ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
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-203
Грищенко О.Р.

Одеса 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 до виконання:

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_locs).

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.

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.

Результати виконання завдань:

1. Аналіз роботи багато версійного протоколу

T1

```
grishenko_oleksandra=> START TRANSACTION;
START TRANSACTION
grishenko_oleksandra=> select txid_current();
 txid_current
--------------
         2696
(1 row)

grishenko_oleksandra=> INSERT INTO airplane VALUES(3, 'RS-65', 1980);
INSERT 0 1
grishenko_oleksandra=> SELECT xmin,xmax,a_id,model,year FROM airplane;
 xmin | xmax | a_id |        model         | year
------+------+------+----------------------+------
 2072 |    0 |    2 | AST-136              | 2000
 2093 | 2095 |    1 | AT-20                | 1970
 2696 |    0 |    3 | RS-65                | 1980
(3 rows)

grishenko_oleksandra=> commit;
COMMIT
```

T2

```
grishenko_oleksandra=> START TRANSACTION;
START TRANSACTION
grishenko_oleksandra=> SELECT xmin,xmax,a_id,model,year FROM airplane;
 xmin | xmax | a_id |        model         | year
------+------+------+----------------------+------
 2072 |    0 |    2 | AST-136              | 2000
 2093 | 2095 |    1 | AT-20                | 1970
(2 rows)

grishenko_oleksandra=> SELECT xmin,xmax,a_id,model,year FROM airplane;
 xmin | xmax | a_id |        model         | year
------+------+------+----------------------+------
 2072 |    0 |    2 | AST-136              | 2000
 2093 | 2095 |    1 | AT-20                | 1970
 2696 |    0 |    3 | RS-65                | 1980
(3 rows)
```

T2 не бачить зміни здійснені T1 до їх фіксації. Після фіксації до таблиці додається новий рядок, xmin якого дорівнює 2696 - номеру T1, яка виконала зміни.

T3

```
grishenko_oleksandra=> START TRANSACTION;
START TRANSACTION
grishenko_oleksandra=> DELETE FROM airplane WHERE a_id=2;
DELETE 1
grishenko_oleksandra=> rollback;
ROLLBACK
```

T2

```
grishenko_oleksandra=> SELECT xmin,xmax,a_id,model,year FROM airplane;
 xmin | xmax | a_id |        model         | year
------+------+------+----------------------+------
 2072 | 2697 |    2 | AST-136              | 2000
 2093 | 2095 |    1 | AT-20                | 1970
 2696 |    0 |    3 | RS-65                | 1980
(3 rows)

grishenko_oleksandra=> SELECT xmin,xmax,a_id,model,year FROM airplane;
 xmin | xmax | a_id |        model         | year
------+------+------+----------------------+------
 2072 | 2697 |    2 | AST-136              | 2000
 2093 | 2095 |    1 | AT-20                | 1970
 2696 |    0 |    3 | RS-65                | 1980
(3 rows)
```

Після видалення рядку та відміни операції xmax рядку змінився на 2697, що показує, що над цим рядком здійснювалися операції транзакцією з номером 2697.

T4

```
grishenko_oleksandra=> START TRANSACTION;
START TRANSACTION
grishenko_oleksandra=> UPDATE airplane SET year=2010 WHERE a_id=3;
UPDATE 1
grishenko_oleksandra=> commit;
COMMIT
```

T2

```
grishenko_oleksandra=> SELECT xmin,xmax,a_id,model,year FROM airplane;
 xmin | xmax | a_id |        model         | year
------+------+------+----------------------+------
 2072 | 2697 |    2 | AST-136              | 2000
 2093 | 2095 |    1 | AT-20                | 1970
 2696 | 2698 |    3 | RS-65                | 1980
(3 rows)

grishenko_oleksandra=> SELECT xmin,xmax,a_id,model,year FROM airplane;
 xmin | xmax | a_id |        model         | year
------+------+------+----------------------+------
 2072 | 2697 |    2 | AST-136              | 2000
 2093 | 2095 |    1 | AT-20                | 1970
 2698 |    0 |    3 | RS-65                | 2010
(3 rows)
```

xmax рядку 3 змінився на 2698, що означає виконування дій над ним, а після фіксації цих змін xmin = 2698, xmax = 0 - поточне значення було створено транзакцією з номером 2698, але поки немає нових версії, створених іншими транзакціями.

2. Аналіз стану транзакцій на різних рівнях багаторівневого блокування

IX - IS

```
grishenko_oleksandra=> START TRANSACTION;
START TRANSACTION
grishenko_oleksandra=> lock table airplane in row exclusive mode;
LOCK TABLE
grishenko_oleksandra=> select relation,locktype,virtualtransaction,pid,mode,gran
ted from pg_locks where locktype='relation';
 relation | locktype | virtualtransaction | pid  |       mode       | granted
----------+----------+--------------------+------+------------------+---------
    16612 | relation | 4/75476            | 5430 | RowShareLock     | t
    11673 | relation | 2/583857           | 5363 | AccessShareLock  | t
    16612 | relation | 2/583857           | 5363 | RowExclusiveLock | t
(3 rows)

grishenko_oleksandra=> commit;
COMMIT
```

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

Блокування IX та IS сумісні. Це можна зрозуміти з таблиці, так як стан блокування(granted) для процесу 5430 дорівнює t, тобто блокування виконано.

SIX - IX

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> lock table airplane in share row exclusive mode;
LOCK TABLE
grishenko_oleksandra=> select relation,locktype,virtualtransaction,pid,mode,gran
ted from pg_locks where locktype='relation';
 relation | locktype | virtualtransaction | pid  |         mode          | grant
ed
----------+----------+--------------------+------+-----------------------+------
---
    11673 | relation | 2/583860           | 5363 | AccessShareLock       | t
    16612 | relation | 4/75478            | 5430 | RowExclusiveLock      | f
    16612 | relation | 2/583860           | 5363 | ShareRowExclusiveLock | t
(3 rows)

grishenko_oleksandra=> commit;
COMMIT
```

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> lock table airplane in row exclusive mode;
LOCK TABLE
grishenko_oleksandra=> commit;
COMMIT
```

Блокування SIX та IX не сумісні. Це можна зрозуміти з таблиці, так як стан блокування(granted) для процесу 5430 дорівнює f, тобто блокування чекає через несумісність із вже запущеним.

SIX - IS

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> lock table airplane in share row exclusive mode;
LOCK TABLE
grishenko_oleksandra=> select relation,locktype,virtualtransaction,pid,mode,gran
ted from pg_locks where locktype='relation';
 relation | locktype | virtualtransaction | pid  |         mode          | grant
ed
----------+----------+--------------------+------+-----------------------+------
---
    11673 | relation | 2/583861           | 5363 | AccessShareLock       | t
    16612 | relation | 4/75479            | 5430 | RowShareLock          | t
    16612 | relation | 2/583861           | 5363 | ShareRowExclusiveLock | t
(3 rows)

grishenko_oleksandra=> commit;
COMMIT
```

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> lock table airplane in row share mode;
LOCK TABLE
grishenko_oleksandra=> commit;
COMMIT
```

Блокування SIX та IS сумісні. Це можна зрозуміти з таблиці, так як стан блокування(granted) для процесу 5430 дорівнює t, тобто блокування виконано.

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

T1

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> set transaction isolation level read committed;
SET
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AT-20                | 1970
(1 row)

grishenko_oleksandra=> update airplane set year=1976 where a_id=1;
UPDATE 1
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AT-20                | 1976
(1 row)

grishenko_oleksandra=> commit;
COMMIT
```

T2

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> set transaction isolation level read committed;
SET
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AT-20                | 1970
(1 row)

grishenko_oleksandra=> update airplane set model='AI-203' where a_id=1;
UPDATE 1
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AI-203               | 1976
(1 row)

grishenko_oleksandra=> 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

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> set transaction isolation level repeatable read;
SET
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AI-203               | 1976
(1 row)

grishenko_oleksandra=> update airplane set year=2019 where a_id=1;
UPDATE 1
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AI-203               | 2019
(1 row)

grishenko_oleksandra=> commit;
COMMIT
```

T2

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> set transaction isolation level repeatable read;
SET
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AI-203               | 1976
(1 row)

grishenko_oleksandra=> update airplane set model='AI-21' where a_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

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> set transaction isolation level serializable;
SET
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AI-203               | 2019
(1 row)

grishenko_oleksandra=> update airplane set model='AI-21' where a_id=1;
UPDATE 1
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AI-21                | 2019
(1 row)

grishenko_oleksandra=> commit;
COMMIT
```

T2

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> set transaction isolation level serializable;
SET
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AI-203               | 2019
(1 row)

grishenko_oleksandra=> update airplane set year=2020 where a_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.

4. Керування квазіпаралельним виконанням транзакцій при наявності тупикових ситуацій.

T1

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> update airplane set year=2008 where a_id=1;
UPDATE 1
grishenko_oleksandra=> select * from airplane where a_id=1;
 a_id |        model         | year
------+----------------------+------
    1 | AI-21                | 2008
(1 row)

grishenko_oleksandra=> update airplane set year=2010 where a_id=2;
UPDATE 1
grishenko_oleksandra=> select * from airplane where a_id=2;
 a_id |        model         | year
------+----------------------+------
    2 | AST-136              | 2010
(1 row)

grishenko_oleksandra=> commit;
COMMIT
```

T2

```
grishenko_oleksandra=> start transaction;
START TRANSACTION
grishenko_oleksandra=> update airplane set year=2004 where a_id=2;
UPDATE 1
grishenko_oleksandra=> select * from airplane where a_id=2;
 a_id |        model         | year
------+----------------------+------
    2 | AST-136              | 2004
(1 row)

grishenko_oleksandra=> update airplane set year=1920 where a_id=1;
ERROR:  deadlock detected
DETAIL:  Process 14241 waits for ShareLock on transaction 2711; blocked by proce
ss 14172.
Process 14172 waits for ShareLock on transaction 2713; blocked by process 14241.
HINT:  See server log for query details.
CONTEXT:  while updating tuple (0,20) in relation "airplane"
```

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: Під час виконання лабораторної роботи жодне завдання не викликало складнощів.