

МІНІСТЕРСТВО ОСВІТИ Й НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
Інститут комп'ютерних систем
Кафедра інформаційних систем

Лабораторна робота № 10
За дисципліною: "Операційні системи"
Тема: ««Керування процесами-транзакціями в базах даних. Частина 2»

Виконав:
Студент групи АІ-205
Шостак Р.С.
Перевірили:
Блажко О.А.
Дрозд М.О.

Одеса 2021

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

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

The image shows two side-by-side screenshots of a PostgreSQL terminal window. The left window shows the execution of transaction T1, and the right window shows the execution of transaction T2.

Transaction T1 (Left Window):

```
shostak_roman@vpsj3leQ:~$ login as: shostak_roman
shostak_roman@91.219.60.189's password:
Last login: Wed May 12 11:48:06 2021 from 62.16.28.110
[shostak_roman@vpsj3leQ ~]$ psql
psql (9.5.25)
Type "help" for help.

shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> select* from university;
 u_id | name | year
-----+-----+-----
 2 | ONEU | 1930
 1 | ONFU | 1940
(2 rows)

shostak_roman=> insert into university values (3,'MECH',1900);
INSERT 0 1
shostak_roman=> select xmin,xmax,year from university;
 xmin | xmax | year
-----+-----+-----
 2628 | 0 | 1930
 2632 | 3560 | 1940
 3883 | 0 | 1900
(3 rows)

shostak_roman=> commit;
COMMIT
shostak_roman=>
```

Transaction T2 (Right Window):

```
shostak_roman@vpsj3leQ:~$ login as: shostak_roman
shostak_roman@91.219.60.189's password:
Last login: Sun May 16 22:48:06 2021 from 188-115-186-177.broadband.tenet.odessa.ua
[shostak_roman@vpsj3leQ ~]$ psql
psql (9.5.25)
Type "help" for help.

shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> select xmin,xmax,year from university;
 xmin | xmax | year
-----+-----+-----
 2628 | 0 | 1930
 2632 | 3560 | 1940
 3883 | 0 | 1900
(3 rows)

shostak_roman=> select xmin,xmax,year from university;
 xmin | xmax | year
-----+-----+-----
 2628 | 0 | 1930
 2632 | 3560 | 1940
 3883 | 0 | 1900
(3 rows)

shostak_roman=>
```

- T3 – видалення рядка з наступною відміною цієї операції;

```
shostak_roman=> select xmin,xmax,year from university;
 xmin | xmax | year
-----+-----+-----
 2628 | 0 | 1930
 2632 | 3560 | 1940
 3883 | 0 | 1900
(3 rows)

shostak_roman=> commit;
COMMIT
shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> delete from university where u_id=3;
DELETE 1
shostak_roman=> select xmin,xmax,year from university;
 xmin | xmax | year
-----+-----+-----
 2628 | 0 | 1930
 2632 | 3560 | 1940
(2 rows)

shostak_roman=> abort ;
ROLLBACK
```

- T4 – зміна значення однієї з колонок рядка.

```
shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> update university set year = '1905' wh
shostak_roman=> update university set year = '1905' wh

shostak_roman=> update university set year = '1905' where u_id=3;
UPDATE 1
shostak_roman=> select xmin,xmax,year from university;
 xmin | xmax | year
-----+-----+-----
 2628 | 0 | 1930
 2632 | 3560 | 1940
 3888 | 0 | 1905
(3 rows)

shostak_roman=> commit;
COMMIT
shostak_roman=>
```

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

```
shostak_roman=> start transaction ;
```

```
START TRANSACTION
```

```
shostak_roman=> lock TABLE uniu
```

```
shostak_roman=> lock TABLE university in row share mode;
```

```
LOCK TABLE
```

```
shostak_roman=> █
```

```
shostak_roman=> start transaction ;
```

```
START TRANSACTION
```

```
shostak_roman=> lock TABLE university in share row exclusive mode;
```

```
LOCK TABLE
```

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

relation	locktype	virtualtransaction	pid	mode	granted
11673	relation	12/21463	19641	AccessShareLock	t
16768	relation	10/29804	19292	RowExclusiveLock	t
3455	relation	11/16696	19632	AccessShareLock	t
2663	relation	11/16696	19632	AccessShareLock	t
2662	relation	11/16696	19632	AccessShareLock	t
2685	relation	11/16696	19632	AccessShareLock	t
2684	relation	11/16696	19632	AccessShareLock	t
2615	relation	11/16696	19632	AccessShareLock	t
1259	relation	11/16696	19632	AccessShareLock	t
16825	relation	11/16696	19632	RowShareLock	t
16870	relation	18/14455	24157	AccessShareLock	t
16870	relation	18/14455	24157	RowExclusiveLock	f
16870	relation	5/166413	16617	AccessShareLock	t
16870	relation	17/7712	24113	AccessShareLock	t
16870	relation	17/7712	24113	RowExclusiveLock	f
16870	relation	13/9830	21818	RowExclusiveLock	t
16870	relation	13/9830	21818	ShareRowExclusiveLock	t
16825	relation	12/21463	19641	ShareRowExclusiveLock	t
16870	relation	16/7742	22887	RowShareLock	t
16870	relation	14/12502	21844	RowShareLock	t
16870	relation	14/12502	21844	RowExclusiveLock	f

(21 rows)

```
shostak_roman=> █
```

, SIX-IX

```
shostak_roman=> start transaction ;
```

```
START TRANSACTION
```

```
shostak_roman=> lock TABLE uniu
```

```
shostak_roman=> lock TABLE university in row share mode;
```

```
LOCK TABLE
```

```
shostak_roman=> █
```

```
shostak_roman=> commit;
COMMIT
shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> lock TABLE university in share row exclusive mode;
LOCK TABLE
shostak_roman=> select relation,locktype,virtualtransaction,pid,mode,granted from pg_locks where locktype ='relation';
```

relation	locktype	virtualtransaction	pid	mode	granted
11673	relation	12/21464	19641	AccessShareLock	t
16768	relation	10/29804	19292	RowExclusiveLock	t
3455	relation	11/16696	19632	AccessShareLock	t
2663	relation	11/16696	19632	AccessShareLock	t
2662	relation	11/16696	19632	AccessShareLock	t
2685	relation	11/16696	19632	AccessShareLock	t
2684	relation	11/16696	19632	AccessShareLock	t
2615	relation	11/16696	19632	AccessShareLock	t
1259	relation	11/16696	19632	AccessShareLock	t
16825	relation	11/16696	19632	RowShareLock	t
16870	relation	18/14455	24157	AccessShareLock	t
16870	relation	18/14455	24157	RowExclusiveLock	f
16870	relation	5/166413	16617	AccessShareLock	t
16870	relation	17/7712	24113	AccessShareLock	t
16870	relation	17/7712	24113	RowExclusiveLock	f
16870	relation	13/9830	21818	RowExclusiveLock	t
16870	relation	13/9830	21818	ShareRowExclusiveLock	t
16825	relation	12/21464	19641	ShareRowExclusiveLock	t
16870	relation	19/9805	24738	AccessShareLock	t
16870	relation	19/9805	24738	RowExclusiveLock	f
16870	relation	16/7742	22887	RowShareLock	t
16870	relation	14/12502	21844	RowShareLock	t
16870	relation	14/12502	21844	RowExclusiveLock	f
16870	relation	20/8997	25022	AccessShareLock	t

(24 rows)

```
shostak_roman=> █
```

, SIX-IS

```
shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> lock TABLE university in share row exclusive mode;
LOCK TABLE
shostak_roman=> select relation,locktype,virtualtransaction,pid,mode,granted from pg_locks where locktype ='relation';
```

relation	locktype	virtualtransaction	pid	mode	granted
11673	relation	12/21465	19641	AccessShareLock	t
16768	relation	10/29804	19292	RowExclusiveLock	t
3455	relation	11/16696	19632	AccessShareLock	t
2663	relation	11/16696	19632	AccessShareLock	t
2662	relation	11/16696	19632	AccessShareLock	t
2685	relation	11/16696	19632	AccessShareLock	t
2684	relation	11/16696	19632	AccessShareLock	t
2615	relation	11/16696	19632	AccessShareLock	t
1259	relation	11/16696	19632	AccessShareLock	t
16825	relation	11/16696	19632	RowShareLock	t
16870	relation	18/14455	24157	AccessShareLock	t
16870	relation	18/14455	24157	RowExclusiveLock	f
16870	relation	22/3140	25367	RowExclusiveLock	f
16870	relation	5/166413	16617	AccessShareLock	t
16870	relation	17/7712	24113	AccessShareLock	t
16870	relation	17/7712	24113	RowExclusiveLock	f
16870	relation	13/9830	21818	RowExclusiveLock	t
16870	relation	13/9830	21818	ShareRowExclusiveLock	t
16825	relation	12/21465	19641	ShareRowExclusiveLock	t
16870	relation	19/9805	24738	AccessShareLock	t
16870	relation	19/9805	24738	RowExclusiveLock	f
16870	relation	16/7742	22887	RowShareLock	t
16870	relation	14/12502	21844	RowShareLock	t
16870	relation	14/12502	21844	RowExclusiveLock	f
16870	relation	20/8997	25022	AccessShareLock	t
16870	relation	21/5597	25052	AccessShareLock	t
16870	relation	21/5597	25052	RowExclusiveLock	f

(27 rows)

```
shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> lock TABLE uniu
```

```
shostak_roman=> lock TABLE university in row share mode;
LOCK TABLE
shostak_roman=> █
```

. 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.

```
START TRANSACTION
```

```
shostak_roman=> set transaction isolation level read committed ;
```

```
SET
```

```
shostak_roman=> select * from university;
```

u_id	name	year
2	ONEU	1930
1	ONPU	1940
3	MECH	1905

(3 rows)

```
shostak_roman=> update university set name = 'MECHNIK' where u_id= 3;
```

```
UPDATE 1
```

```
shostak_roman=> █
```

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.

```

shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> set transaction isolation level repeatable read ;
SET
shostak_roman=> select* from university;\
  u_id |          name          | year
-----+-----+-----
      2 | ONEU                   | 1930
      1 | ONPU                    | 1940
      3 | MECH                    | 1905
(3 rows)

Invalid command \. Try \? for help.
shostak_roman=> select* from university;
  u_id |          name          | year
-----+-----+-----
      2 | ONEU                   | 1930
      1 | ONPU                    | 1940
      3 | MECH                    | 1905
(3 rows)

shostak_roman=> update university set name = 'MECHNIK' u_id = 3;
ERROR:  syntax error at or near "u_id"
LINE 1: update university set name = 'MECHNIK' u_id = 3;
                                                ^

shostak_roman=> update university set name = 'MECHNIK' where u_id = 3;
ERROR:  current transaction is aborted, commands ignored until end of transaction block
shostak_roman=> █

```

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.

```

shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> set transaction isolation level read committed ;
SET
shostak_roman=> select * from university;
  u_id |          name          | year
-----+-----+-----
      2 | ONEU                   | 1930
      1 | ONPU                    | 1940
      3 | MECH                    | 1905
(3 rows)

shostak_roman=> update university set name = 'MECHNIK' where u_id = 3;
UPDATE 1
shostak_roman=> update university set name = 'MECHNIK' where u_id = 3;
UPDATE 1
shostak_roman=> set transaction isolation level serializable ;
ERROR:  SET TRANSACTION ISOLATION LEVEL must be called before any query
shostak_roman=> █

shostak_roman=> update university set name = 'MECHNIK' where u_id = 3;
ERROR:  current transaction is aborted, commands ignored until end of
shostak_roman=> abort;
ROLLBACK
shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> set transaction isolation level serializable;
SET
shostak_roman=> select* from university;
  u_id |          name          | year
-----+-----+-----
      2 | ONEU                   | 1930
      1 | ONPU                    | 1940
      3 | MECH                    | 1905
(3 rows)

shostak_roman=> update university set name = 'MECHNIK' where u_id = 3;
UPDATE 1
shostak_roman=> █

```

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.

```
shostak_roman=> start transaction ;
START TRANSACTION
shostak_roman=> select * from university;
```

u_id	name	year
2	ONEU	1930
1	ONPU	1940
3	MECH	1905

(3 rows)

```
START TRANSACTION
shostak_roman=> select * from university;
```

u_id	name	year
2	ONEU	1930
1	ONPU	1940
3	MECH	1905

(3 rows)

```
shostak_roman=> update university set u_id = 4 where u_id = 2;
^CCancel request sent
ERROR: canceling statement due to user request
```