

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».

Виконала:
студентка групи АІ-204
Томчук Вікторія Борисівна

Перевірив:
Блажко О.А.

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.

2 Завдання

Завдання 1. Аналіз роботи багато версійного протоколу

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

- 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.

Завдання 1.

- T1 – отримання номеру транзакції, внесення нового рядка в таблицю та перегляд вмісту таблиці;
- T2 – постійний перегляд вмісту таблиці

В даних транзакціях можна помітити, як xmax має відмінне від інших значення. До цього була робота з даними рядками, і збирається видалити рядок 2185. Спочатку не бачить рядка, тому що T1 не було завершено, після успішного завершення T1, у T2 починає відображатись створений рядок.

```
tomchuk_viktoriya@vpsj3leQ:~  
tomchuk_viktoriya=> START TRANSACTION;  
START TRANSACTION  
tomchuk_viktoriya=> SELECT xmin, xmax, name FROM employer;  
xmin | xmax | name  
-----  
2134 | 0 | Robin  
2183 | 2185 | Admin  
(2 rows)  
  
tomchuk_viktoriya=> SELECT xmin, xmax, name FROM employer;  
xmin | xmax | name  
-----  
2134 | 0 | Robin  
2183 | 2185 | Admin  
3155 | 0 | Solo  
(3 rows)  
  
tomchuk_viktoriya=> 
```

```
tomchuk_viktoriya@vpsj3leQ:~  
tomchuk_viktoriya=> START TRANSACTION;  
START TRANSACTION  
tomchuk_viktoriya=> INSERT INTO employer VALUES (3, 'Solo', 5600);  
INSERT 0 1  
tomchuk_viktoriya=> SELECT xmin, xmax, name FROM employer;  
xmin | xmax | name  
-----  
2134 | 0 | Robin  
2183 | 2185 | Admin  
3155 | 0 | Solo  
(3 rows)  
  
tomchuk_viktoriya=> COMMIT;  
COMMIT  
tomchuk_viktoriya=> 
```

- T3 – видалення рядку з наступною відміною цієї операції;

Після видалення рядка у транзакція його бачить із номером транзакції, що збирається видалити рядок.

```
tomchuk_viktoriya@vpsj3leQ:~  
login as: tomchuk_viktoriya  
tomchuk_viktoriya@91.219.60.189's password:  
Last login: Tue May 4 23:20:21 2021 from 78.26.152.249  
[tomchuk_viktoriya@vpsj3leQ ~]$ psql  
psql (9.5.25)  
Type "help" for help.  
  
tomchuk_viktoriya=> START TRANSACTION;  
START TRANSACTION  
tomchuk_viktoriya=> DELETE FROM employer WHERE e_id = 2;  
DELETE 1  
tomchuk_viktoriya=> SELECT xmin, xmax, name FROM employer;  
xmin | xmax | name  
-----  
2183 | 2185 | Admin  
3155 | 0 | Solo  
(2 rows)  
  
tomchuk_viktoriya=> ROLLBACK;  
ROLLBACK  
tomchuk_viktoriya=> 
```

```
tomchuk_viktoriya@vpsj3leQ:~  
Last login: Tue May 4 23:30:40 2021 from 78.26.152.249  
[tomchuk_viktoriya@vpsj3leQ ~]$ psql  
psql (9.5.25)  
Type "help" for help.  
  
tomchuk_viktoriya=> START TRANSACTION;  
START TRANSACTION  
tomchuk_viktoriya=> SELECT xmin, xmax, name FROM employer;  
xmin | xmax | name  
-----  
2134 | 3162 | Robin  
2183 | 2185 | Admin  
3155 | 0 | Solo  
(3 rows)  
  
tomchuk_viktoriya=> SELECT xmin, xmax, name FROM employer;  
xmin | xmax | name  
-----  
2134 | 3162 | Robin  
2183 | 2185 | Admin  
3155 | 0 | Solo  
(3 rows)  
  
tomchuk_viktoriya=> 
```

- T4 – зміна значення однієї з колонок рядка.

Після прочитання, до наступної транзакції приходить інформація про зміни рядка з транзакцією, що можна помітити в таблиці xmax.

```
tomchuk_viktoriya@vpsj3leQ:~  
tomchuk_viktoriya=> START TRANSACTION;  
START TRANSACTION  
tomchuk_viktoriya=> UPDATE employer SET name = 'NoAdmin' WHERE e_id = 2;  
UPDATE 1  
tomchuk_viktoriya=> SELECT xmin, xmax, name FROM employer;  
xmin | xmax | name  
-----  
2183 | 2185 | Admin  
3155 | 0 | Solo  
3168 | 0 | NoAdmin  
(3 rows)  
  
tomchuk_viktoriya=> COMMIT;  
COMMIT  
tomchuk_viktoriya=> 
```

```
tomchuk_viktoriya@vpsj3leQ:~  
tomchuk_viktoriya=> START TRANSACTION;  
START TRANSACTION  
tomchuk_viktoriya=> SELECT xmin, xmax, name FROM employer;  
xmin | xmax | name  
-----  
2134 | 3168 | Robin  
2183 | 2185 | Admin  
3155 | 0 | Solo  
(3 rows)  
  
tomchuk_viktoriya=> SELECT xmin, xmax, name FROM employer;  
xmin | xmax | name  
-----  
2183 | 2185 | Admin  
3155 | 0 | Solo  
3168 | 0 | NoAdmin  
(3 rows)  
  
tomchuk_viktoriya=> 
```

Завдання 2. Виконайте послідовно в двох терміналах наступні комбінації блокувань таблиці: IX-IS, SIX-IX, SIX-IS. Надайте висновки про сумісність блокувань.

IX-IS:

Як можна побачити, режим IX (блокування окремих таблиць) сумісний з режимом IS (блокування з взаємним доступом окремих рядків таблиць).

```
tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoriya=> LOCK TABLE employer IN ROW EXCLUSIVE MODE;
LOCK TABLE
tomchuk_viktoriya=> █

tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoriya=> LOCK TABLE employer IN ROW SHARE MODE;
LOCK TABLE
tomchuk_viktoriya=> █

tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> select relation,locktype,virtualtransaction,pid,mode,granted from pg_locks where locktype = 'relation';
relation | locktype | virtualtransaction | pid | mode | granted
-----+-----+-----+-----+-----+-----
16633 | relation | 47/657 | 3468 | RowExclusiveLock | t
16633 | relation | 7/20621 | 3765 | RowShareLock | t
```

SIX-IX:

Як можна помітити, через вплив блокування SIX (блокування зі взаємним доступом блокування всіх рядків таблиці та монопольне блокування окремих рядків таблиці) на першій транзакції, друга транзакція з IX (блокування окремих таблиць) не може себе закінчити. Виходячи з цього, режими не є сумісними.

```
tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoriya=> LOCK TABLE employer IN SHARE ROW EXCLUSIVE MODE;
LOCK TABLE
tomchuk_viktoriya=> █

tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoriya=> LOCK TABLE employer IN ROW EXCLUSIVE MODE;
█

tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> SELECT relation,locktype,virtualtransaction,pid,mode,granted
from pg_locks where locktype = 'relation' and relation = 16633;
relation | locktype | virtualtransaction | pid | mode | granted
-----+-----+-----+-----+-----+-----
16633 | relation | 27/1052 | 6657 | RowExclusiveLock | f
16633 | relation | 30/1629 | 6473 | ShareRowExclusiveLock | t
(2 rows)

tomchuk_viktoriya=> █
```

SIX-IS:

Транзакції вдало заблокували таблиці. SIX (блокування зі взаємним доступом блокування всіх рядків таблиці та монопольне блокування окремих рядків таблиці) – IS (блокування зі взаємним доступом окремих рядків таблиці) сумісні.

```
tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoriya=> LOCK TABLE employer IN SHARE ROW EXCLUSIVE MODE;
LOCK TABLE
tomchuk_viktoriya=>

tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoriya=> LOCK TABLE employer IN ROW SHARE MODE;
LOCK TABLE
tomchuk_viktoriya=>

tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> SELECT relation,locktype,virtualtransaction,pid,mode,granted
from pg_locks where locktype = 'relation' and relation = 16633;
relation | locktype | virtualtransaction | pid | mode | granted
-----+-----+-----+-----+-----+-----
16633 | relation | 27/1053 | 6657 | RowShareLock | t
16633 | relation | 30/1631 | 6473 | ShareRowExclusiveLock | t
(2 rows)
tomchuk_viktoriya=>
```

Завдання 3. Підготовлюють транзакції, які було створено у завданні 3.1 рішення попередньої лабораторної роботи, а саме, створюють дві транзакції, кожна з яких повинна включати операції:

Транзакція 1	Транзакція 2
START TRANSACTION	START TRANSACTION
SELECT * FROM employer WHERE e_id = 1;	SELECT * FROM employer WHERE e_id = 1;
UPDATE employer SET name = 'Admin' WHERE e_id = 1;	UPDATE employer SET salary = 1600 WHERE e_id = 1;
SELECT * FROM employer WHERE e_id = 1;	SELECT * FROM employer WHERE e_id = 1;
COMMIT;	COMMIT;

Завдання 3.1. 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.

tomchuk_viktoria@vps3leQ~

tomchuk_viktoria=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoria=> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
tomchuk_viktoria=> SELECT * FROM employer;
e_id | name | salary
-----+-----+-----
1 | Admin | 1600
3 | Solo | 5600
2 | NoAdmin | 150
(3 rows)

tomchuk_viktoria=> UPDATE employer SET name = 'Admin' WHERE e_id = 1;
UPDATE 1
tomchuk_viktoria=> COMMIT;
COMMIT
tomchuk_viktoria=>

tomchuk_viktoria@vps3leQ~

tomchuk_viktoria=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoria=> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
tomchuk_viktoria=> SELECT * FROM employer;
e_id | name | salary
-----+-----+-----
1 | Admin | 1600
3 | Solo | 5600
2 | NoAdmin | 150
(3 rows)

tomchuk_viktoria=> UPDATE employer SET salary = 1600 WHERE e_id = 1;
UPDATE 1
tomchuk_viktoria=>
tomchuk_viktoria=> SELECT * FROM employer;
e_id | name | salary
-----+-----+-----
3 | Solo | 5600
2 | NoAdmin | 150
1 | Admin | 1600
(3 rows)

tomchuk_viktoria=> COMMIT;
COMMIT
tomchuk_viktoria=>

Завдання 3.2. 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.

tomchuk_viktoria@vps3leQ~

tomchuk_viktoria=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoria=> SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET
tomchuk_viktoria=> SELECT * FROM employer
tomchuk_viktoria-> ;
e_id | name | salary
-----+-----+-----
3 | Solo | 5600
2 | NoAdmin | 150
1 | Admin | 1600
(3 rows)

tomchuk_viktoria=> UPDATE employer SET name = 'Admin' WHERE e_id = 1;
UPDATE 1
tomchuk_viktoria=> SELECT * FROM employer;
e_id | name | salary
-----+-----+-----
3 | Solo | 5600
2 | NoAdmin | 150
1 | Admin | 1600
(3 rows)

tomchuk_viktoria=> COMMIT;
COMMIT
tomchuk_viktoria=>

tomchuk_viktoria@vps3leQ~

tomchuk_viktoria=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoria=> SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET
tomchuk_viktoria=> SELECT * FROM employer;
e_id | name | salary
-----+-----+-----
3 | Solo | 5600
2 | NoAdmin | 150
1 | Admin | 1600
(3 rows)

tomchuk_viktoria=> UPDATE employer SET salary = 1600 WHERE e_id = 1;

ERROR: could not serialize access due to concurrent update
tomchuk_viktoria=>
tomchuk_viktoria=> SELECT * FROM employer;
ERROR: current transaction is aborted, commands ignored until end of transaction block
tomchuk_viktoria=> UPDATE employer SET salary = 1600 WHERE e_id = 1;
ERROR: current transaction is aborted, commands ignored until end of transaction block
tomchuk_viktoria=> rollback;
ROLLBACK
tomchuk_viktoria=>

Завдання 3.3. 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.

tomchuk_viktoria@vps3leQ~

tomchuk_viktoria=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoria=> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET
tomchuk_viktoria=> SELECT * FROM employer;
e_id | name | salary
-----+-----+-----
3 | Solo | 5600
2 | NoAdmin | 150
1 | Admin | 1600
(3 rows)

tomchuk_viktoria=> UPDATE employer SET name = 'Admin' WHERE e_id = 1;
UPDATE 1
tomchuk_viktoria=> SELECT * FROM employer;
e_id | name | salary
-----+-----+-----
3 | Solo | 5600
2 | NoAdmin | 150
1 | Admin | 1600
(3 rows)

tomchuk_viktoria=> COMMIT;
COMMIT
tomchuk_viktoria=>

tomchuk_viktoria@vps3leQ~

tomchuk_viktoria=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoria=> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET
tomchuk_viktoria=> SELECT * FROM employer;
e_id | name | salary
-----+-----+-----
3 | Solo | 5600
2 | NoAdmin | 150
1 | Admin | 1600
(3 rows)

tomchuk_viktoria=> UPDATE employer SET salary = 1600 WHERE e_id = 1;

ERROR: could not serialize access due to concurrent update
tomchuk_viktoria=>
tomchuk_viktoria=> SELECT * FROM employer;
ERROR: current transaction is aborted, commands ignored until end of transaction block
tomchuk_viktoria=>

Завдання 4.1. Виконують модифікацію транзакцій так, щоб вони призводили до тупикової ситуації.

START TRANSACTION;	START TRANSACTION;
UPDATE employer SET name = 'Admin' WHERE e_id = 1;	UPDATE employer SET salary = 1200 WHERE e_id = 2;
UPDATE employer SET salary = 1600 WHERE e_id = 2;	UPDATE employer SET name = 'Addmin' WHERE e_id = 1;

Завдання 4.2. Виконують дві модифіковані транзакції.

Як можна помітити, виникає в обох транзакціях тупикова ситуація, яка закінчується скиданням навантаження, однак перша транзакція, виходячи з її першості, була збережена.

```
tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoriya=> UPDATE employer SET name = 'Admin
tomchuk_viktoriya'> ' WHERE e_id = 1;
UPDATE 1
tomchuk_viktoriya=> UPDATE employer SET salary = 1600 WHERE e_id = 2;
UPDATE 1
tomchuk_viktoriya=>

tomchuk_viktoriya@vpsj3leQ:~
tomchuk_viktoriya=> START TRANSACTION;
START TRANSACTION
tomchuk_viktoriya=> UPDATE employer SET salary = 1200 WHERE e_id = 2;
UPDATE 1
tomchuk_viktoriya=> UPDATE employer SET name = 'Addmin' WHERE e_id = 1;
ERROR:  deadlock detected
DETAIL:  Process 18503 waits for ShareLock on transaction 3223; blocked by process 18357.
Process 18357 waits for ShareLock on transaction 3226; blocked by process 18503.
HINT:  See server log for query details.
CONTEXT:  while updating tuple (0,15) in relation "employer"
tomchuk_viktoriya=>
```

Для аналізу результатів роботи транзакції додатково переглядають таблицю процесів. Бачимо, що обидва процеси (транзакції) є лідерами сесії та сплять (статус Ss):

```
tomchuk_viktoriya@vpsj3leQ:~
[ tomchuk_viktoriya@vpsj3leQ ~]$ ps -u postgres -o pid,ppid,stat,cmd | egrep "tomchuk_viktoriya"
3468 8763 Ss postgres: tomchuk_viktoriya tomchuk_viktoriya [local] idle in transaction
3765 8763 Ss postgres: tomchuk_viktoriya tomchuk_viktoriya [local] idle in transaction
[ tomchuk_viktoriya@vpsj3leQ ~]$
```

ВИСНОВОК: Вдалося дослідити поведінку процесів-транзакцій в базах даних та засоби керування ними через механізм блокування з використанням сучасних систем керування базами даних. Всі завдання було цікаво розбирати, використовуючи і вивчаючи процес роботи механізму брокіровок транзакцій.