Gestió i Processament de Transaccions: planificacions

Gestió i Administració de Bases de Dades.
Grau en Enginyeria Informàtica

Oriol Ramos Terrades

Carles Sánchez Ramos

Departament de Ciències de la Computació

Continguts

- Seqüenciable per conflictes
- Seqüenciable per vista.
- Graf de precedència.

Seqüencialitat: planificació

Objectiu: entendre quines planificacions concurrents són equivalents a execucions seqüencials

SEQÜENCIALITAT: Assegura consistència si la planificació triada és equivalent (té el mateix efecte) que una planificació seqüencial.

Problemes de concurrència:

- Pèrdua d'actualització
- Lectura bruta

Seqüencialitat: exemple d'execució seqüencial

t	T1	T2	X	Y	N	M
1	read_item(X)		90	50	3	2
2	X:=X-N		90	50	3	2
3	write_item(X)		**87**	50	3	2
4	read_item(Y)		87	50	3	2
5	Y:=Y+N		87	50	3	2
6	write_item(Y)		87	**53**	3	2
7	commit()		87	53	3	2
8		read_item(X)	87	53	3	2
9		X:=X+M	87	53	3	2
10		write_item(X)	**89**	53	3	2
11		commit()	89	53	3	2

Opció fàcil, poc òptima.

t	T1	T2	X	Y	N	M
1		read_item(X)	90	50	3	2
2		X:=X+M	90	50	3	2
3		write_item(X)	**92**	50	3	2
4		commit()	92	50	3	2
5	read_item(X)		92	50	3	2
6	X:=X-N		92	50	3	2
7	write_item(X)		**89**	50	3	2
8	read_item(Y)		87	50	3	2
9	Y:=Y+N		87	50	3	2
10	write_item(Y)		89	**53**	3	2
11	commit()		89	53	3	2

Prob. Concurrència: pèrdua d'actualització

Operació d'actualització no finalitzada.

T_1 T_2

90 → 87 → 89

T_2 T_1

90 → 92 → 89

t	T_1	T2	X	Y	N	M	$X(T_1)$	$X(T_2)$
1	read_item(X)		90	50	3	2	90	
2	X:=X-N		90	50	3	2	87	
3		read_item(X)	90	50	3	2	87	90
4		X:=X+M	90	50	3	2	87	92
5	write_item(X)		87	50	3	2	87	92
6	read_item(Y)		87	50	3	2	87	92
7		write_item(X)	92					92
8	Y:=Y+N							
9	write_item(Y)							

Prob. Concurrència: lectura bruta

Actualització temporal.

t	*T1*	*T2*
1	`read_item(X)`	
2	`X:=X-N`	
3	`write_item(X)`	
4		`read_item(X)`
5		`X:=X+M`
6		`write_item(X)`
7	`rollback()`	

Si T1 falla, cal recuperar el valor *X* anterior a l'actualització de *T1*, però *T2* ja ha llegit el valor *X* canviat per *T1*.

Seqüencialitat: planificació de transaccions

Planificació $S: I_1, I_2 \ldots I_m$ **:** Seqüència d'operacions de diferents transaccions $T_1 \ldots T_r$ amb 2 dues restriccions:

1. Operacions de la transacció T_i estan ordenades
2. Operacions de la transacció T_j poden interpolar-se amb les de T_i

Operacions d'una transacció:

- r (read) – $r_n(X)$: Llegir item X de transacció n
- w (write) – $w_n(Y)$: Escriure item Y de transacció n
- c (commit) – c_n: Confirmar transacció n
- a (rollback) – a_n: Avortar transacció n

Planificació de transaccions: exemple

S1: r1(X), r2(X), w1(X), r1(Y), w2(X), c2, w1(Y), c1;

PLANIFICACIÓ S1		
t	***T1***	***T2***
1	read_item(X)	
2	X:=X-N	
3		read_item(X)
4		X:=X+M
5	write_item(X)	
6	read_item(Y)	
7		write_item(X)
8		commit()
9	Y:=Y+N	
10	write_item(Y)	
11	commit()	

Planificació de transaccions: seqüencial

Si per totes les transaccions Ti de S, les operacions d'una transacció s'executen consecutivament sense que s'interpolin les operacions de l'altra transacció.

$A: r_1(X), w_1(X), r_1(Y), w_1(Y), c_1, r_2(X), w_2(X), c_2$

t	T1	T2
1	read_item(X)	
2	X:=X-N	
3	write_item(X)	
4	read_item(Y)	
5	Y:=Y+N	
6	write_item(Y)	
7	commit()	
8		read_item(X)
9		X:=X+M
10		write_item(X)
11		commit()

Planificació de transaccions: seqüencial

A, B planificacions seqüencials:

PLANIFICACIÓ A		
t	*T1*	*T2*
1	read_item(X)	
2	X:=X-N	
3	write_item(X)	
4	read_item(Y)	
5	Y:=Y+N	
6	write_item(Y)	
7	commit()	
8		read_item(X)
9		X:=X+M
10		write_item(X)
11		commit()

PLANIFICACIÓ B		
t	*T1*	*T2*
1		read_item(X)
2		X:=X+M
3		write_item(X)
4		commit()
5	read_item(X)	
6	X:=X-N	
7	write_item(X)	
8	read_item(Y)	
9	Y:=Y+N	
10	write_item(Y)	
11	commit()	

- Sols hi ha una transacció activa en cada instant. No importa l'ordre execució.
- **PROBLEMA:** Limiten la concurrència, baix aprofitament CPU en operacions E/S.

Planificacions seqüencials: exemple

PLANIFICACIÓ A						
t	***T1***	***T2***	***X***	***Y***	***N***	***M***
1	read_item(X)		90	50	3	2
2	X:=X-N		90	50	3	2
3	write_item(X)		**87**	50	3	2
4	read_item(Y)		87	50	3	2
5	Y:=Y+N		87	50	3	2
6	write_item(Y)		87	**53**	3	2
7	commit()		87	53	3	2
8		read_item(X)	87	53	3	2
9		X:=X+M	87	53	3	2
10		write_item(X)	**89**	53	3	2
11		commit()	89	53	3	2

PLANIFICACIÓ B						
t	***T1***	***T2***	***X***	***Y***	***N***	***M***
1		read_item(X)	90	50	3	2
2		X:=X+M	90	50	3	2
3		write_item(X)	**92**	50	3	2
4		commit()	92	50	3	2
5	read_item(X)		92	50	3	2
6	X:=X-N		92	50	3	2
7	write_item(X)		**89**	50	3	2
8	read_item(Y)		87	50	3	2
9	Y:=Y+N		87	50	3	2
10	write_item(Y)		89	**53**	3	2
11	commit()		89	53	3	2

Mateix resultat final.

Planificacions no seqüencials

C, D no seqüencials:

PLANIFICACIÓ C		
t	***T1***	***T2***
1	read_item(X)	
2	X:=X-N	
3		read_item(X)
4		X:=X+M
5	write_item(X)	
6	read_item(Y)	
7		write_item(X)
8		commit()
9	Y:=Y+N	
10	write_item(Y)	
11	commit()	

PLANIFICACIÓ D		
t	***T1***	***T2***
1	read_item(X)	
2	X:=X-N	
3	write_item(X)	
4		read_item(X)
5		X:=X+M
6		write_item(X)
7		commit()
8	read_item(Y)	
9	Y:=Y+N	
10	write_item(Y)	
11	commit()	

- S'interpolen operacions de diferents transaccions.
- Millor aprofitament CPU.
- **PROBLEMA:** Poden donar incorrectes per la concurrència.

Planificacions no seqüencials: no consistents

Exemple *C*:
X=90, Y=50, N=3, M=2:

PLANIFICACIÓ C								
t	***T1***	***T2***	***X***	***Y***	***N***	***M***	***X(T1)***	***X(T2)***
1	`read_item(X)`		*90*	*50*	*3*	*2*	*90*	
2	`X:=X-N`		*90*	*50*	*3*	*2*	***87***	
3		`read_item(X)`	*90*	*50*	*3*	*2*	*87*	*90*
4		`X:=X+M`	*90*	*50*	*3*	*2*	*87*	***92***
5	`write_item(X)`		***87***	*50*	*3*	*2*	*87*	*92*
6	`read_item(Y)`		*87*	*50*	*3*	*2*	*87*	*92*
7		`write_item(X)`	***92***	*50*	*3*	*2*	*87*	*92*
8		`commit()`	*92*	*50*	*3*	*2*	*87*	*92*
9	`Y:=Y+N`		*92*	*50*	*3*	*2*	*87*	*92*
10	`write_item(Y)`		*92*	***53***	*3*	*2*	*87*	*92*
11	`commit()`		*92*	*53*	*3*	*2*	*87*	*92*

C **no consistent:** *T2* llegeix valor *X* abans que *T1* l'hagi actualitzat. Resultat no coincideix amb planificacions seqüencials *A, B*.

Planificacions no seqüencials: consistents

Exemple D:

$X=90,\ Y=50,\ N=3,\ M=2$:

PLANIFICACIÓ D						
t	***T1***	***T2***	***X***	***Y***	***N***	***M***
1	`read_item(X)`		*90*	*50*	*3*	*2*
2	`X:=X-N`		*90*	*50*	*3*	*2*
3	`write_item(X)`		***87***	*50*	*3*	*2*
4		`read_item(X)`	*87*	*50*	*3*	*2*
5		`X:=X+M`	*87*	*50*	*3*	*2*
6		`write_item(X)`	*89*	*50*	*3*	*2*
7		`commit()`	*89*	*50*	*3*	*2*
8	`read_item(Y)`		*89*	*50*	*3*	*2*
9	`Y:=Y+N`		*89*	*50*	*3*	*2*
10	`write_item(Y)`		*89*	***53***	*3*	*2*
11	`commit()`		*89*	*53*	*3*	*2*

D **consistent:** Resultat final igual que planificacions seqüencials A i B.

Planificació de transaccions

- No totes les planificacions porten a un estat consistent.
- Donat un conjunt de transaccions, combinant operacions hi ha moltes planificacions possibles, unes porten a un estat consistent a la BD, altres no.
- El SGBD ha d'assegurar que la planificació que decideixi porti a un estat consistent.

Seqüencialitat i seqüenciable

- **Seqüencialitat (o serialització):** Donada una planificació P, buscar una planificació seqüencial equivalent.
- **Seqüenciable:** una planificació és seqüenciable si existeix una planificació seqüencial equivalent.
- Per entendre les planificacions que asseguren consistència, i les que no, s'ha d'analitzar dues operacions:
  - `read_item(Q)` - $r(Q)$
  - `write_item(Q)` - $w(Q)$.
- Diferents tipus d'equivalència entre planificacions:
  - **Seqüencialitat per conflictes**
  - **Seqüencialitat per vistes**

Seqüenciable per conflicte

- Suposem una planificació S que conté les instruccions I_i, I_j pertanyents a les transaccions T_i, T_j.
- Tenint en compte les operacions llegir ($r(Q)$) un element Q o escriure ($w(Q)$), es poden considerar quatre casos de possible conflicte:
  1. $I_i = r(Q)$, $I_j = r(Q)$: No hi ha conflicte.
  2. $I_i = r(Q)$, $I_j = w(Q)$: Si I_i s'executa abans que I_j el primer no llegeix el valor del segon. Conflicte.
  3. $I_i = w(Q)$, $I_j = r(Q)$: Conflicte.
  4. $I_i = w(Q)$, $I_j = w(Q)$: Conflicte.

Seqüenciable per conflicte

- Donada una planificació S

$$S: I_1, I_2 \ldots I_n$$

- Dues operacions I_i, I_j $(i <> j)$ d'una planificació entren en **conflicte** si es satisfan tres condicions:
  - Pertanyen a transaccions diferents.
  - Accedeixen al mateix element de dades X.
  - Almenys una de les dues operacions és $w(X)$.

Seqüenciable per conflicte

Donades, dues transaccions

- $T1: r_1(X), w_1(X)$
- $T2: r_2(X), w_2(X),$

les operacions en conflicte són:

- $r_1(X) – w_2(X)$
- $r_2(X) – w_1(X)$
- $w_1(X) – w_2(X)$

	r1(X)	*w1(X)*	*r2(X)*	*w2(X)*	*r1(Y)*
r1(X)		*No*	*No*	*Si*	*No*
w1(X)	*No*		*Si*	*Si*	*No*
r2(X)	*No*	*Si*		*No*	*No*
w2(X)	*Si*	*Si*	*No*		*No*
r1(Y)	*No*	*No*	*No*	*No*	

Seqüencialible per conflicte: exemple

PLANIFICACIÓ D		
t	*T1*	*T2*
1	read_item(X)	
2	X:=X-N	
3	write_item(X)	
4		read_item(X)
5		X:=X+M
6		write_item(X)
7		commit()
8	read_item(Y)	
9	Y:=Y+N	
10	write_item(Y)	
11	commit()	

→→

PLANIFICACIÓ D'		
t	*T1*	*T2*
1	read_item(X)	
2	X:=X-N	
3	write_item(X)	
4		read_item(X)
5		X:=X+M
6	read_item(Y)	
7		write_item(X)
8		commit()
9	Y:=Y+N	
10	write_item(Y)	
11	commit()	

- Operació $w_1(X)$ en conflicte amb $r_2(X)$.
- Operació $w_2(X)$ no en conflicte amb $r_1(Y)$ → **INTERCANVI**

Seqüenciable per conflicte

Donada una planificació P

$$P: I_1, I_2 \ldots I_i, I_j, \ldots I_n$$

Si dues operacions I_i, I_j (I_i *de* T_1, I_j *de* T_2) no estan en conflicte, es pot canviar l'ordre de les dues obtenint una nova planificació S equivalent a P.

$$S: I_1, I_2 \ldots I_j, I_i, \ldots I_n$$

Seqüenciable per conflicte

Seguint intercanvis $r_1(Y) \leftrightarrow r_2(X)$, $w_1(Y) \leftrightarrow w_2(X)$, *etc.*:

PLANIFICACIÓ D'		
t	*T1*	*T2*
1	read_item(X)	
2	X:=X-N	
3	write_item(X)	
4		read_item(X)
5		X:=X+M
6	read_item(Y)	
7		write_item(X)
8		commit()
9	Y:=Y+N	
10	write_item(Y)	
11	commit()	

→→

PLANIFICACIÓ A		
T	*T1*	*T2*
1	read_item(X)	
2	X:=X-N	
3	write_item(X)	
4	read_item(Y)	
5	Y:=Y+N	
6	write_item(Y)	
7	commit()	
8		read_item(X)
9		X:=X+M
10		write_item(X)
11		commit()

- Planificació D s'ha pogut transformar en A amb intercanvis d'operacions no conflictives: **equivalència per conflictes.**

Seqüenciable per conflicte

Dues planificacions P, S **són equivalents per conflicte** si l'ordre de qualsevol parell I_i, I_j d'operacions en conflicte és el mateix en les dues planificacions.

Si dues operacions conflictives s'apliquen en ordre diferent en dues planificacions → **planificacions no equivalents per conflicte**.

Exemples:

- P_1: $r_1(X)$, $w_2(X)$;
- S_1: $w_2(X)$, $r_1(X)$; $P1, S1$ no equivalents per conflicte
- P_2: $w_1(X)$, $r_1(Y)$, $w_2(X)$;
- S_2: $w_1(X)$, $w_2(X)$, $r_1(Y)$; $P2, S2$ equivalents per conflicte

Seqüenciable per conflicte

Una planificació P és seqüenciable per conflicte si és equivalent per conflicte a alguna planificació seqüencial S.

Cal reordenar les operacions no conflictives de P per trobar la planificació equivalent seqüencial S.

Seqüenciable per conflicte: exemple I

PLANIFICACIÓ D (seqüenciable per conflicte)		
t	***T1***	***T2***
1	read_item(X)	
2	X:=X-N	
3	write_item(X)	
4		read_item(X)
5		X:=X+M
6		write_item(X)
7		commit()
8	read_item(Y)	
9	Y:=Y+N	
10	write_item(Y)	
11	commit()	

PLANIFICACIÓ A (SEQÜENCIAL)		
t	***T1***	***T2***
1	read_item(X)	
2	X:=X-N	
3	write_item(X)	
4	read_item(Y)	
5	Y:=Y+N	
6	write_item(Y)	
7	commit()	
8		read_item(X)
9		X:=X+M
10		write_item(X)
11		commit()

Planificació D equivalent a A:

- $r_2(X)$ llegeix X escrit per T_1 en ambdues planificacions.
- $r_1(Y), w_1(Y)$ no en conflicte amb $r_2(X), w_2(X)$→avançar $r_1(Y), w_1(Y)$ en D és A.

Seqüenciable per conflicte: exemple II

PLANIFICACIÓ C			
t	***T1***	***T2***	***T***
1	read_item(X)		r1(X)
2	X:=X-N		
3		read_item(X)	r2(X)
4		X:=X+M	
5	write_item(X)		w1(X)
6	read_item(Y)		r1(Y)
7		write_item(X)	w2(X)
8		commit()	
9	Y:=Y+N		
10	write_item(Y)		w1(Y)
11	commit()		

PLANIFICACIÓ A (SEQÜENCIAL)			
t	***T1***	***T2***	***T***
1	read_item(X)		r1(X)
2	X:=X-N		
3	write_item(X)		w1(X)
4	read_item(Y)		r1(Y)
5	Y:=Y+N		
6	write_item(Y)		w1(Y)
7	commit()		
8		read_item(X)	r2(X)
9		X:=X+M	
10		write_item(X)	w2(X)
11		commit()	

Planificació C no equivalent a A:

- $r_2(X)$, $w_1(X)$ en conflicte: no podem moure $r_2(X)$ avall sense canviar resultat.

Planificació de transaccions

- No totes les planificacions porten a un estat consistent.
- Donat un conjunt de transaccions, combinant operacions hi ha moltes planificacions possibles, unes porten a un estat consistent a la BD, altres no.
- El SGBD ha d'assegurar que la planificació que decideixi porti a un estat consistent.
- **SEQÜENCIALITAT:** Assegura consistència si la planificació triada és equivalent (té el mateix efecte) que una planificació seqüencial.

Seqüenciable per vista

Dues planificacions P i S són **equivalents en vista** si compleixen les 3 condicions següents:

1. El mateix conjunt de transaccions i operacions estan incloses en P i S.

2. Per a qualsevol operació $r_i(Q)$ en T_i de P, si el valor Q llegit ha estat escrit per una operació $w_j(Q)$ de T_j, la mateixa condició s'ha de mantenir per $r_i(Q)$ en T_i de S.

3. Si l'operació $w_k(Y)$ de T_k és la darrera operació en escriure Y en P, llavors $w_k(Y)$ de T_k també ha de ser per S.

Seqüenciable per vista

Equivalents en vista: si cada operació de lectura llegeix el mateix valor d'escriptura (tant a l'inici com al mig) en ambdues planificacions, el resultat final igual.

Una planificació S és **seqüenciable per vista** si la planificació és equivalent en vista una planificació seqüencial.

Seqüenciable per vista

Seqüenciable per vista i per conflicte són similars si es compleix la condició de **suposició d'escriptura restringida** en totes les transaccions d'una planificació:

- Tota operació $w_i(X)$ en T_i ve precedida d'una operació $r_i(X)$ en T_i.
- Valor escrit per $w_i(X)$ en T_i sols depèn del valor X llegit a $r_i(X)$: Nou valor $X'=f(X)$ on X s'ha obtingut d'un $r_i(X)$.

Seqüenciable per vista: exemple

PLANIFICACIÓ A

t	T1	T2	X	Y	N	M
1	read_item(X)		90	50	3	2
2	X:=X-N		90	50	3	2
3	write_item(X)		87	50	3	2
4	read_item(Y)		87	50	3	2
5	Y:=Y+N		87	50	3	2
6	write_item(Y)		87	53	3	2
7	commit()		87	53	3	2
8		read_item(X)	87	53	3	2
9		X:=X+M	87	53	3	2
10		write_item(X)	89	53	3	2
11		commit()	89	53	3	2

PLANIFICACIÓ B

t	T1	T2	X	Y	N	M
1		read_item(X)	90	50	3	2
2		X:=X+M	90	50	3	2
3		write_item(X)	92	50	3	2
4		commit()	92	50	3	2
5	read_item(X)		92	50	3	2
6	X:=X-N		92	50	3	2
7	write_item(X)		89	50	3	2
8	read_item(Y)		87	50	3	2
9	Y:=Y+N		87	50	3	2
10	write_item(Y)		89	53	3	2
11	commit()		89	53	3	2

Planificacions A i B no equivalents per vista: $r_1(X)$ no llegeix els mateixos valors en A i B.

Seqüenciable per vista: exemple

PLANIFICACIÓ G		
T3	***T4***	***T5***
`read_item(Q)`		
`Q:=Q+25`		
	`Q=25`	
	`write_item(Q)`	
`write_item(Q)`		
		`Q=50`
		`write_item(Q)`

PLANIFICACIÓ H (seqüencial)		
T3	***T4***	***T5***
`read_item(Q)`		
`Q:=Q+25`		
`write_item(Q)`		
	`Q=25`	
	`write_item(Q)`	
		`Q=50`
		`write_item(Q)`

G i H **equivalents per vista**: $r_3(Q)$ llegeix el valor inicial de Q i T_5 escriu el valor final de Q en G i H.

G **seqüenciable per vista** al ser equivalent per vista a la planificació seqüencial H.

Seqüenciable per vista

Seqüenciable per vista menys restrictiva que seqüenciable per conflicte**.**

En **seqüenciable per vista,** es poden fer **escriptures cegues,** (el valor escrit $w_i(Q)$ en Ti pot ser independent del valor que tingui, no depèn de $r_i(Q)$).

Exemple planificació G: $r_3(Q)$, $w_4(Q)$, $w_3(Q)$, $w_5(Q)$, c_3, c_4, c_5;

- $w_4(Q)$, $w_5(Q)$ són escriptures cegues.
- G **seqüenciable per vista:** equivalent per vista a planificació seqüencial H: T_3, T_4, T_5 → H: $r_3(Q)$, $w_3(Q)$, c_3, $w_4(Q)$, c_4, $w_5(Q)$, c_5;
- G **no seqüenciable per conflicte:** no és equivalent per conflicte a cap planificació seqüencial. Si s'intercanvia canvia qualsevol $w_k(Q)$ s'altera el resultat.

Seqüenciable per vista: resum

- Una planificació seqüenciable per vista ~~→~~ seqüenciable per conflicte.
- Tota planificació seqüenciable per conflicte → seqüenciable per vista.

Seqüencialitat: comprovació

Donada una planificació S, construir un **graf de precedència o de seqüencialitat**, definit com $G=(N,E)$ on:

- Conjunt de nodes $N=\{T_1...T_n\}$ que són transaccions d'una planificació.
- Conjunt d'arcs dirigits $E=\{e_1...e_n\}$, un arc $T_j \rightarrow T_k$ quan una operació de T_j apareix abans que alguna operació de conflicte de T_k en la planificació.

Per a detectar seqüencialitat de la planificació, cal verificar cicles del graf.

Seqüencialitat: comprovació

Algorisme de comprovació de seqüencialitat d'una planificació $S=\{T_1 \ldots T_n\}$:

1. Per a cada transacció T_j en S es crea un node T_j en el graf.
2. Per a cada cas on T_j executi $r_j(X)$ després que T_i executi un $w_i(X)$, crear un arc dirigit $T_i \rightarrow T_j$
3. Per a cada cas on T_j executi $w_j(X)$ després que T_i executi un $r_i(X)$, crear un arc dirigit $T_i \rightarrow T_j$

Ti	*Tj*
`write_item(X)`	
	`read_item(X)`

Ti	*Tj*
`read_item(X)`	
	`write_item(X)`

Seqüencialitat: comprovació

Algorisme comprovació de seqüencialitat d'una planificació $S=\{T_1 \ldots T_n\}$:

4. Per a cada cas on T_j executi $w_j(X)$ després que Ti executi un $w_i(X)$, crear un arc dirigit $T_i \rightarrow T_j$

T_i	T_j
`write_item(X)`	
	`write_item(X)`

4. La planificació S és seqüenciable ⟺ **graf de precedència no té cicles.**

Seqüencialitat: comprovació

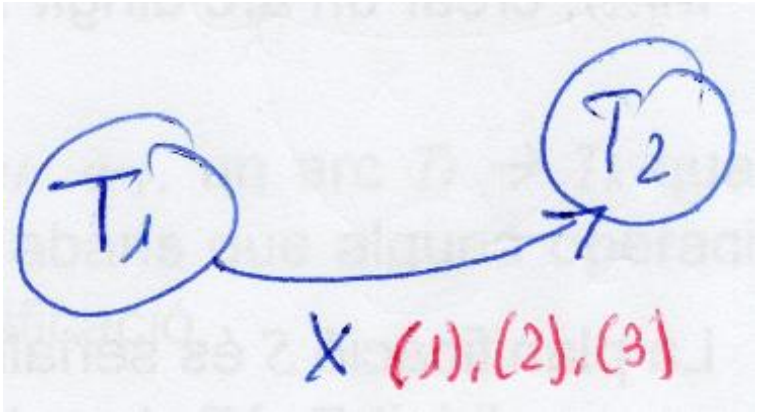
Un arc $T_i \rightarrow T_j$ significa que T_i ha d'executar-se abans que T_j en qualsevol planificació equivalent a S.

Si no hi ha cap cicle a l'arc: es pot trobar una planificació seqüencial equivalent a S, ordenant d'aquesta forma les transaccions.

Si existeix un cicle, no existeix una única precedència, per tant no es pot trobar una planificació seqüencial equivalent → **Planificació no seqüenciable**

Comprovació de seqüencialitat: exemple I

PLANIFICACIÓ A (SEQÜENCIAL)				
t	***T1***	***Arcs T1***	***T2***	***Arcs T2***
1	`read_item(X)`	`(2)`		
2	`X:=X-N`			
3	`write_item(X)`	`(3),(1)`		
4	`read_item(Y)`			
5	`Y:=Y+N`			
6	`write_item(Y)`			
7	`commit()`			
8			`read_item(X)`	`(1)`
9			`X:=X+M`	
10			`write_item(X)`	`(3),(2)`
11			`commit()`	

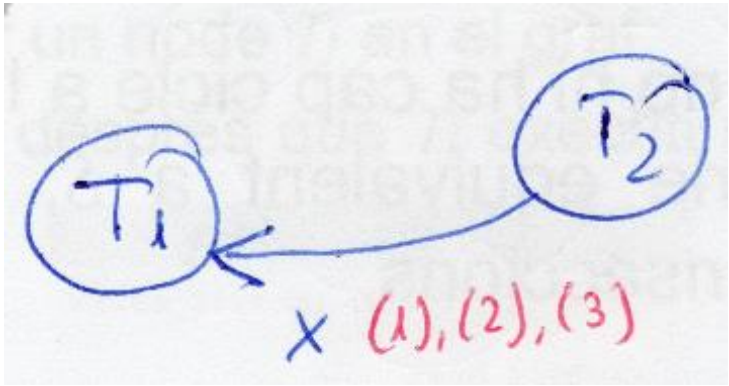


Planificació seqüenciable

Comprovació de seqüencialitat: exemple II

PLANIFICACIÓ B (SEQÜENCIAL)				
t	***T1***	***Arcs T1***	***T2***	***Arcs T2***
1			`read_item(X)`	`(2)`
2			`X:=X+M`	
3			`write_item(X)`	`(3),(1)`
4			`commit()`	
5	`read_item(X)`	`(1)`		
6	`X:=X-N`			
7	`write_item(X)`	`(3),(2)`		
8	`read_item(Y)`			
9	`Y:=Y+N`			
10	`write_item(Y)`			
11	`commit()`			

Graf

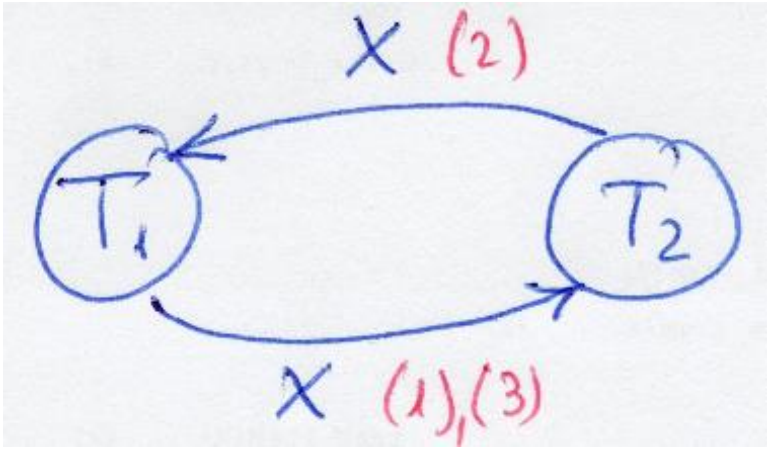


Planificació seqüenciable

Comprovació de seqüencialitat: exemple III

PLANIFICACIÓ C				
t	***T1***	***Arcs T1***	***T2***	***Arcs T2***
1	`read_item(X)`	`(1)`		
2	`X:=X-N`			
3			`read_item(X)`	`(2)`
4			`X:=X+M`	
5	`write_item(X)`	`(2),(3)`		
6	`read_item(Y)`			
7			`write_item(X)`	`(1),(3)`
8			`commit()`	
9	`Y:=Y+N`			
10	`write_item(Y)`			
11	`commit()`			

Graf

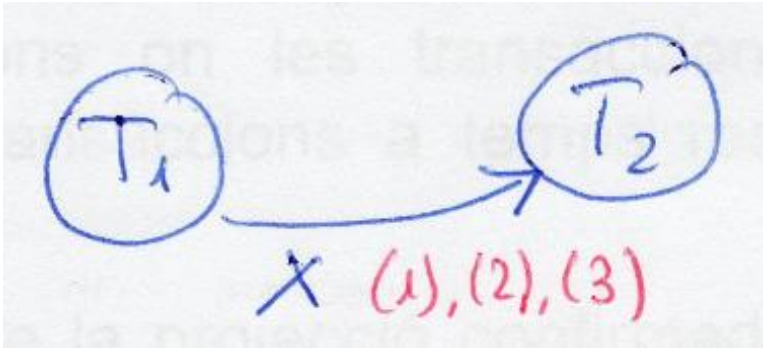


Planificació NO seqüenciable

Comprovació de seqüencialitat: exemple IV

PLANIFICACIÓ D				
T	***T1***	***Arcs T1***	***T2***	***Arcs T2***
1	read_item(X)	(2)		
2	X:=X-N			
3	write_item(X)	(3),(1)		
4			read_item(X)	(1)
5			X:=X+M	
6			write_item(X)	(3),(2)
7			commit()	
8	read_item(Y)			
9	Y:=Y+N			
10	write_item(Y)			
11	commit()			

Graf

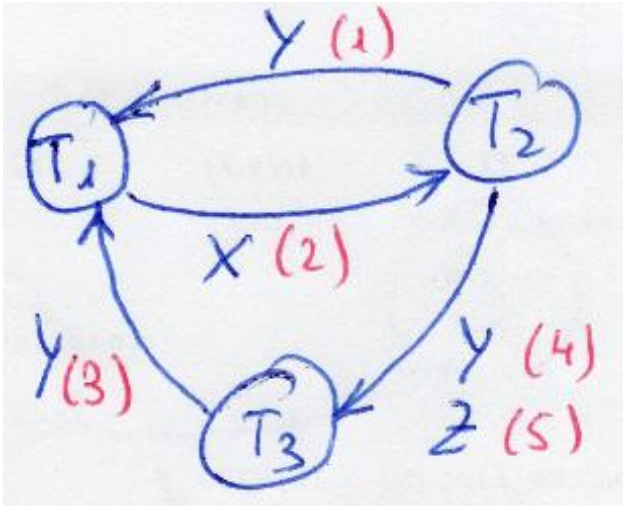


Planificació seqüenciable, equivalent a planificació A

Comprovació de seqüencialitat: exemple V

PLANIFICACIÓ E						
t	***T1***	***Arcs T1***	***T2***	***Arcs T2***	***T3***	***Arcs T3***
1			read_item(Z)	(5)		
2			read_item(Y)			
3			write_item(Y)	(4),(1)		
4					read_item(Y)	(4)
5					read_item(Z)	
6	read_item(X)					
7	write_item(X)	(2)				
8					write_item(Y)	(3)
9					write_item(Z)	(5)
10					commit()	
11			read_item(X)	(2)		
12	read_item(Y)	(3)				
13	write_item(Y)	(1)				
14	commit()					
15			write_item(X)			
16			commit()			

Graf



Planificació NO seqüenciable

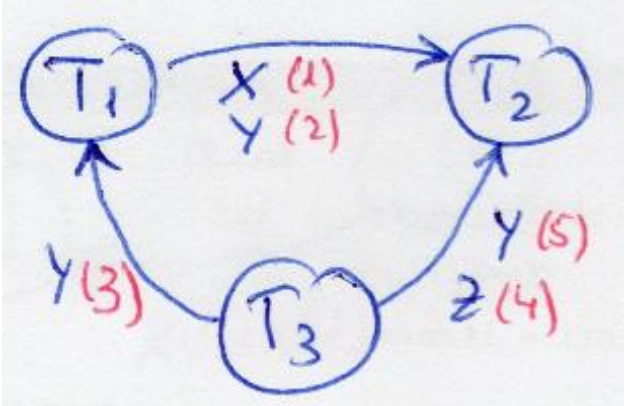
Cicles:

- *T1* → *T2* → *T1*
- *T1* → *T2* → *T3* → *T1*

Comprovació de seqüencialitat: exemple VI

PLANIFICACIÓ F						
t	***T1***	***Arcs T1***	***T2***	***Arcs T2***	***T3***	***Arcs T3***
1					read_item(Y)	
2					read_item(Z)	
3	read_item(X)					
4	write_item(X)	(1)				
5					write_item(Y)	(5), (3)
6					write_item(Z)	(4)
7			read_item(Z)	(4)		
8	read_item(Y)	(3)				
9	write_item(Y)	(2)				
10			read_item(Y)	(5),(2)		
11			write_item(Y)			
12			read_item(X)	(1)		
13			write_item(X)			

Graf



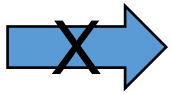
Planificació seqüenciable

Planificació seqüencial equivalent:

- $T3 \rightarrow T1 \rightarrow T2$

Usos de la seqüencialitat

Planificació seqüenciable per conflicte és com dir que S és correcta, aprofitant concurrència operacions.

Planificació seqüenciable S  S seqüencial.

En la pràctica, molt difícil trobar la seqüencialitat d'una planificació. Criteris de planificació:

- Càrrega sistema
- Temps esperat transacció
- Prioritat transacció

Difícil preveure com s'interpolen operacions d'una planificació per a garantir seqüencialitat.

Usos de la seqüencialitat

Comprovació a posteriori (transaccions s'executen en un ordre i si no es poden seqüencialitzar, es cancel·len), és poc pràctic.

Metodologia SGBD comercials: Dissenyar **protocols de concurrència** que aplicats a transaccions garanteixen la seqüencialitat de totes les planificacions on les transaccions participen. Difícil definir seqüència estable transaccions a temps real (entrada contínua transaccions).

Usos de la seqüencialitat

Protocols de concurrència que garanteixen seqüencialitat:

- Bloqueig a dues fases (utilitzada majoritàriament en SGBD)
- Ordenació per marca de temps
- Protocols multi-versió
- Protocols optimistes (certificació o validació)

En resum...

Planificacions en base a seqüencialitat (per conflicte i per vista) i recuperabilitat garanteixen consistència BD.

Usos de la seqüencialitat