Basi di Dati
Modulo Tecnologie

Esecuzione concorrente di transazioni (I parte)

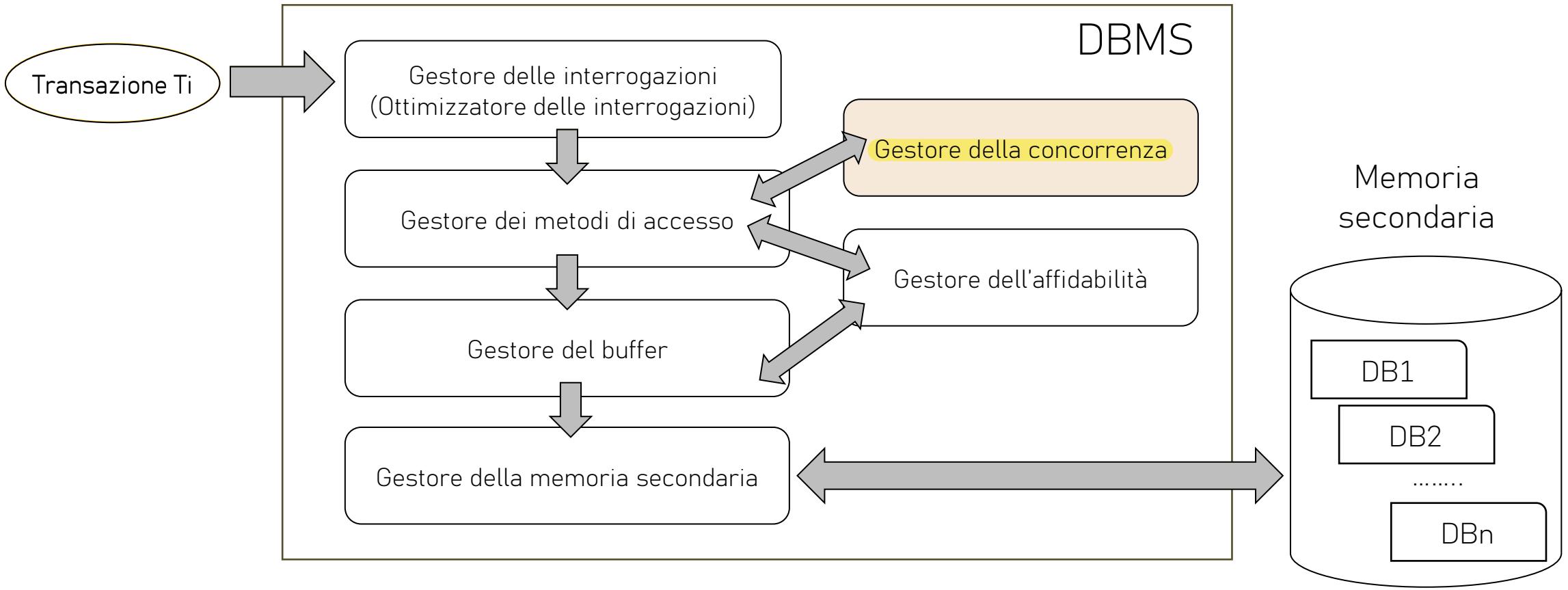
Dr. Sara Migliorini

Osservazione

- L'unità di misura utilizzata per caratterizzare il carico applicativo di un DBMS è il numero di transazioni al secondo (TPS = transaction per second).
- Per gestire con prestazione accettabili il carico di lavoro tipico delle applicazioni gestionali (100 o 1000 tps) un DBMS deve eseguire le transazioni in modo concorrente.
- L'esecuzione concorrente di transazioni senza controllo può generare anomalie o problemi di correttezza (come vedremo in alcuni esempi).
- È quindi necessario introdurre dei meccanismi di controllo nell'esecuzione delle transazioni per evitare tali anomalie.

Architettura di riferimento di un DBMS

Riceve richieste di accesso ai dati e decide se autorizzarle o meno, eventualmente riordinandole (Scheduler)



Anomalie di esecuzione concorrente

- Anomalie tipiche
  - Perdita di aggiornamento
    - Gli effetti di una transazione concorrente sono persi.
  - Lettura inconsistente
    - Accessi successivi ad uno stesso dato all'interno di una transazione ritornano valori diversi.
  - Lettura sporca
    - Viene letto un dato che rappresenta uno stato intermedio nell'evoluzione di una transazione
  - Aggiornamento fantasma
    - Osservazione di uno stato intermedio che non soddisfa i vincoli di integrità.
  - Inserimento fantasma
    - Valutazioni diverse di valori aggregati a causa di inserimenti intermedi.

Anomalie di esecuzione concorrente

- Notazione:
  - Indichiamo con t_i una transazione
  - $r_i(x)$: è una operazione di lettura eseguita dalla transazione t_i sulla risorsa x
  - $w_i(x)$: è una operazione di scrittura eseguita dalla transazione t_i sulla risorsa x

Anomalia: PERDITA DI AGGIORNAMENTO

- Gli effetti di una transazione concorrente sono persi.
- Consideriamo le due seguenti transazioni, descritte con la notazione precedentemente illustrata:
  - t_1: bot $r_1(x)$; x=x+1; $w_1(x)$; commit; eot
  - t_2: bot $r_2(x)$; x=x+1; $w_2(x)$; commit; eot
- Vogliamo simulare l'esecuzione concorrente di t_1 e t_2 in modo che si generi l'anomalia detta perdita di aggiornamento.

Anomalia: PERDITA DI AGGIORNAMENTO

Operazioni eseguite da t1	Segmento di memoria centrale di t1	Memoria secondaria [buffer]	Segmento di memoria centrale di t2	Operazioni eseguite da t2
Stato iniziale		x = 2	Stato iniziale	
bot		2		
$r_1(x)$	2 ←	[2]		
x=x+1	3	[2]		bot
	3	[2] →	2	$r_2(x)$
	3	[2]	3	x=x+1
	3	[3] ←	3	$w_2(x)$
	3		3	commit
	3			eot
$w_1(x)$	3 →	[3]		
commit	3	3		
eot		3		

Il valore finale di x è 3, mentre una qualsiasi esecuzione seriale avrebbe prodotto per x il valore 4!

Anomalia: LETTURA INCONSISTENTE

- Accessi successivi ad uno stesso dato all'interno di una transazione ritornano valori diversi.

- Consideriamo le due seguenti transazioni, descritte con la notazione precedentemente illustrata:
  - t_1: bot $r_1(x)$; $r_1'(x)$; commit; eot
  - t_2: bot $r_2(x)$; x=x+1; $w_2(x)$; commit; eot

- Vogliamo simulare l'esecuzione concorrente di t1 e t2 in modo che si generi l'anomalia detta lettura inconsistente.

Anomalia: LETTURA INCONSISTENTE

Operazioni eseguite da t1	Segmento di memoria centrale di t1	Memoria secondaria [buffer]	Segmento di memoria centrale di t2	Operazioni eseguite da t2
Stato iniziale		x = 2	Stato iniziale	
bot		2		
$r_1(x)$	2 ←	[2]		
	2	[2]		bot
	2	[2] →	2	$r_2(x)$
	2	[2]	3	x=x+1
	2	[3]	3	$w_2(x)$
	2	3	3	commit
	2	3		eot
$r_1'(x)$	3 →	[3]		
commit	3	3		
eot		3		

Il valore di x è diventato 3, ma la transazione t_1 non ha modificato x!

Anomalia: LETTURA SPORCA

- Viene letto un dato che rappresenta uno stato intermedio nell'evoluzione di una transazione

- Consideriamo le due seguenti transazioni, descritte con la notazione precedentemente illustrata:
  - t_1: bot $r_1(x)$; commit; eot
  - t_2: bot $r_2(x)$; x=x+1; $w_2(x)$; … abort;

- Vogliamo simulare l'esecuzione concorrente di t_1 e t_2 in modo che si generi l'anomalia detta lettura sporca.

Anomalia: LETTURA SPORCA

Operazioni eseguite da T1	Segmento di memoria centrale di T1	Memoria secondaria [buffer]	Segmento di memoria centrale di T2	Operazioni eseguite da T2
Stato iniziale		x = 2	Stato iniziale	
bot		2		
		2		bot
		[2]	2	$r_2(x)$
		[2]	3	x=x+1
		[3]	3	$w_2(x)$
$r_1(x)$	3	[3]	3	abort
commit	3	2		
eot		2		

Il valore di x letto da t_1 è 3, ma la transazione t_2 non ha ancora eseguito il commit!

Anomalia: AGGIORNAMENTO FANTASMA

- Osservazione di uno stato intermedio che non soddisfa i vincoli di integrità.

- Consideriamo le due seguenti transazioni, descritte con la notazione precedentemente illustrata:

  - Risorse: x, y, z
  - Vincoli: x+y+z = 100

  - t_1: bot $r_1(y)$; $r_1(x)$; $r_1(z)$; s = x+y+z; eot
  - t_2: bot $r_2(y)$; y = y + 10; $r_2(z)$; z = z – 10; $w_2(y)$; $w_2(z)$; commit; eot

- Vogliamo simulare l'esecuzione concorrente di t_1 e t_2 in modo che si generi l'anomalia detta aggiornamento fantasma.

Anomalia: AGGIORNAMENTO FANTASMA

Operazioni eseguite da T1	Segmento di memoria centrale di T1	Memoria secondaria [buffer]	Segmento di memoria centrale di T2	Operazioni eseguite da T2
Stato iniziale		(x,y,z)=20,20,60	Stato iniziale	
bot		20,20,60		
r1 (y)	-,20,-	20,[20],60		
	-,20,-	20,[20],60		bot
	-,20,-	20,[20],60	-,20,-	r2(y)
	-,20,-	20,[20],60	-,30,-	y = y + 10
	-,20,-	20,[20],[60]	-,30,60	r2(z)
	-,20,-	20,[20],[60]	-,30,50	z = z - 10
	-,20,-	20,[30],[60]	-,30,50	w2(y)
	-,20,-	20,[30],[50]	-,30,50	w2(z)
	-,20,-	20,30,50	-,30,50	commit
	-,20,-	20,30,50		eot
r1 (x)	20,20,-	[20],30,50		
r1 (z)	20,20,50	[20],30,[50]		
s=x-y-z	20,20,50	[20],30,[50]		
eot		20,30,50		

S = 90 → VIOLAZIONE VINCOLO

Anomalia: INSERIMENTO FANTASMA

- Valutazioni diverse di valori aggregati a causa di inserimenti intermedi.
- Transazione che valuta un valore aggregato relativo a tutti gli elementi che soddisfano un predicato di selezione (es. voto medio degli studenti del terzo anno).
- Se il valore aggregato viene valutato due volte e tra la prima e la seconda valuazione viene inserito un nuovo studente, i due valori medi letti potrebbero essere diversi.
- Questa anomalia non si può risolvere facendo riferimento solo ai dati già presenti nella base di dati: è necessario tener presente che potrebbero esserci delle nuove tuple che compaiono «improvvisamente».