ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІЕРСИТЕТ

Інститут комп’ютерних систем

Кафедра інформаційних систем

Лабораторна робота №10

З дисципліни «Операційні системи»

Тема: «Керування процесами-транзакціями в базах даних»

Виконав:

Студент групи АІ-202

Полянський М.О.

Перевірили:

Блажко О. А.

Дрозд М.О.

Одесса 2021

Мета роботи: дослідити поведінку процесів-транзакцій в базах даних та засоби керуванням 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

− 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.

T1	T2	T3	T4
start transaction;			
select txid_current();			
Insert into employer values (2,'Ivanov', 250);			
select xmin,xmax, salary from employer			
commit;			
	start transaction;		
	select xmin,xmax, salary from employer		
		start transaction;	
		Delete from employer where e_id = 2;	
	select xmin,xmax, salary from employer		

		rollback;	
	select xmin,xmax, salary from employer		
			start transaction;
			update employer set name = 'Petrov' where e_id =1;
	select xmin,xmax, salary from employer		
			commit;
	select xmin,xmax, salary from employer		
	commit;		

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_locs).

```
polyanskij_mikola=> start transaction;
START TRANSACTION
polyanskij_mikola=> lock table employer in row exclusive mode;
LOCK TABLE
polyanskij_mikola=> select relation,locktype,virtualtransaction,pid,mode,granted
from pg_locks
where locktype = 'relation';
 relation | locktype | virtualtransaction |  pid  |       mode       | granted
----------+----------+--------------------+-------+------------------+---------
    11673 | relation | 6/111822           |  4730 | AccessShareLock  | t
    16762 | relation | 6/111822           |  4730 | RowExclusiveLock | t
    16762 | relation | 9/48739            | 13632 | RowShareLock     | t
(3 rows)

polyanskij_mikola=>
```

```
polyanskij_mikola=> start transaction;
START TRANSACTION
polyanskij_mikola=> lock table employer in row share mode;
LOCK TABLE
polyanskij_mikola=>
```

SIX-IX

```
polyanskij_mikola=> start transaction;
START TRANSACTION
polyanskij_mikola=> lock table employer in share row exclusive mode;
LOCK TABLE
polyanskij_mikola=> select relation,locktype,virtualtransaction,pid,mode,granted
from pg_locks
where locktype = 'relation';
 relation | locktype | virtualtransaction |  pid  |         mode          | granted
----------+----------+--------------------+-------+-----------------------+---------
    11673 | relation | 2/1612207          | 22636 | AccessShareLock       | t
    11673 | relation | 6/111823           |  4730 | AccessShareLock       | t
     1259 | relation | 10/19611           | 15538 | AccessShareLock       | t
    16762 | relation | 6/111823           |  4730 | ShareRowExclusiveLock | t
    16762 | relation | 9/48740            | 13632 | RowExclusiveLock      | f
(5 rows)
```

```
polyanskij_mikola=> start transaction;
START TRANSACTION
polyanskij_mikola=> lock table employer in row exclusive mode;
```

SIX-IS

```
polyanskij_mikola=> start transaction;
START TRANSACTION
polyanskij_mikola=> lock table employer in share row exclusive mode;
LOCK TABLE
polyanskij_mikola=> select relation,locktype,virtualtransaction,pid,mode,granted
from pg_locks
where locktype = 'relation';
 relation | locktype | virtualtransaction |  pid  |         mode          | granted
----------+----------+--------------------+-------+-----------------------+---------
    11673 | relation | 2/1612207          | 22636 | AccessShareLock       | t
    11673 | relation | 6/111824           |  4730 | AccessShareLock       | t
    16762 | relation | 6/111824           |  4730 | ShareRowExclusiveLock | t
    16762 | relation | 9/48741            | 13632 | RowShareLock          | t
(4 rows)
```

```
polyanskij_mikola=> start transaction;
START TRANSACTION
polyanskij_mikola=> lock table employer in row share mode;
LOCK TABLE
polyanskij_mikola=>
```

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 COMMITED. Проаналізуйте реакцію СКБД на операцію UPDATE 2-ї транзакції (яка виконується пізніше) та дайте свої висновки.

```
polyanskij_mikola=> start transaction;
START TRANSACTION
polyanskij_mikola=> set transaction isolation level read committed;
SET
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name         | salary
------+----------------------+--------
    1 | Petrov               |    200
(1 row)

polyanskij_mikola=> update employer set name = 'Ivanov' where e_id =1;
UPDATE 1
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name         | salary
------+----------------------+--------
    1 | Ivanov               |    200
(1 row)

polyanskij_mikola=> commit;
COMMIT
```

```
START TRANSACTION
polyanskij_mikola=> set transaction isolation level read committed;
SET
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name         | salary
------+----------------------+--------
    1 | Petrov               |    200
(1 row)

polyanskij_mikola=> update employer set name = 'Polyanskiy' where e_id =1;
UPDATE 1
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name         | salary
------+----------------------+--------
    1 | Polyanskiy           |    200
(1 row)

polyanskij_mikola=> commit;
COMMIT
```

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.

```
polyanskij_mikola=> start transaction;
START TRANSACTION
polyanskij_mikola=> set transaction isolation level repeatable read;
SET
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name          | salary
------+-----------------------+--------
    1 | Petrov                |    200
(1 row)

polyanskij_mikola=> update employer set name = 'Polyanskiy' where e_id =1;
UPDATE 1
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name          | salary
------+-----------------------+--------
    1 | Polyanskiy            |    200
(1 row)

polyanskij_mikola=> commit;
COMMIT
```

```
START TRANSACTION
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name          | salary
------+-----------------------+--------
    1 | Ivanov                |    200
(1 row)

polyanskij_mikola=> update employer set name = 'Petrov' where e_id =1;
UPDATE 1
polyanskij_mikola=> commit;
COMMIT
polyanskij_mikola=> start transaction;
START TRANSACTION
polyanskij_mikola=> set transaction isolation level repeatable read;
SET
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name          | salary
------+-----------------------+--------
    1 | Petrov                |    200
(1 row)

polyanskij_mikola=> update employer set name = 'Ivanov' where e_id =1;
ERROR:  could not serialize access due to concurrent update
```

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.

```
START TRANSACTION
polyanskij_mikola=> set transaction isolation level serializable;
SET
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name         | salary
------+----------------------+--------
    1 | Polyanskiy           |    200
(1 row)

polyanskij_mikola=> update employer set name = 'Ivanov' where e_id =1;
UPDATE 1
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name         | salary
------+----------------------+--------
    1 | Ivanov               |    200
(1 row)

polyanskij_mikola=> commit;
COMMIT
polyanskij_mikola=>
```

```
START TRANSACTION
polyanskij_mikola=> set transaction isolation level serializable;
SET
polyanskij_mikola=> select *from employer where e_id = 1;
 e_id |         name         | salary
------+----------------------+--------
    1 | Polyanskiy           |    200
(1 row)

polyanskij_mikola=> update employer set name = 'Petrov' where e_id =1;
ERROR:  could not serialize access due to concurrent update
polyanskij_mikola=> update employer set name = 'Petrov' where e_id =1;
ERROR:  current transaction is aborted, commands ignored until end of transactio
n block
polyanskij_mikola=> select *from employer where e_id = 1;
ERROR:  current transaction is aborted, commands ignored until end of transactio
n block
polyanskij mikola=>
```

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

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

```
START TRANSACTION
polyanskij_mikola=> update employer set name = 'Polyanskiy' where e_id =1;
UPDATE 1
polyanskij_mikola=> update employer set name = 'Ivanov' where e_id =2;
UPDATE 1
polyanskij_mikola=>
```

```
START TRANSACTION
polyanskij_mikola=> update employer set name = 'Petrov' where e_id =2;
UPDATE 1
polyanskij_mikola=> update employer set name = 'Petrov' where e_id =1;
ERROR:  deadlock detected
DETAIL:  Process 13632 waits for ShareLock on transaction 3768; blocked by proce
ss 4730.
Process 4730 waits for ShareLock on transaction 3769; blocked by process 13632.
HINT:  See server log for query details.
CONTEXT:  while updating tuple (0,18) in relation "employer"
polyanskij mikola=>
```

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.

```
[polyanskij_mikola@vpsj3IeQ ~]$ ps -u polyanskij_mikola -o pid,ppid,cmd
  PID  PPID CMD
 4688  4666 sshd: polyanskij_mikola@pts/5
 4689  4688 -bash
 4729  4689 psql polyanskij_mikola
13602 13579 sshd: polyanskij_mikola@pts/8
13603 13602 -bash
13631 13603 psql polyanskij_mikola
24675 24653 sshd: polyanskij_mikola@pts/9
24676 24675 -bash
25402 24676 ps -u polyanskij_mikola -o pid,ppid,cmd
```

Висновок: в ході лабораторної роботи дослідили поведінку процесів-транзакцій в базах даних та засоби керуванням ними через механізм блокування з використанням сучасних систем керування базами даних.