Одеська Політехніка
Інститут комп`ютерних систем
Кафедра інформаційних систем
Дисципліна «Операційні системи»

Тема 2: Керування процесами
Лекція 10: Керування процесами-транзакціями в базах даних

Олександр А. Блажко,
доцент кафедри інформаційних систем,
E-mail: blazhko@ieee.org

Одеса, 08 травня 2023 року

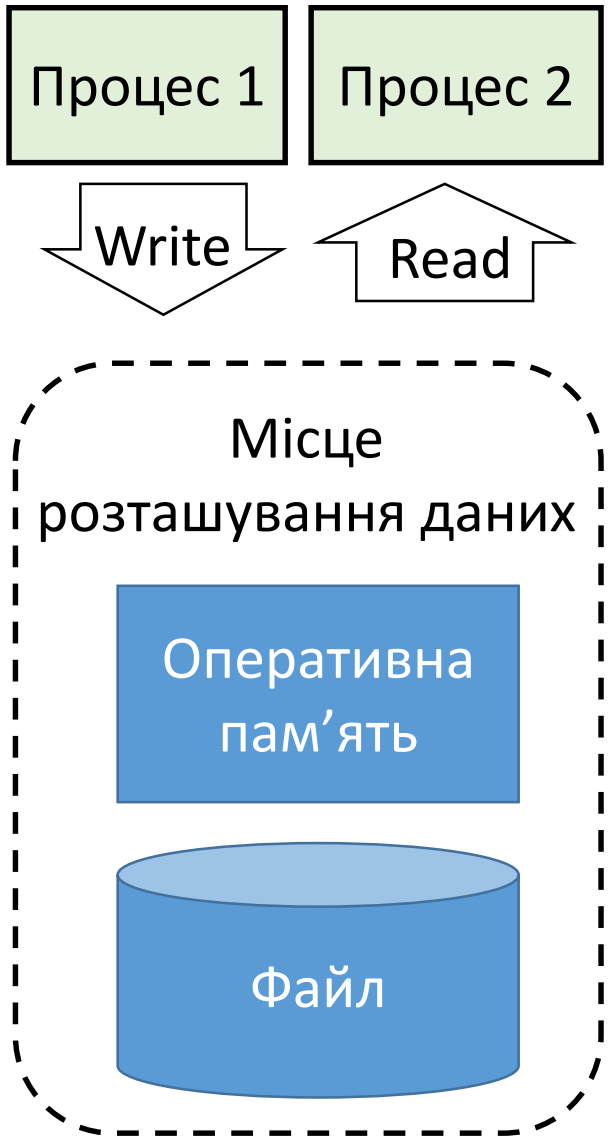
Рівні розміщення та деталіація даних як ресурсів для процесів

Процеси, які виконуються одночасно, завжди конкурують за ресурси ОС:

- процесор;
- *INPUT/OUTPUT*-пристрої для даних.

Дані розміщуються на різних рівнях деталізації:

- змінна в оперативній пам`яті, наприклад, *int B*;
- масив змінних в оперативній пам`яті, наприклад, *int M1={1,2}*;
- структура даних з декількома змінними в оперативній пам`яті, наприклад, *Struct S1 (int B; int D)*;
- окремий рядок у файлі файлової системи;
- файл файлової системи із множиною рядків.



Процеси-транзакції

В сучасних інформаційних системах процеси, які працюють з даними, часто називають Транзакціями.

Транзакція (англ. *Transactions* – ділова угода між учасниками) – це:

- впорядкована послідовність *CRUD*-операцій з даними: *Create B*, *Read B*, *Update B*, *Delete B*;
- оперативна зміна значень даних в оперативній пам'яті при виконанні операцій до моменту завершення транзакції;
- перенос всіх змінених даних з оперативної пам’яті у файл на диску під час завершення транзакції;
- задоволення *ACID*-вимог роботи з даними.

ACID-вимоги до роботи транзакцій

- Атомарність (***A****tomicity*) - виконується або вся транзакція цілком з усіма її CRUD-операціями, або всі її результати операцій скасовуються;
- Узгодженість (***C****onsistency*) - транзакція переводить дані із одного узгодженого стану в інший узгоджений стан, але всередині транзакції стан даних тимчасово може бути неузгоджений;
- Ізоляція (***I****solation*) – результат *CRUD*-операцій однієї транзакції не видно іншим транзакцій до тих пір, поки ця транзакція не завершиться;
- Довговічність (***D****urability*) - як тільки транзакція успішно завершилася, всі результати її *CRUD*-операцій залишаються незмінними і не можуть загубитися.

Абстрагування роботи транзакцій

Нехай O_{ij} – *j*-та операція *i*-ї транзакції.

Тоді транзакція №1 – $T_1 = O_{11}, O_{12}, O_{13}, \ldots, O_{1n}$

Атомарність забезпечує два стани, в які може перейти транзакція:

- стан успішного завершення;
- стан відміни всіх операцій.

Якщо операція *Oij стала* помилкою, тоді транзакція повинна відмінити результати попередніх операцій.

З урахуванням можливих станів внесемо додаткові позначки:

- C_i (***C**ommit*) – операція фіксації всіх змін даних без можливості відміни;
- A_i (***A**bort або Rollback*) –операція відміни всіх змін з даними, виконаних попередніми операціями.

Операції *Read* та *Write* можуть бути записані у скороченій нотації:

- $w_i[B]$ – *Write*-операція *i*-ї транзакції у змінну *B*;
- $r_i[B]$ – *Read*-операція *i*-ї транзакції зі змінної *B*.

Стан узгодженності транзакцій. Приклад

Приклад транзакції переносу суми грошей з банківського рахунку *B* на банківський рахунок *D*.

№	Операції T1	Значення
1	*r1[B]*	*B := 10*
2	*B := B - 5*	*B := 5*
3	*w1[B]*	*B := 5*
4	*r1[D]*	*D := 10*
5	*D := D + 5*	*D := 15*
6	*w1[D]*	*D := 15*
7	*C1*	

Абстрагування роботи транзакцій

- При одночасній, паралельній роботі транзакцій кожна операція, як процес ОС, виконується на процесорі на основі витісняючої мультизадачності, по черзі змінюючи одна одну.
- Але на рівні самих транзакцій та на рівні користувача, який їх розпочав, існує уявлення, що транзакції виконуються одночасно на процесорі.
- Для транзакцій робота є напівпаралельною або *квазіпаралельною*.
- Квазіпаралельна робота транзакцій формує *історію виконання транзакцій H (**H**istory).*
- Нехай одночасно розпочали свою роботу дві транзакції *T1, T2:*
- $T1 = O_{11} O_{12} O_{13}$
- $T2 = O_{21} O_{22} O_{23}$
- Тоді історія $H_{T1,T2} = O_{11} O_{21} O_{12} O_{22} O_{13} O_{23}$

Суперечливість станів транзакцій. Феномени

Квазіпаралельна робота транзакцій може призвести до суперечливого стану даних у вигляді так званих феноменів (*PHENOMENA*):

- Феномен *P1* - "Брудне Читання" ("*Dirty Read*");
- феномен *P2* - "Брудна Модифікація" ("*Dirty Write*");
- феномен *P3* - "Неповторне Читання" ("*Non-repeatable Read*")

  Враховуючи введені позначки, феномени *P1, P2, P3* можуть бути представлені як заборона на послідовність операцій, яка призводить до такої історії виконання транзакцій:
- *P1*: *w1[x] ... r2[x] ...* (*a1* AND *c2* у будь-якому порядку)
- *P2*: *w1[x] ... w2[x] ...* (*a1* AND *c2* у будь-якому порядку)
- *P3*: *r1[P] ... w2[y in P] ... a2 ... r1[P] ... c1*

Приклад феномену *P1* - "*Dirty Read*"

N	Операції T1	Операції T2	Значення
1	*r1[B]*		*B := 10*
2	*B := B - 5*		*B := 5*
3	*w1[B]*		*B := 5*
4		*r2[B]*	*B := 5*
5	*A1*		*B := 10*

В результаті роботи всіх операцій, на 5-му кроці транзакція *T2* вже виконала "*Dirty Read* "- операцію *r2[B],* тому що значення, яке вона читала на 4-му кроці, вже не існує.

Приклад феномену *P2* - "*Dirty Write*"

N	Операції *T1*	Операції *T2*	Значення
1	*r1[B]*		*B := 10*
2	*B := B - 5*		*B := 5*
3	*w1[B]*		*B := 5*
4		*r2[B]*	*B := 5*
5		*B := B – 5*	*B := 0*
6		*w2[B]*	*B := 0*
7	*A1*		*B := 10*

В результаті роботи всіх операцій, на 7-му кроці транзакція *T2* вже виконала "*Dirty Write*"-операцію *w2[B],* тому що значення, яке вона змінила читала на 6-му кроці, вже не існує.

Приклад феномену *P3* - "*Non-repeatable Read*"

N	Операції T1	Операції T2	Значення
1	*r1[B]*		*B := 10*
2	B := B - 5		*B := 5*
3	*w1[B]*		*B := 5*
4		*r2[B]*	*B := 5*
5		*r2[D]*	*D := 10*
6	*r1[D]*		*D := 10*
7	*D := D + 5*		*D := 15*
8	*w1[D]*		*D := 15*
9	*A1*		*B := 10, D := 10*
10		*C2*	

На 4-му та 5-му кроках дані знаходяться в неузгодженому стані і транзакція T2 отримає неправильний результат підрахунку балансу *B* та *D*.

Протидія появі феноменів

Якщо *Транзакція* – це документ з текстом ділової угоди між декількома учасниками,
тоді *Протокол* – це документ, в якому фіксуються всі дії в хронологічному порядку, які призвели до успішного створення транзакції.
Для ІТ *Протокол* – це алгоритм взаємодії двох та більше процесів.
При обміні мережевими повідомленнями між процесами використовується *Комунікаційний протокол* – правила передачі даних.
Для протидії появі феноменів пропонується *Протокол блокування* через дві додаткові операції:

- *Si[B]* – операція загального блокування (***S****hared lock*) змінної *B*, яка виконується *i*-ю транзакцією;
- *Xi[B]* – операція монопольного блокування (*e****X****clusive lock*) змінної *B*, яка виконується *i*-ю транзакцією.

Додаткова операція *Ui* – зняття всіх блокувань, встановлені транзакцію *Ti*

Протокол 1-го ступеня блокування. Правила

1) перед операцією *wi[B]* включається запит на блокування *Xi[B]*
2) запит на блокування приймається, якщо він сумісний із вже встановленими блокуваннями *B* іншими транзакціями;
3) запити на блокування однієї транзакції завжди сумісні;
4) сумісність блокувань визначається матрицею сумісності;
5) якщо запит на блокування не сумісний із вже встановленими блокуваннями інших транзакцій, цей запит переходить у стан очікування (*Wait*), а транзакція переходить до стану чекання;
6) перед виконанням операції фіксації або відміни операцій транзакція знімає всі свої встановлені блокування, виконуючи операцію *U;*
7) після завершення будь-якої транзакції перевіряються всі запити на блокування, які ще чекають.

T1/T2	*X*	*S*	-
X	-	-	+
S	-	+	+
-	+	+	+

Керування протоколу блокування

Для забезпечення контролю запитів на блокування використовується таблиця блокувань, яка містить три колонки:

- назва змінної (стовпчика) з бази даних (таблиці);
- перелік успішно встановлених блокувань транзакціями;
- перелік запитів на блокування збоку транзакцій, які є несумісними із вже встановленими блокуваннями збоку інших транзакцій

Назва змінної	Перелік встановлених блокувань	Перелік запитів на блокування
B	*X1*	*X2*

Приклад роботи протоколу 1-го ступеня блокування

Протокол 1-го ступеня виключає виникнення феномену “*Dirty Write*"

N	Операції *T1*	Операції *T2*	Значення
1	*r1[B]*		*B := 10*
2	*B := B - 5*		*B := 5*
3	*X1[B]*		
4	*w1[B]*		*B := 5*
5		*r2[B]*	*B := 5*
6		*B := B – 5*	*B := 0*
7		*X2[B]*	*Wait*
8	*A1*		*B := 10*
9	*U1*		
10		*X2[B]*	
11		*W2[B]*	*B := 0*
12		*C2*	
13		*U2*	

- 3-й крок: транзакція *T1* виконує запит на блокування *X1[B]*. Змінна *B* ще не має інших блокування, тому запит успішно виконується.
- 7-м крок: транзакція *T2* виконує запит на блокування X2[B]. Але він не сумісний із вже встановленим блокуванням транзакції T1, тому він переходить у стан очікування.

Протокол 2-го ступеня блокування

1) використовує всі правила з протоколу 1-го ступеня
2) додаткове правило протоколу: перед операцією *ri[B]* включається запит на блокування *Si[B]*.

Протокол 2-го ступеня виключає виникнення феномену “*Dirty Read*"

N	Операції *T1*	Операції *T2*	Значення
1	*S1[B]*		*B := 10*
2	*r1[B]*		*B := 10*
3	*B := B - 5*		*B := 5*
4	*X1[B]*		*B := 5*
5	*w1[B]*		*B := 5*
6		*S2[B]*	*Wait*
7	*A1*		*B := 10*
8	*U1*		
9		*S2[B]*	
10		*r2[B]*	*B := 10*
11		*B := B – 5*	*B := 5*
12		*X2[B]*	
13		*W2[B]*	*B := 5*
14		*C2*	
15		*U2*	

Deadlock-стани транзакцій

На жаль, використання блокувань призводить до виникнення іншої проблеми – переходу транзакцій у *Deadlock-стан* (глухий кут), коли:

- дві чи більше транзакції одночасно знаходяться в стані очікування;
- для продовження роботи кожна з транзакцій очікує припинення виконання іншої транзакції.

Приклад виникнення *Deadlock*-стану для двох транзакцій:

- *T1 = r1[B] w1[D] C1*
- *T2 = r2[D] w2[B] C2*

Історія квазіпаралельного виконання транзакцій для протоколу 2-го ступеня блокування:

$H_{T1,T2}$ *= S1[B] r1[B] S2[D] r2[D] X1[D]–Wait X2[B]–Wait*

Виявлення *Deadlock*-станів транзакцій

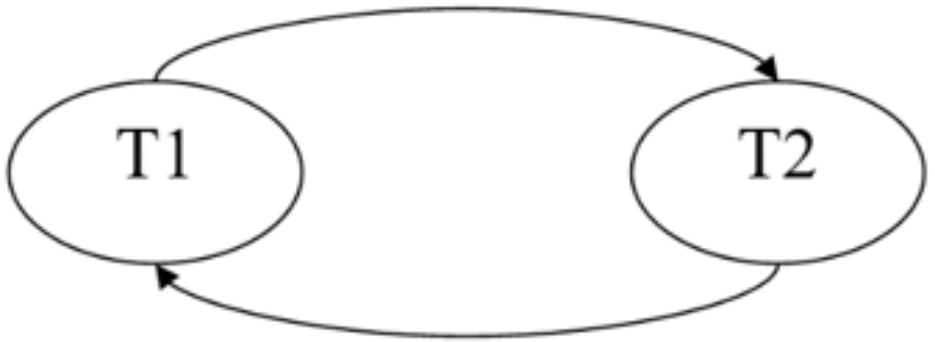
Для визначення *Deadlock*-стану взаємного блокування транзакцій створюється граф очікування транзакцій, використовуючи правила:

1) кожна транзакція визначає вузол графу (*T1, T2, T3* …);
2) між вузлом *T1* та вузлом *T2* створюється направлена дуга, якщо:
   - транзакція *T1* намагається виконати запит на блокування;
   - запит на блокування не сумісний із вже встановленими блокуваннями транзакції *T2*.

Якщо у графі буде знайдено цикл, це вказує на наявність *Deadlock*-стану.

Приклад історії:

H = S1[B] r1[B] S2[D] r2[D]
X1[D]–Wait
X2[B]–Wait



Керування множиною даних у формі реляційних таблиць

Для спрощення процесу керування множиною даних рекомендується їх представляти у вигляді бази даних (БД) як будь-який впорядкований набір даних різної структури.

Структурований текстовий *CSV*-формат БД:

- історичний *comma-separated values* – значення, розділені комою;
- *character-separated values* – будь-який символ-роздільник.

Одночасно з *Unix*-подібними ОС вже понад 50 років існують реляційні БД (Relations – відношення):

- файли з даними – таблиці з рядками однакової структури;
- рядки різних таблиць пов`язані між собою через однакові значення окремих стовпчиків

Файл */etc/passwd*

```
halt:x:7:0:halt:/sbin:/sbin/halt
operator:x:11:0:operator:/root:/sbin/nologin
root:x:0:0:root:/root:/bin/bash
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
sync:x:5:0:sync:/sbin:/bin/sync
```

Реляційна таблиця як файл */etc/group*

```
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:
```

Об’єднання таблиць-файлів

```
sort -n -t: -k 4 /etc/passwd > passwd_sort
sort -n -t: -k 3 /etc/group > group_sort
join -t: -j1 4 -j2 3 -o 1.1 2.1 1.5 ./passwd_sort ./group_sort
```

Одеська Політехніка
Інститут комп`ютерних систем
Кафедра інформаційних систем
Дисципліна «Операційні системи»

Дякую за увагу!
Запитання?

Олександр А. Блажко,
доцент кафедри інформаційних систем,
E-mail: blazhko@ieee.org
Telegram-канал: t.me/Operating_Systems_IS

Одеса, 08 травня 2023 року