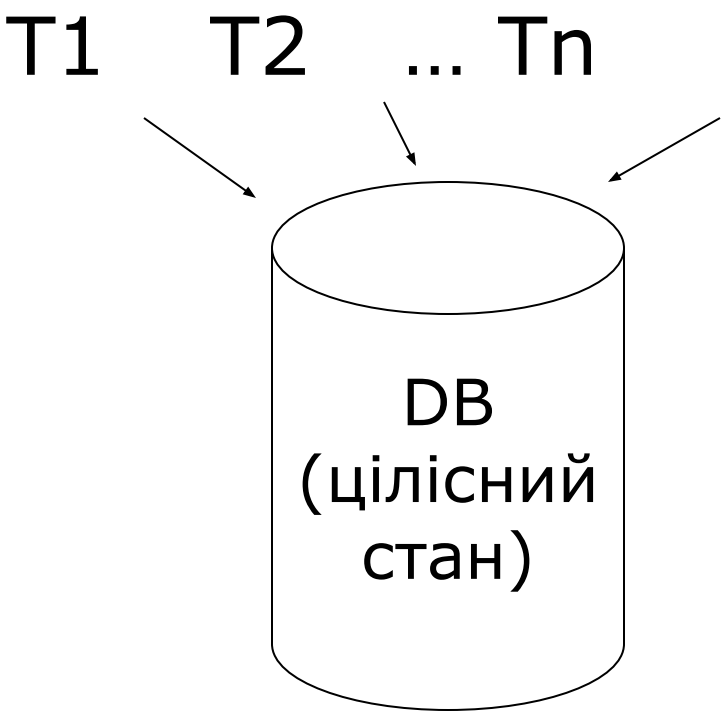
Керування транзакціями

Паралелізм

Три проблеми паралелізму

- При обробці транзакцій в загальному випадку можливі три типи ситуацій, при яких паралельне виконання транзакцій, кожна з яких сама по собі є коректною, через взаємні завади здатні привести до невірного результату.


T1
T2
… Tn
DB
(цілісний
стан)

Приклад:

T1:	Read(A)	T2:	Read(A)
	A ← A+100		A ← A×2
	Write(A)		Write(A)
	Read(B)		Read(B)
	B ← B+100		B ← B×2
	Write(B)		Write(B)

Constraint: A=B

Графік A - послідовний

T1	T2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
Read(B); B ← B+100;			
Write(B);			125
	Read(A);A ← A×2;		
	Write(A);	250	
	Read(B);B ← B×2;		
	Write(B);		250
		250	250

Графік B - послідовний

T1	T2	A	B
		25	25
	Read(A);A ← A×2;		
	Write(A);	50	
	Read(B);B ← B×2;		
	Write(B);		50
Read(A); A ← A+100			
Write(A);		150	
Read(B); B ← B+100;			
Write(B);			150
		150	150

Графік С – умовно-послідовний

T1	T2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
	Read(A);A ← A×2;		
	Write(A);	250	
Read(B); B ← B+100;			
Write(B);			125
	Read(B);B ← B×2;		
	Write(B);		250
		250	250

Графік D – не умовно-послідовний

T1	T2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
	Read(A);A ← A×2;		
	Write(A);	250	
	Read(B);B ← B×2;		
	Write(B);		50
Read(B); B ← B+100;			
Write(B);			150
		250	150

Такий як Графік D,
тільки з новою T2’

Графік E - умовно-послідовний

T1	T2’
Read(A); A ← A+100	
Write(A);	
	Read(A);A ← A×1;
	Write(A);
	Read(B);B ← B×1;
	Write(B);
Read(B); B ← B+100;	
Write(B);	

A	B
25	25
125	
125	
	25
	125
125	125

Проблеми паралелізму

- Неправильний кінцевий результат виникає через безконтрольне чергування операцій двох правильний транзакцій.

Проблема втрати результатів оновлення

T1	T2	A	B
		25	25
Read(A);		100	
	Read(A);		100
Write(A);		250	
Commit;			
	Write(A);		125
	Commit;		
		125	125

Проблема залежності від незафіксованих результатів

T1	T2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
	Read(A);A ← A×2;		250
Rollback		25	
	Write(A);		250
	Commit;		
			250
		250	250

Проблема неузгодженої обробки

A	B	bal_x	bal_y	Bal_z	Sum
	Start transaction	100	50	25	
Start transaction	Sum = 0	100	50	25	0
Read(bal_x)	Read(bal_z)	100	50	25	0
bal_x=bal_x-10	Sum = sum +bal_z	100	50	25	100
Write(bal_x)	Read(bal_z)	90	50	25	100
Read(bal_z)	Sum = sum +bal_z	90	50	25	150
bal_z=bal_z+10		90	50	25	150
Write(bal_z)		90	50	35	150
Commit	Read(bal_z)	90	50	35	150
	Sum = Sum + bal_z	90	50	35	185
	Commit	90	50	35	185

Конфлікти

- Конфлікти типу RR
  - операції читання не можуть порушувати роботу одна одної
- Конфлікти типу WR
  - може виникнути проблема неузгодженої обробки
- Конфлікти типу RW
  - може виникнути проблема незафіксованих оновлень
- Конфлікти типу WW
  - може виникнути проблема втраченого оновлення

Блокування

- У випадку, коли при виконанні деякої транзакції необхідно мати гарантії, що певний об’єкт бази даних не буде неочікувано змінений без відома транзакції, потрібний об’єкт блокується.

Механізм блокувань

- В системі підтримуються блокування двох типів: виняткові блокування (блокування X - exclusive) і колективні блокування (блокування S - shared).
- Якщо транзакція A володіє винятковим блокуванням (X), то запит від деякої іншої транзакції B на отримання блокування кортежу t будь-якого типу не може бути негайно задоволений.
- Якщо транзакція A володіє колективним блокуванням (S) кортежу t, то виконуються наступні умови:
  - запит деякої іншої транзакції B на отримання блокування X кортежу t не може бути негайно задоволений;
  - запит деякої іншої транзакції B на отримання блокування S кортежу t може і повинен бути негайно задоволений (це означає, що з цього часу транзакція B також буде володіти блокуванням S кортежу).

Матриця сумісності

	X	S	-
X	F	F	T
S	F	T	T
-	T	T	

Проблема втрати результатів оновлення

T1	T2
Read(A);	lock S
	Read(A); lock S
Write(A);	lock X
WAIT	
WAIT	Write(A); lock X
WAIT	WAIT
WAIT	WAIT
WAIT	WAIT

A	B
25	25
100	
	100

Проблема залежності від незафіксованих результатів

T1	T2	A	B
		25	25
Read(A); lock S			
A ← A+100		125	
Write(A);lock X			
	Read(A);lock S		
	WAIT	25	
	WAIT		25
Rollback ;(unlock A) A ← A×2;			50
	Write(A); lock X		50
	Commit;(unlock A)	50	50

A	B	bal_x	bal_y	Bal_z	Sum
	Start transaction	100	50	25	
Start transaction	Sum = 0	100	50	25	0
Read(bal_x) lock S	Read(bal_x) lock S	100	50	25	0
bal_x=bal_x-10	Sum = sum +bal_x	100	50	25	100
Write(bal_x) lock X	Read(bal_y) lock S	100	50	25	100
WAIT		100	50	25	100
WAIT	Sum = sum +bal_y	100	50	25	150
WAIT		100	50	25	150
WAIT		100	50	25	150
WAIT	Read(bal_z) lock S	100	50	25	150
WAIT	Sum = Sum + bal_z	100	50	25	175
WAIT	Commit (unlock ALL)	100	50	25	175
Write(bal_x)		90	50	25	175
Read(bal_z) lock S		90	50	25	175
bal_z=bal_z+10		90	50	25	175
Write(bal_z) lock X		90	50	35	175
Commit (unlock ALL)		90	50	35	175

Взаємне блокування

- Взаємоблокування являє собою таку ситуацію, в якій дві або кілька транзакцій одночасно перебувають у стані очікування, причому кожна з них очікує, поки одна з решти транзакцій не звільнить блокування.
- Вирішення – вибір транзакції «жертви»

Упорядкованість

- Упорядкованість - це загальноприйнятий критерій правильної організації почергового виконання набору транзакцій; іншими словами, така організація виконання вважається правильною тоді і тільки тоді, коли вона є впорядковуваною (?).
- Тобто, якщо при почерговому виконанні заданої множини транзакцій буде отримано такий самий результат, що й при послідовному

Обґрунтування

1. Окремі транзакції є правильними
2. Будь-яка послідовність транзакцій, на підставі якої може бути організовано виконання транзакцій одна за одною, також є правильною
3. Почергове виконання операцій окремих транзакцій можна вважати коректним, якщо отримані результати будуть еквівалентні тим, що були б отримані при послідовному виконанні цих же транзакцій

Графік транзакцій

- Послідовність запуску операцій декількох паралельно виконуваних транзакцій, що зберігає черговість виконання операцій у кожній окремій транзакції.
- **Послідовний графік**. Графік, в якому операції кожної з транзакцій виконуються строго послідовно і не можуть чергуватися з операціями, виконуваними в інших транзакціях.
- **Непослідовний графік**. Графік, в якому чергуються операції з деякого набору одночасно виконуваних транзакцій.