das duas políticas foi aplicada?

Inanição (starvation)



Uma transação fica esperando por um período indefinido devido às políticas de espera por itens bloqueados for injusto

- Uma transação com maior prioridade toma a vez de uma transação que espera
- Pode ser resolvido com uma fila simples (FIFO – primeiro a chegar, primeiro a ser atendido)

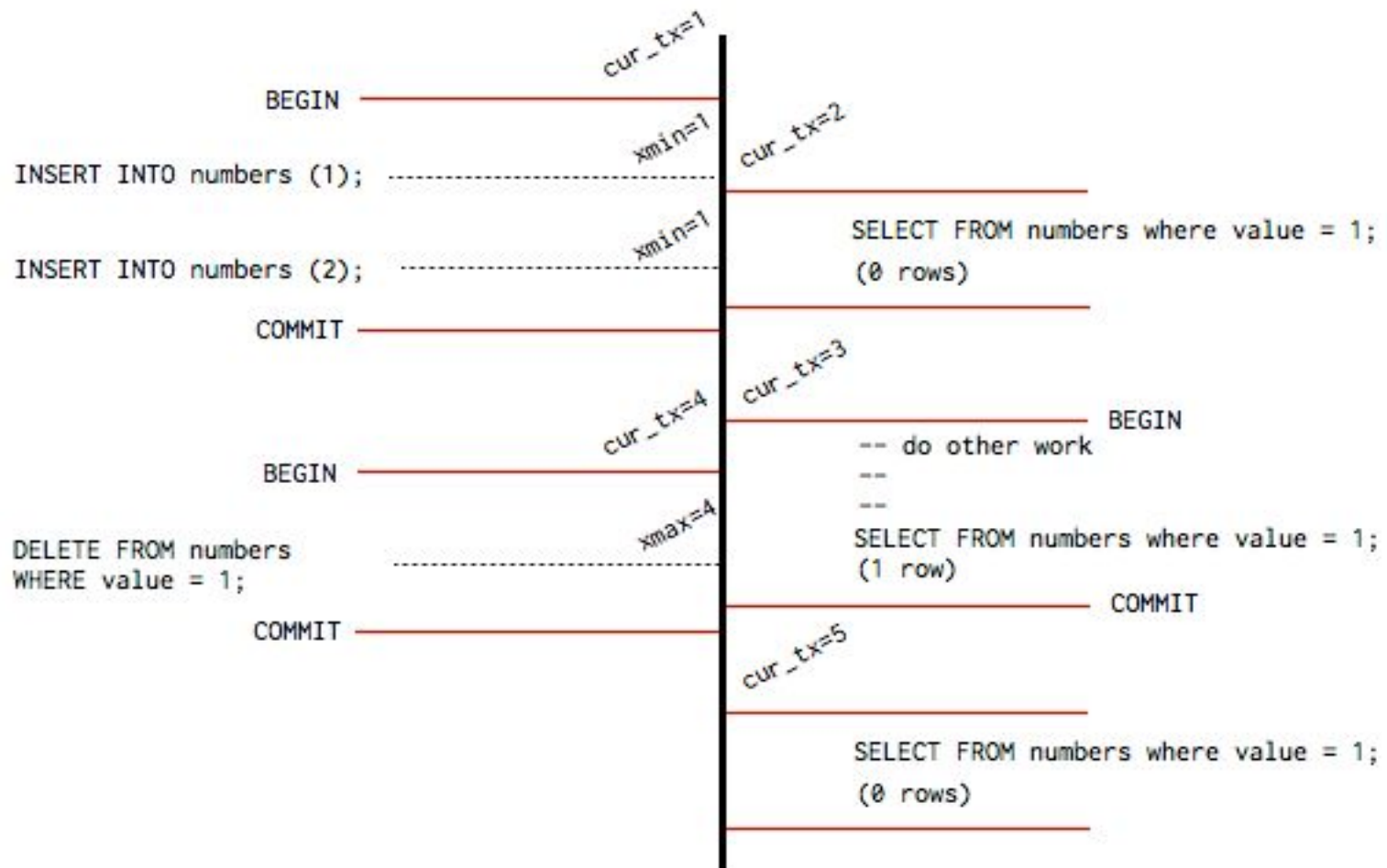
Controle de concorrência Postgres

- No Postgres, um Delete nao apaga o dado de fato

Multiversion Concurrency Control- MVCC

- Baseado no conceito que conflitos são infrequentes
- Deixar executar as transações concorrentemente
- Se ocorrer um conflito, uma das transações é abortada
- Cada transação enxerga sobre uma cópia dos dados e não “lê” alterações de outras transações não comitadas
- Dados são salvos em cópias
- Ao iniciar uma transação é mantido uma lista de todas as outras em progresso.

MVCC



MVCC

Variáveis:

- Xmin: armazena a transação que fez a última alteração
- Xmax: reporta se o registro está em processo de remoção

Exemplo 1

A) Crie a tabela teste (id integer, value char(500))

B) Adicione 10 linhas

C) Verifique o tamanho da tabela

```
SELECT pg_size_pretty( pg_total_relation_size('table name') )
```

D) Verifique a posição de cada registro com o comando

```
SELECT ctid,* from teste
```

E) Atualize todas as linhas para o id receber +1

```
Update teste set id=id+1;
```

F) Verifique o tamanho da tabela novamente e a posição de cada registro. **Descreva o que aconteceu.**

cada alteração nos dados são salvos através de cópias dos dados. Isso permite a criação de cópias das informações evitando a questão do dirty read.

Atividade 2

A) Abra uma transacao A:

Rode `SELECT txid_current()`

Rode `SELECT xmin, xmax, ctid, * FROM teste`

B) Apague uma tupla em outra transação B (outro terminal)

`SELECT txid_current()`

C) Rode `SELECT xmin, xmax, ctid, * FROM teste`

D) Rode novamente o `SELECT xmin, xmax, ctid, * FROM teste`. **Explique o que aconteceu?**

E) Faça o mesmo com o comando `update`, atividade A-D.. **Explique novamente o que aconteceu?**

G) Explique o motivo do Postgres não apagar um dado quando solicitado.

Atividade 3 - para entregar em dupla

A) Crie a tabela teste (id integer primary key, value char(500))

B) Adicione 10 linhas

C) Verifique o tamanho da tabela

D) Insira uma com id =10 em uma transação A

E) Insira tupla com id =10 em uma transação B.

Explique o que aconteceu já que os dados são operados em cópias diferentes?

Simulação

<https://github.com/amughrabi/cc.git>