

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

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»

Виконав:

Студент групи AI-202

Сідельніков М. В.

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.

Завдання:

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 дані 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.

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

2.2 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

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.

1-й термінал:

```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ psql sidelnikov_mikita sidelnikov_mikita  
psql (9.5.25)  
Type "help" for help.  
  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> SELECT txid_current();  
txid_current  
-----  
3233  
(1 row)  
  
sidelnikov_mikita=> INSERT INTO student VALUES(3, 'Potapenko', 4);  
INSERT 0 1  
sidelnikov_mikita=> SELECT xmin, xmax, * FROM student;  
xmin | xmax | s_id | name | kurs  
-----+-----+-----+-----+-----  
2579 | 0 | 2 | Bashirov | 2  
2586 | 0 | 1 | Rabinovich | 3  
3233 | 0 | 3 | Potapenko | 4  
(3 rows)  
  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> █
```

2-й термінал:

```
sidelnikov_mikita@vpsj3leQ:~  
[sidelnikov_mikita@vpsj3leQ ~]$ psql sidelnikov_mikita sidelnikov_mikita  
psql (9.5.25)  
Type "help" for help.  
  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> SELECT xmin, xmax, * FROM student;  
xmin | xmax | s_id | name | kurs  
-----+-----+-----+-----+-----  
2579 | 0 | 2 | Bashirov | 2  
2586 | 0 | 1 | Rabinovich | 3  
(2 rows)  
  
sidelnikov_mikita=> SELECT xmin, xmax, * FROM student;  
xmin | xmax | s_id | name | kurs  
-----+-----+-----+-----+-----  
2579 | 3239 | 2 | Bashirov | 2  
2586 | 0 | 1 | Rabinovich | 3  
(2 rows)  
  
sidelnikov_mikita=> SELECT xmin, xmax, * FROM student;  
xmin | xmax | s_id | name | kurs  
-----+-----+-----+-----+-----  
2579 | 3239 | 2 | Bashirov | 2  
2586 | 3241 | 1 | Rabinovich | 3  
3233 | 0 | 3 | Potapenko | 4  
(3 rows)  
  
sidelnikov_mikita=> SELECT xmin, xmax, * FROM student;  
xmin | xmax | s_id | name | kurs  
-----+-----+-----+-----+-----  
2579 | 3239 | 2 | Bashirov | 2  
2586 | 3241 | 1 | Rabinovich | 3  
3233 | 0 | 3 | Potapenko | 4  
(3 rows)  
  
sidelnikov_mikita=> SELECT xmin, xmax, * FROM student;  
xmin | xmax | s_id | name | kurs  
-----+-----+-----+-----+-----  
2579 | 3239 | 2 | Bashirov | 2  
3233 | 0 | 3 | Potapenko | 4  
3241 | 0 | 1 | Rabinovich | 1  
(3 rows)  
  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> ~
```

3-й термінал:

```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ psql sidelnikov_mikita sidelnikov_mikita  
psql (9.5.25)  
Type "help" for help.  
  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> DELETE FROM student WHERE s_id = 2;  
DELETE 1  
sidelnikov_mikita=> SELECT xmin, xmax, * FROM student  
sidelnikov_mikita-> ;  
  xmin | xmax | s_id |      name      | kurs  
-----+-----+-----+-----+-----  
 2586 |    0 |    1 | Rabinovich     |    3  
(1 row)  
  
sidelnikov_mikita=> ABORT;  
ROLLBACK  
sidelnikov_mikita=> █
```

4-й термінал:

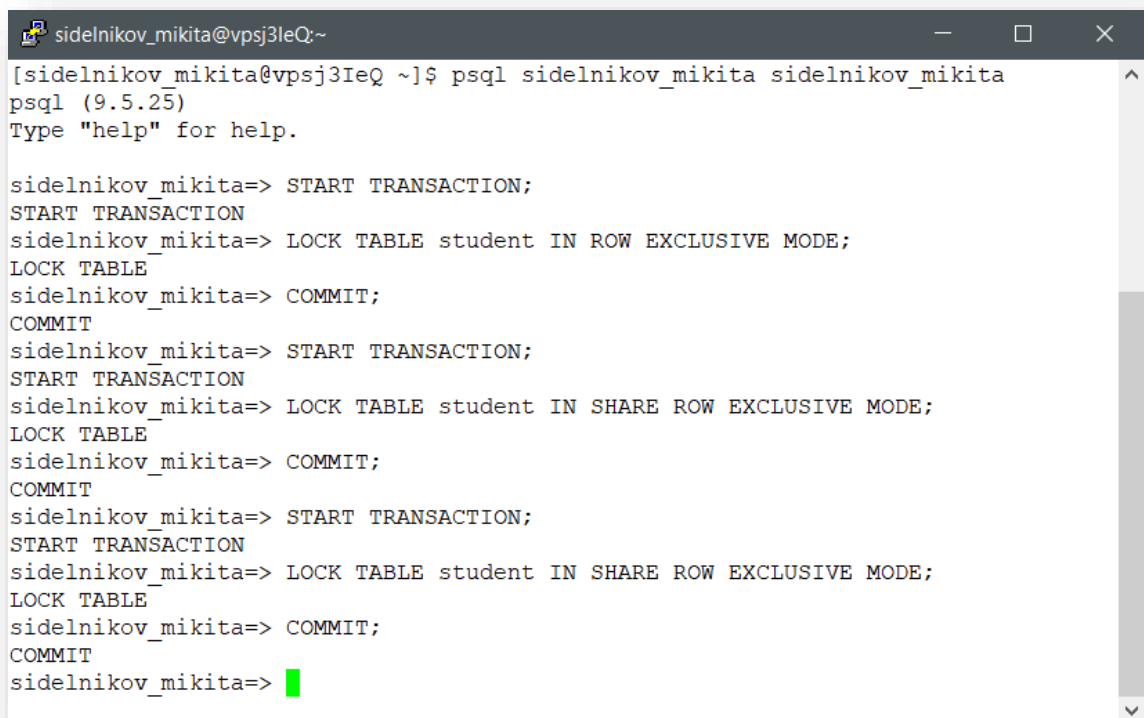
```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ psql sidelnikov_mikita sidelnikov_mikita  
psql (9.5.25)  
Type "help" for help.  
  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> UPDATE student SET kurs = 1 WHERE s_id = 1;  
UPDATE 1  
sidelnikov_mikita=> SELECT xmin, xmax, * FROM student;  
  xmin | xmax | s_id |      name      | kurs  
-----+-----+-----+-----+-----  
 2579 | 3239 |    2 | Bashirov       |    2  
 3241 |    0 |    1 | Rabinovich     |    1  
(2 rows)  
  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> █
```

Як бачимо, до завершення операції, інші бачити її результати не можуть.

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`).

1-й термінал:



```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ psql sidelnikov_mikita sidelnikov_mikita  
psql (9.5.25)  
Type "help" for help.  
  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> LOCK TABLE student IN ROW EXCLUSIVE MODE;  
LOCK TABLE  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> LOCK TABLE student IN SHARE ROW EXCLUSIVE MODE;  
LOCK TABLE  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> LOCK TABLE student IN SHARE ROW EXCLUSIVE MODE;  
LOCK TABLE  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> █
```


2-й термінал:

```
sidelnikov_mikita@vpsj3leQ:~  
[sidelnikov_mikita@vpsj3leQ ~]$ psql sidelnikov_mikita sidelnikov_mikita  
psql (9.5.25)  
Type "help" for help.  
  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> LOCK TABLE student IN ROW SHARE MODE;  
LOCK TABLE  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> LOCK TABLE student IN ROW EXCLUSIVE MODE;  
LOCK TABLE  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> LOCK TABLE student IN ROW SHARE MODE;  
LOCK TABLE  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> █
```

3-й термінал:

```
sidelnikov_mikita@vpsj3leQ:~  
sidelnikov_mikita=> SELECT relation, locktype, virtualtransaction, pid, mode, granted FROM pg_locks WHERE locktype = 'relation';  
relation | locktype | virtualtransaction | pid | mode | granted  
-----  
16801 | relation | 3/87114 | 15626 | RowShareLock | t  
16801 | relation | 2/800893 | 16015 | RowExclusiveLock | t  
11673 | relation | 12/16930 | 15739 | AccessShareLock | t  
16687 | relation | 15/7265 | 13237 | ShareRowExclusiveLock | f  
16687 | relation | 13/8291 | 12691 | RowExclusiveLock | f  
16687 | relation | 7/21581 | 10235 | RowExclusiveLock | t  
16687 | relation | 8/28030 | 11548 | ShareRowExclusiveLock | f  
16687 | relation | 10/11535 | 12113 | RowExclusiveLock | f  
16687 | relation | 9/21125 | 12686 | RowExclusiveLock | f  
16687 | relation | 11/10838 | 12168 | RowExclusiveLock | f  
(10 rows)
```

```
sidelnikov_mikita@vpsj3leQ:~$ SELECT relation, locktype, virtualtransaction, pid, mode, granted FROM pg_locks WHERE locktype = 'relation';
```

relation	locktype	virtualtransaction	pid	mode	granted
11673	relation	19/4100	16544	AccessShareLock	t
11673	relation	12/16931	15739	AccessShareLock	t
16687	relation	15/7265	13237	ShareRowExclusiveLock	f
16801	relation	3/87130	16564	RowExclusiveLock	f
16654	relation	5/85531	16288	RowExclusiveLock	f
16687	relation	13/8291	12691	RowExclusiveLock	f
16654	relation	4/108897	16092	ShareRowExclusiveLock	t
16687	relation	7/21581	10235	RowExclusiveLock	t
16687	relation	8/28030	11548	ShareRowExclusiveLock	f
16687	relation	10/11535	12113	RowExclusiveLock	f
16801	relation	2/800894	16015	ShareRowExclusiveLock	t
16687	relation	9/21125	12686	RowExclusiveLock	f
16687	relation	11/10838	12168	RowExclusiveLock	f

(13 rows)

```
sidelnikov_mikita@vpsj3leQ:~$ SELECT relation, locktype, virtualtransaction, pid, mode, granted FROM pg_locks WHERE locktype = 'relation';
```

relation	locktype	virtualtransaction	pid	mode	granted
11673	relation	19/4100	16544	AccessShareLock	t
11673	relation	12/16932	15739	AccessShareLock	t
16687	relation	15/7265	13237	ShareRowExclusiveLock	f
16801	relation	3/87131	16564	RowShareLock	t
16654	relation	5/85531	16288	RowExclusiveLock	f
16687	relation	13/8291	12691	RowExclusiveLock	f
16654	relation	4/108897	16092	ShareRowExclusiveLock	t
16687	relation	7/21581	10235	RowExclusiveLock	t
16687	relation	8/28030	11548	ShareRowExclusiveLock	f
16687	relation	10/11535	12113	RowExclusiveLock	f
16801	relation	2/800895	16015	ShareRowExclusiveLock	t
16687	relation	9/21125	12686	RowExclusiveLock	f
16687	relation	11/10838	12168	RowExclusiveLock	f

(13 rows)

```
sidelnikov_mikita=>
```

Тільки у другій комбінації отримали режим очікування, бо IX не сумісний SIX, отже друга транзакція очікує завершення першої.

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

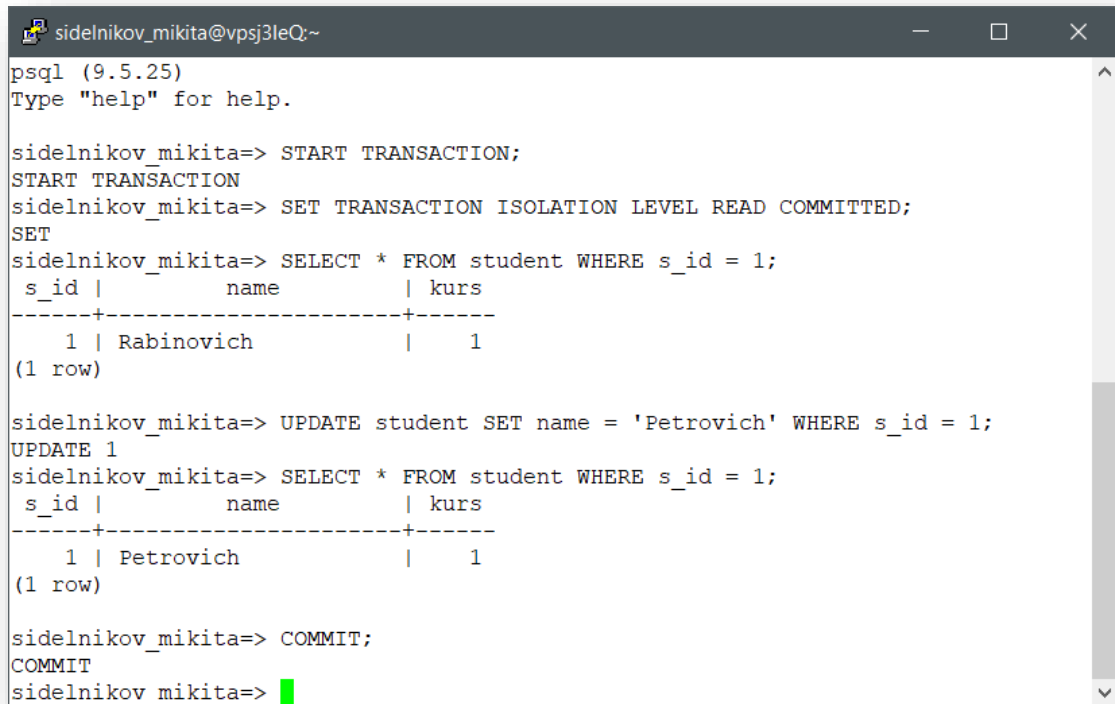
Підготуйте транзакції, які було створено у завданні 3.1 рішення попередньої лабораторної роботи, а саме, створіть дві транзакції, кожна з яких повинна включати такі операції:

- 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-й термінал:



```
sidelnikov_mikita@vpsj3leQ:~  
psql (9.5.25)  
Type "help" for help.  
  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
SET  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
 s_id |      name      | kurs  
-----+-----+-----  
    1 | Rabinovich     |    1  
(1 row)  
  
sidelnikov_mikita=> UPDATE student SET name = 'Petrovich' WHERE s_id = 1;  
UPDATE 1  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
 s_id |      name      | kurs  
-----+-----+-----  
    1 | Petrovich      |    1  
(1 row)  
  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> 
```

2-й термінал:

```
sidelnikov_mikita@vpsj3leQ:~  
psql (9.5.25)  
Type "help" for help.  
  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
SET  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
 s_id |      name      | kurs  
-----+-----+-----  
    1 | Rabinovich     |    1  
(1 row)  
  
sidelnikov_mikita=> UPDATE student SET name = 'Ivanovich' WHERE s_id = 1;  
UPDATE 1  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
 s_id |      name      | kurs  
-----+-----+-----  
    1 | Ivanovich      |    1  
(1 row)  
  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> 
```

Команда UPDATE 2-ї транзакції перевела її у режим очікування. Після завершення 1-ї транзакції продовжила своє виконання і замістило результат виконання 1-ї.

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-й термінал:

```
sidelnikov_mikita@vpsj3leQ:~  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;  
SET  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
s_id | name | kurs  
-----+-----+-----  
1 | Ivanovich | 1  
(1 row)  
  
sidelnikov_mikita=> UPDATE student SET name = 'Petrovich' WHERE s_id = 1;  
UPDATE 1  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
s_id | name | kurs  
-----+-----+-----  
1 | Petrovich | 1  
(1 row)  
  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>
```

2-й термінал:

```
sidelnikov_mikita@vpsj3leQ:~  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;  
SET  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
s_id | name | kurs  
-----+-----+-----  
1 | Ivanovich | 1  
(1 row)  
  
sidelnikov_mikita=> UPDATE student SET name = 'Samplovich' WHERE s_id = 1;  
ERROR: could not serialize access due to concurrent update  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>
```

Команда UPDATE 2-ї транзакції перевела її у режим очікування. Після завершення 1-ї транзакції визвала помилку через конкуруючий апдейт.

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.

1-й термінал:

```
sidelnikov_mikita@vpsj3leQ:~  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
SET  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
s_id | name | kurs  
-----+-----  
1 | Petrovich | 1  
(1 row)  
  
sidelnikov_mikita=> UPDATE student SET name = 'Optovich' WHERE s_id = 1;  
UPDATE 1  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
s_id | name | kurs  
-----+-----  
1 | Optovich | 1  
(1 row)  
  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>  
sidelnikov_mikita=>
```

2-й термінал:

```
sidelnikov_mikita@vpsj3leQ:~  
sidelnikov_mikita=> START TRANSACTION;  
ERROR: current transaction is aborted, commands ignored until end of transaction block  
sidelnikov_mikita=> ABORT;  
ROLLBACK  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
SET  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
s_id | name | kurs  
-----+-----+-----  
1 | Petrovich | 1  
(1 row)  
  
sidelnikov_mikita=> UPDATE student SET name = 'Samplovich' WHERE s_id = 1;  
ERROR: could not serialize access due to concurrent update  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
ERROR: current transaction is aborted, commands ignored until end of transaction block  
sidelnikov_mikita=> ABORT;  
ROLLBACK  
sidelnikov_mikita=>  
sidelnikov_mikita=>
```

Команда UPDATE 2-ї транзакції перевела її у режим очікування. Після завершення 1-ї транзакції визвала помилку через конкуруючий апдейт.

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

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

2.2 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.

1-й термінал:

```
sidelnikov_mikita@vpsj3IeQ:~  
[sidelnikov_mikita@vpsj3IeQ ~]$ psql sidelnikov_mikita sidelnikov_mikita  
psql (9.5.25)  
Type "help" for help.  
  
sidelnikov_mikita=> START TRANSACTION;  
START TRANSACTION  
sidelnikov_mikita=> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
SET  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;  
 s_id |      name      | kurs  
-----+-----+-----  
    1 | Petrov         |    1  
(1 row)  
  
sidelnikov_mikita=> UPDATE student SET name = 'Ivanov' WHERE s_id = 1;  
UPDATE 1  
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 2;  
 s_id |      name      | kurs  
-----+-----+-----  
    2 | Bashirov       |    2  
(1 row)  
  
sidelnikov_mikita=> UPDATE student SET name = 'Ivanov' WHERE s_id = 2;  
UPDATE 1  
sidelnikov_mikita=> COMMIT;  
COMMIT  
sidelnikov_mikita=> █
```

2-й термінал:


```
sidelnikov_mikita@vpsj3IeQ:~$ psql sidelnikov_mikita sidelnikov_mikita
psql (9.5.25)
Type "help" for help.

sidelnikov_mikita=> START TRANSACTION;
START TRANSACTION
sidelnikov_mikita=> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 2;
 s_id |      name      | kurs
-----+-----+-----
    2 | Bashirov       |    2
(1 row)

sidelnikov_mikita=> UPDATE student SET name = 'Ivanov' WHERE s_id = 2;
UPDATE 1
sidelnikov_mikita=> SELECT * FROM student WHERE s_id = 1;
 s_id |      name      | kurs
-----+-----+-----
    1 | Petrov         |    1
(1 row)

sidelnikov_mikita=> UPDATE student SET name = 'Ivanov' WHERE s_id = 1;
ERROR:  deadlock detected
DETAIL:  Process 22868 waits for ShareLock on transaction 3353; blocked by process 22831.
Process 22831 waits for ShareLock on transaction 3354; blocked by process 22868.
HINT:  See server log for query details.
CONTEXT:  while updating tuple (0,18) in relation "student"
sidelnikov_mikita=>
```

Тупик виникає при взаємном очікуванні двох процесів.

Транзакція 3353 процесу 22868 намагається змінити другий рядок, але він зайнятий другою транзакцією, тому вона переходить у режим очікування.

Транзакція 3354 процесу 22831 намагається змінити другий рядок, але він зайнятий першою транзакцією, тому вона переходить у режим очікування.

Дві транзакції очікують завершення один одного, отже виникає тупик.

Висновки: під час виконання лабораторної роботи було досліджено поведінку процесів-транзакцій в базах даних та засоби керування ними через механізм блокування з використанням сучасних систем керування базами даних.