

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

Лабораторна робота №10

з дисципліни

“Операційні системи”

Тема

«Керування процесами-транзакціями в базах даних. Частина 2»

Варіант 6(1)

Виконав:
Студент групи АІ-203
Сиваш А.І.

Одеса 2021

Мета: дослідити поведінку процесів-транзакцій в базах даних та засоби керування ними через механізм блокування з використанням сучасних систем керування базами даних.

Завдання для виконання:

Для кожної транзакції підготуйте окремий термінал, в якому виконайте команду доступу до вашої БД з використанням утиліти *psql*.

Завдання 1. Аналіз роботи багато версійного протоколу

В завданні 1 рішення попередньої лабораторної роботи було створено таблицю з декількома рядками.

Підготуйте чотири транзакції за прикладом з рисунку 2:

– T1 – отримання номеру транзакції, внесення нового рядка в таблицю та перегляд вмісту таблиці;

– T2 – постійний перегляд вмісту таблиці

– T3 – видалення рядку з наступною відміною цієї операції;

– T4 – зміна значення однієї з колонок рядка.

В операцію читання рядка таблиці додайте системні колонки *xmin*, *xmax*.

На кожному кроці виконання транзакції переглядайте значення колонок *xmin*, *xmax* та зробіть відповідні висновки.

Завдання 2. Аналіз стану транзакцій на різних рівнях багаторівневого блокування

Виконайте послідовно в двох терміналах наступні комбінації блокувань таблиці: *IX-IS*, *SIX-IX*, *SIX-IS*. Надайте висновки про сумісність блокувань.

Для кожної комбінації блокувань перед завершенням 1-ї транзакції (яка розпочалася раніше) в додатковому терміналі через команду *psql* отримайте данні про стан транзакцій (таблиця *pg_locks*).

Завдання 3. Керування квазіпаралельним виконанням транзакцій на різних рівнях ізоляції транзакцій

Підготуйте транзакції, які було створено у завданні 3.1 рішення попередньої лабораторної роботи, а саме, створіть дві транзакції, кожна з яких повинна включати такі операції:

- операція читання першого рядку таблиці;
- операція редагування однієї із змінних таблиці в першому рядку;
- повторна операція читання першого рядку таблиці;
- операція фіксації всіх змін.

1.1 Виконайте роботу транзакцій при умові їх роботи на рівні ізоляції *READ COMMITTED*. Проаналізуйте реакцію СКБД на операцію *UPDATE* 2-ї транзакції (яка виконується пізніше) та дайте свої висновки.

1.2 Повторіть роботу транзакцій при умові їх роботи на рівні ізоляції *REPEATABLE READ*. Проаналізуйте реакцію СКБД на операцію *UPDATE* 2-ї транзакції (яка виконується пізніше) та дайте свої висновки.

1.3 Повторіть роботу транзакцій при умові їх роботи на рівні ізоляції *SERIALIZABLE*. Проаналізуйте реакцію СКБД на операцію *UPDATE* 2-ї транзакції (яка виконується пізніше) та дайте свої висновки.

Завдання 4. Керування квазіпаралельним виконанням транзакцій при наявності тупикових ситуацій.

3.1 Виконайте модифікацію транзакцій так, щоб вони призводили до тупикової ситуації.

3.2 Виконайте дві модифіковані транзакції.

Проаналізуйте реакцію СКБД на операцію *UPDATE* 2-ї транзакції (яка виконується пізніше) та яка призвела до тупику. Дайте свої висновки з урахуванням:

- ідентифікаторів процесів
- номерів транзакцій.

Хід роботи

Завдання 1. Аналіз роботи багато версійного протоколу

T1

```
sivash_andrij=> SELECT * FROM student;
```

s_id	name	kurs
2	Petrov	1
1	Sivash	4

(2 rows)

```
sivash_andrij=> START TRANSACTION;
```

```
START TRANSACTION
```

```
sivash_andrij=> SELECT txid_current();
```

```
txid_current
```

```
-----  
3473
```

(1 row)

```
sivash_andrij=> INSERT INTO student VALUES(3, 'Ivanov', '3');
```

```
INSERT 0 1
```

```
sivash_andrij=> SELECT xmin, xmax, s_id, name, kurs FROM student;
```

xmin	xmax	s_id	name	kurs
3468	0	2	Petrov	1
3470	0	1	Sivash	4
3473	0	3	Ivanov	3

(3 rows)

```
sivash_andrij=> COMMIT;
```

```
COMMIT
```

T2

```
sivash_andrij=> SELECT xmin, xmax, s_id, name, kurs FROM student;
```

xmin	xmax	s_id	name	kurs
3468	0	2	Petrov	1
3470	0	1	Sivash	4

T2 не бачить зміни здійснені T1 до їх фіксації. Після фіксації до таблиці додається новий рядок, xmin якого дорівнює 3473 - номеру T1, яка виконала зміни.

T3

```

sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> DELETE FROM student WHERE s_id =2;
DELETE 1
sivash_andrij=> rollback;
ROLLBACK
sivash_andrij=> █

```

T2

```

sivash_andrij=> SELECT xmin, xmax, s_id, name, kurs FROM student;
  xmin | xmax | s_id |      name      | kurs
-----+-----+-----+-----+-----
  3470 |    0 |    1 | Sivash         |    4
  3473 |    0 |    3 | Ivanov         |    3
  3484 |    0 |    2 | Petrov         |    1
(3 rows)

```

```

sivash_andrij=> SELECT xmin, xmax, s_id, name, kurs FROM student;
  xmin | xmax | s_id |      name      | kurs
-----+-----+-----+-----+-----
  3470 |    0 |    1 | Sivash         |    4
  3473 |    0 |    3 | Ivanov         |    3
  3484 | 3485 |    2 | Petrov         |    1
(3 rows)

```

Після видалення рядку та відміни операції xmax рядку змінився на 3485, що показує, що над цим рядком здійснювалися операції транзакцією з номером 3485.

T4

```

kurgan_roman=> START TRANSACTION;
START TRANSACTION
kurgan_roman=> UPDATE worker SET bd='10/10/2010' WHERE p_id=3;
UPDATE 1
kurgan_roman=> commit;
COMMIT
kurgan_roman=> █

```

T2

```
sivash_andrij=> SELECT xmin, xmax, s_id, name, kurs FROM student;
```

xmin	xmax	s_id	name	kurs
3470	0	1	Sivash	4
3484	3485	2	Petrov	1
3487	0	3	Ivanov	2

(3 rows)

```
sivash_andrij=> SELECT xmin, xmax, s_id, name, kurs FROM student;
```

xmin	xmax	s_id	name	kurs
3470	0	1	Sivash	4
3484	3485	2	Petrov	1
3487	3488	3	Ivanov	2

(3 rows)

```
sivash_andrij=> SELECT xmin, xmax, s_id, name, kurs FROM student;
```

xmin	xmax	s_id	name	kurs
3470	0	1	Sivash	4
3484	3485	2	Petrov	1
3487	0	3	Ivanov	2

(3 rows)

хтах рядку 3 змінився на 3488, що означає виконання дій над ним, а після фіксації цих змін xmin = 3488, xmax = 0 - поточне значення було створено транзакцією з номером 3487, але поки немає нових версій, створених іншими транзакціями.

2. Аналіз стану транзакцій на різних рівнях багаторівневого блокування

IX - IS

```
sivash_andrij@vpsj3leQ:~
login as: sivash_andrij
sivash_andrij@91.219.60.189's password:
Access denied
sivash_andrij@91.219.60.189's password:
Last failed login: Wed May 5 23:28:59 EEST 2021 from 79.135.215.186 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Wed May 5 23:15:55 2021 from 79.135.215.186
[sivash_andrij@vpsj3leQ ~]$ psql
psql (9.5.25)
Type "help" for help.

sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> lock table student in row exclusive mode;
LOCK TABLE
sivash_andrij=> select relation, locktype, virtualtransaction, pid, mode, granted
d from pg_locks where locktype = 'relation';
 relation | locktype | virtualtransaction | pid | mode | granted
-----
11679 | relation | 2/825038 | 14439 | AccessShareLock | t
16858 | relation | 2/825038 | 14439 | RowExclusiveLock | t
16837 | relation | 7/23499 | 12677 | AccessShareLock | t
3455 | relation | 5/88130 | 13499 | AccessShareLock | t
2663 | relation | 5/88130 | 13499 | AccessShareLock | t
2662 | relation | 5/88130 | 13499 | AccessShareLock | t
2685 | relation | 5/88130 | 13499 | AccessShareLock | t
2684 | relation | 5/88130 | 13499 | AccessShareLock | t
2615 | relation | 5/88130 | 13499 | AccessShareLock | t
1259 | relation | 5/88130 | 13499 | AccessShareLock | t
16702 | relation | 3/94520 | 6750 | RowShareLock | t
(11 rows)

sivash_andrij=>
```

```
sivash_andrij@vpsj3leQ:~
Last login: Wed May 5 23:29:12 2021 from 79.135.215.186
[sivash_andrij@vpsj3leQ ~]$ psql
psql (9.5.25)
Type "help" for help.

sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> DELETE from student where s_id=2;
DELETE 1
sivash_andrij=> rollback;
ROLLBACK
sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> lock table student in row share mode;
LOCK TABLE
sivash_andrij=> rollback;
ROLLBACK
sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> lock table student in row share mode;
LOCK TABLE
sivash_andrij=> commit;
COMMIT
sivash_andrij=>
```

```
sivash_andrij=> select relation, locktype, virtualtransaction, pid, mode, granted from pg_locks where locktype = 'relation';
```

relation	locktype	virtualtransaction	pid	mode	granted
11673	relation	2/825038	14439	AccessShareLock	t
16858	relation	2/825038	14439	RowExclusiveLock	t
16855	relation	9/26853	14260	AccessShareLock	t
16855	relation	9/26853	14260	RowExclusiveLock	t
16837	relation	7/23489	12677	AccessShareLock	t
16855	relation	5/88131	13499	AccessShareLock	t
16702	relation	3/94520	6750	RowShareLock	t

(7 rows)

Блокування IX та IS сумісні.

SIX-IX

```
sivash_andrij=> select relation, locktype, virtualtransaction, pid, mode, granted from pg_locks where locktype = 'relation';
```

relation	locktype	virtualtransaction	pid	mode	granted
11673	relation	2/825039	14439	AccessShareLock	t
16855	relation	9/26854	14260	AccessShareLock	t
16855	relation	9/26854	14260	RowExclusiveLock	t
16837	relation	7/23489	12677	AccessShareLock	t
16855	relation	5/88132	13499	AccessShareLock	t

```
sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> lock table student in row exclusive mode;
LOCK TABLE
```

Блокування SIX та IX не сумісні.

```
sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> select relation, locktype, virtualtransaction, pid, mode, granted from pg_locks where locktype = 'relation';
```

relation	locktype	virtualtransaction	pid	mode	granted
11673	relation	4/140888	15549	AccessShareLock	t
16858	relation	4/140888	15549	RowExclusiveLock	t
11673	relation	2/825040	14439	AccessShareLock	t
16837	relation	7/24387	18257	ShareRowExclusiveLock	t
16837	relation	8/30637	18302	RowShareLock	t

```
sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> lock table student in row share mode;
LOCK TABLE
sivash_andrij=> commit;
COMMIT
```

SIX та IS сумісні.

3. Виконайте роботу транзакцій при умові їх роботи на рівні ізоляції READ COMMITTED

T1

```
sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=>
set transaction isolation level read committed ;
SET
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
    1 | Sivash                  |    4
(1 row)

sivash_andrij=> update student set name='Ivanov' where s_id=1;
UPDATE 1
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
    1 | Ivanov                  |    4
(1 row)

sivash_andrij=> commit;
COMMIT
```

T2

```
sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> set transaction isolation level read committed ;
SET
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
    1 | Sivash                  |    4
(1 row)

sivash_andrij=> update student set kurs='2' where s_id=1;
UPDATE 1
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
    1 | Ivanov                  |    2
(1 row)

sivash_andrij=> commit;
COMMIT
```

При виконанні операції update у T2 вона переходить в режим очікування і після завершення T1 успішно змінює дані.

Повторіть роботу транзакцій при умові їх роботи на рівні ізоляції REPEATABLE READ:

T1

```
sivash_andrij=> START TRANSACTION ;
START TRANSACTION
sivash_andrij=> set transaction isolation level repeatable read;
SET
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
    1 | Ivanov                 |    2
(1 row)

sivash_andrij=> update student set name='Petrov' where s_id=1;
UPDATE 1
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
    1 | Petrov                 |    2
(1 row)

sivash_andrij=> commit;
COMMIT
```

T2

```
sivash_andrij=> start transaction ;
START TRANSACTION
sivash_andrij=> set transaction isolation level repeatable read;
SET
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
    1 | Ivanov                 |    2
(1 row)

sivash_andrij=> update student set kurs='3' where s_id=1;
ERROR:  could not serialize access due to concurrent update
```

При виконанні операції update у T2 вона переходить в режим очікування і після завершення T1 повідомляє про помилку та завершує транзакцію без зміни даних.

Повторіть роботу транзакцій при умові їх роботи на рівні ізоляції
SERIALIZABLE:

T1

```
sivash_andrij=> Start transaction ;
START TRANSACTION
sivash_andrij=> set transaction isolation level serializable ;
SET
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
  1  | Petrov                 |    2
(1 row)

sivash_andrij=> update student set name='Ivanov' where s_id=1;
UPDATE 1
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
  1  | Ivanov                 |    2
(1 row)

sivash_andrij=> commit;
COMMIT
```

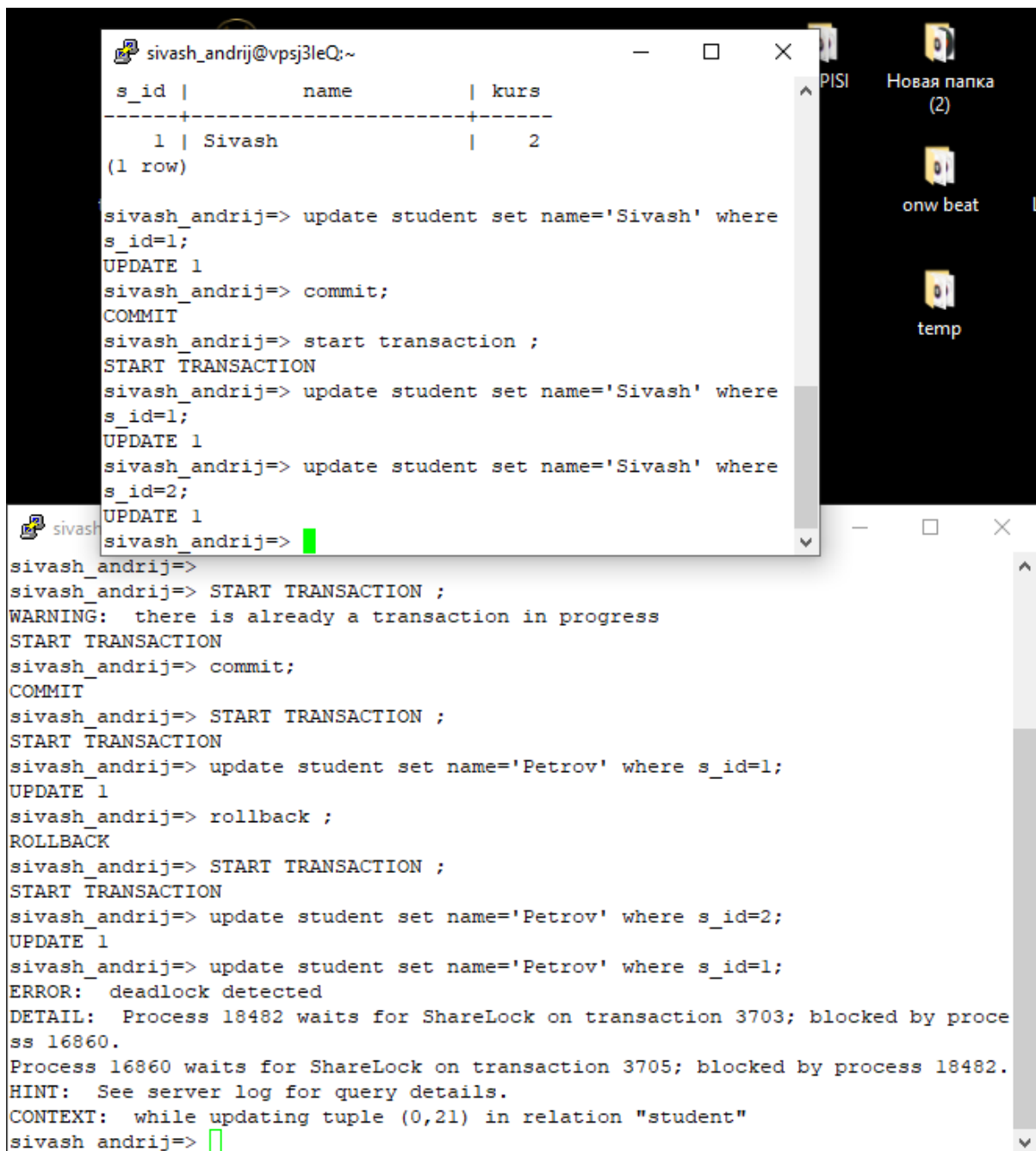
T2

```
sivash_andrij=> start TRANSACTION ;
START TRANSACTION
sivash_andrij=> set transaction isolation level serializable ;
SET
sivash_andrij=> select * from student where s_id=1;
s_id |          name          | kurs
-----+-----+-----
  1  | Petrov                 |    2
(1 row)

sivash_andrij=> update student set kurs='4' where s_id=1;
ERROR: canceling statement due to user request
UPDATE 1
```

При виконанні операції update у T2 вона переходить в режим очікування і після завершення T1 повідомляє про помилку та завершує транзакцію без зміни даних.

4. Керування квазіпаралельним виконанням транзакцій при наявності тупикових ситуацій.



```
sivash_andrij@vpsj3leQ:~  
s_id | name | kurs  
-----+-----+-----  
1 | Sivash | 2  
(1 row)  
  
sivash_andrij=> update student set name='Sivash' where  
s_id=1;  
UPDATE 1  
sivash_andrij=> commit;  
COMMIT  
sivash_andrij=> start transaction ;  
START TRANSACTION  
sivash_andrij=> update student set name='Sivash' where  
s_id=1;  
UPDATE 1  
sivash_andrij=> update student set name='Sivash' where  
s_id=2;  
UPDATE 1  
sivash_andrij=>  
  
sivash_andrij=>  
sivash_andrij=> START TRANSACTION ;  
WARNING: there is already a transaction in progress  
START TRANSACTION  
sivash_andrij=> commit;  
COMMIT  
sivash_andrij=> START TRANSACTION ;  
START TRANSACTION  
sivash_andrij=> update student set name='Petrov' where s_id=1;  
UPDATE 1  
sivash_andrij=> rollback ;  
ROLLBACK  
sivash_andrij=> START TRANSACTION ;  
START TRANSACTION  
sivash_andrij=> update student set name='Petrov' where s_id=2;  
UPDATE 1  
sivash_andrij=> update student set name='Petrov' where s_id=1;  
ERROR: deadlock detected  
DETAIL: Process 18482 waits for ShareLock on transaction 3703; blocked by process 16860.  
Process 16860 waits for ShareLock on transaction 3705; blocked by process 18482.  
HINT: See server log for query details.  
CONTEXT: while updating tuple (0,21) in relation "student"  
sivash_andrij=>
```

При виконанні операції update у T2 вона отримала повідомлення про помилку, так як очікувала завершення T1, а T1 очікувала завершення T2 - це призвело до тупіка, тому необхідно було примусово завершити транзакцію, що призвела до нього, тобто T2.

Висновок:

Найважчим у даній роботі виявилась робота із транзакціями.