

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»

Виконала:
ст.гр. АІ -204
Сіренко Марія

Перевірили:
Блажко О.А.
Дрозд М.О

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

Завдання

Для кожної транзакції підготуйте окремий термінал, в якому виконайте команду доступу до вашої БД з використанням утиліти `psql`.

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_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 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.

Виконання завдань:

Підготуємо чотири транзакції :

- 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) Транзакція 1 та частина транзакції 2

Ми можемо побачити ,що транзакція 1 (номер 2754) додала до таблиці auto рядок 3 (за колонкою xmin).Зміни можливо побачити після успішного завершення 1 транзакції

```
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> select txid_current();
txid_current
-----
2754
(1 row)

sirenko_mariya=> INSERT INTO auto VALUES (3,'Lanos DEO',2010);
INSERT 0 1
sirenko_mariya=> SELECT xmin,xmax,* FROM auto;
xmin | xmax | a_id | name | year
-----+-----+-----+-----+-----
2090 | 0 | 2 | Lanos DEO | 2010
2102 | 0 | 1 | Porche | 1999
2754 | 0 | 3 | Lanos DEO | 2010
(3 rows)

sirenko_mariya=> COMMIT;
COMMIT
sirenko_mariya=> █
```

```
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> SELECT xmin,xmax, * FROM auto;
xmin | xmax | a_id | name | year
-----+-----+-----+-----+-----
2090 | 0 | 2 | Lanos DEO | 2010
2102 | 0 | 1 | Porche | 1999
(2 rows)

sirenko_mariya=> SELECT xmin,xmax, * FROM auto;
xmin | xmax | a_id | name | year
-----+-----+-----+-----+-----
2090 | 0 | 2 | Lanos DEO | 2010
2102 | 0 | 1 | Porche | 1999
2754 | 0 | 3 | Lanos DEO | 2010
(3 rows)
```

2) Транзакція 3 та частина транзакції 2

Ми бачимо ,що у колонці xmax для a_id =3 , транзакція 3(номер 2755) намагається видалити рядок.Транзакція 3 скасовується ,але значення xmax залишається незмінним

```
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> DELETE FROM auto WHERE a_id = 3;
DELETE 1
sirenko_mariya=> ROLLBACK;
ROLLBACK
sirenko_mariya=> □
```

```
sirenko_mariya=> SELECT xmin,xmax, * FROM auto;
xmin | xmax | a_id |      name      | year
-----+-----+-----+-----+-----
2090 |    0 |    2 | Lanos DE0      | 2010
2102 |    0 |    1 | Porche         | 1999
2754 | 2755 |    3 | Lanos DE0      | 2010
(3 rows)

sirenko_mariya=> SELECT xmin,xmax, * FROM auto;
xmin | xmax | a_id |      name      | year
-----+-----+-----+-----+-----
2090 |    0 |    2 | Lanos DE0      | 2010
2102 |    0 |    1 | Porche         | 1999
2754 | 2755 |    3 | Lanos DE0      | 2010
(3 rows)

sirenko_mariya=> █
```

2) Транзакція 4 та частина транзакції 2

Ми бачимо ,що у колонці xmax для a_id =3 , транзакція 4(номер 2757) намагається змінити данні.Транзакція 4 успішно завершується ,а значення xmax змінюється на 0

```
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> UPDATE auto SET name = ' Mazda-3' WHERE a_id=3;
UPDATE 1
sirenko_mariya=> COMMIT;
COMMIT
sirenko_mariya=> █
```

```
sirenko_mariya=> SELECT xmin,xmax, * FROM auto;
xmin | xmax | a_id |      name      | year
-----+-----+-----+-----+-----
2090 |    0 |    2 | Lanos DE0      | 2010
2102 |    0 |    1 | Porche         | 1999
2754 | 2757 |    3 | Lanos DE0      | 2010
(3 rows)

sirenko_mariya=> SELECT xmin,xmax, * FROM auto;
xmin | xmax | a_id |      name      | year
-----+-----+-----+-----+-----
2090 |    0 |    2 | Lanos DE0      | 2010
2102 |    0 |    1 | Porche         | 1999
2757 |    0 |    3 | Mazda-3        | 2010
(3 rows)

sirenko_mariya=> █
```

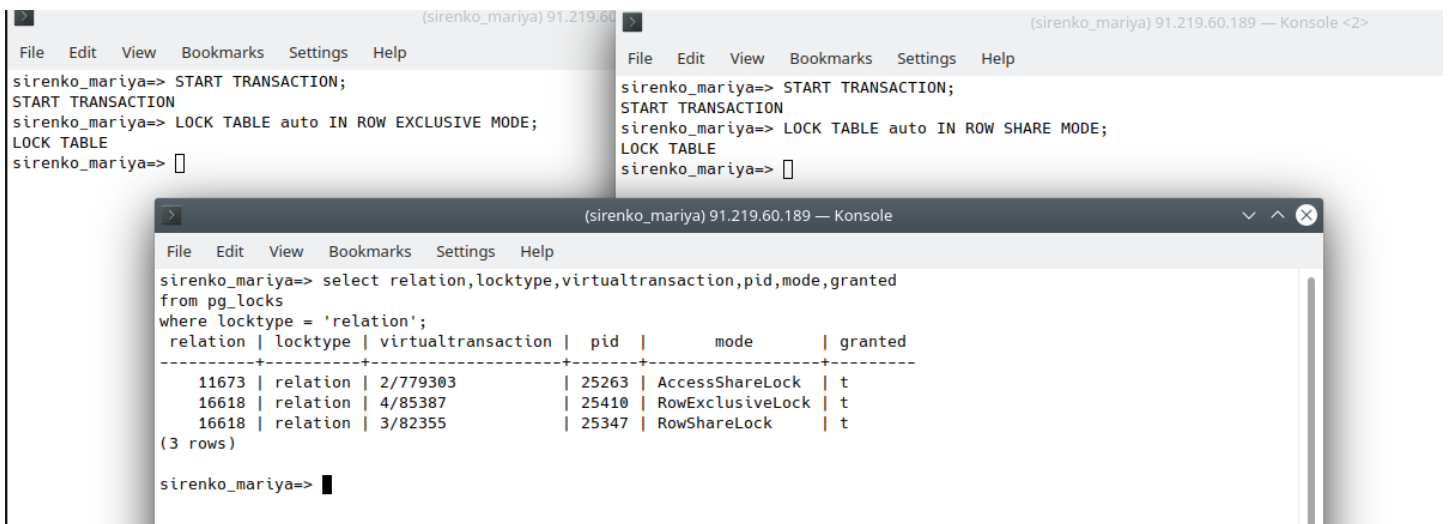
Завдання 2. Аналіз стану транзакцій на різних рівнях багаторівневого блокування

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) Блокування IX-IS є сумісними ,тому транзакції проходять без проблем. У колонці `grated` стоїть `t`



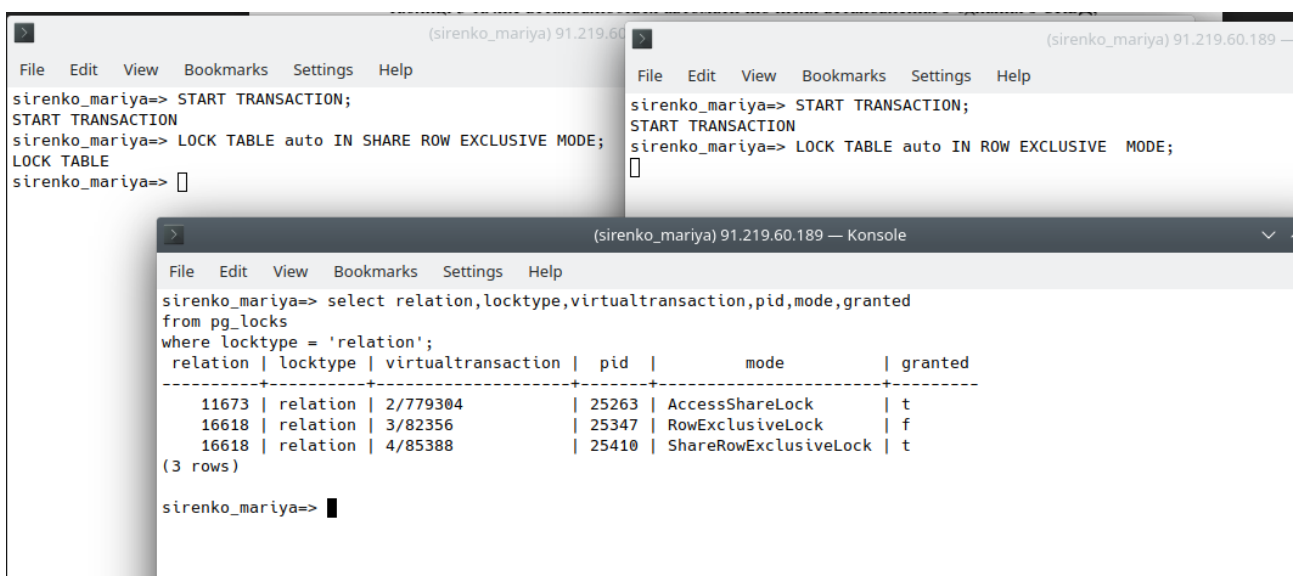
```
(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
sirenko_mariya=> START TRANSACTION
sirenko_mariya=> LOCK TABLE auto IN ROW EXCLUSIVE MODE;
LOCK TABLE
sirenko_mariya=> []

(sirenko_mariya) 91.219.60.189 — Konsole
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> LOCK TABLE auto IN ROW SHARE MODE;
LOCK TABLE
sirenko_mariya=> []

(sirenko_mariya) 91.219.60.189 — Konsole
File Edit View Bookmarks Settings Help
sirenko_mariya=> select relation,locktype,virtualtransaction,pid,mode,granted
from pg_locks
where locktype = 'relation';
relation | locktype | virtualtransaction | pid | mode | granted
-----+-----+-----+-----+-----+-----
11673 | relation | 2/779303 | 25263 | AccessShareLock | t
16618 | relation | 4/85387 | 25410 | RowExclusiveLock | t
16618 | relation | 3/82355 | 25347 | RowShareLock | t
(3 rows)

sirenko_mariya=> []
```

2) Блокування SIX-IX є не сумісними ,тому транзакція 2 чекає на кінець першої. У колонці `grated` стоїть `f` навпроти IX



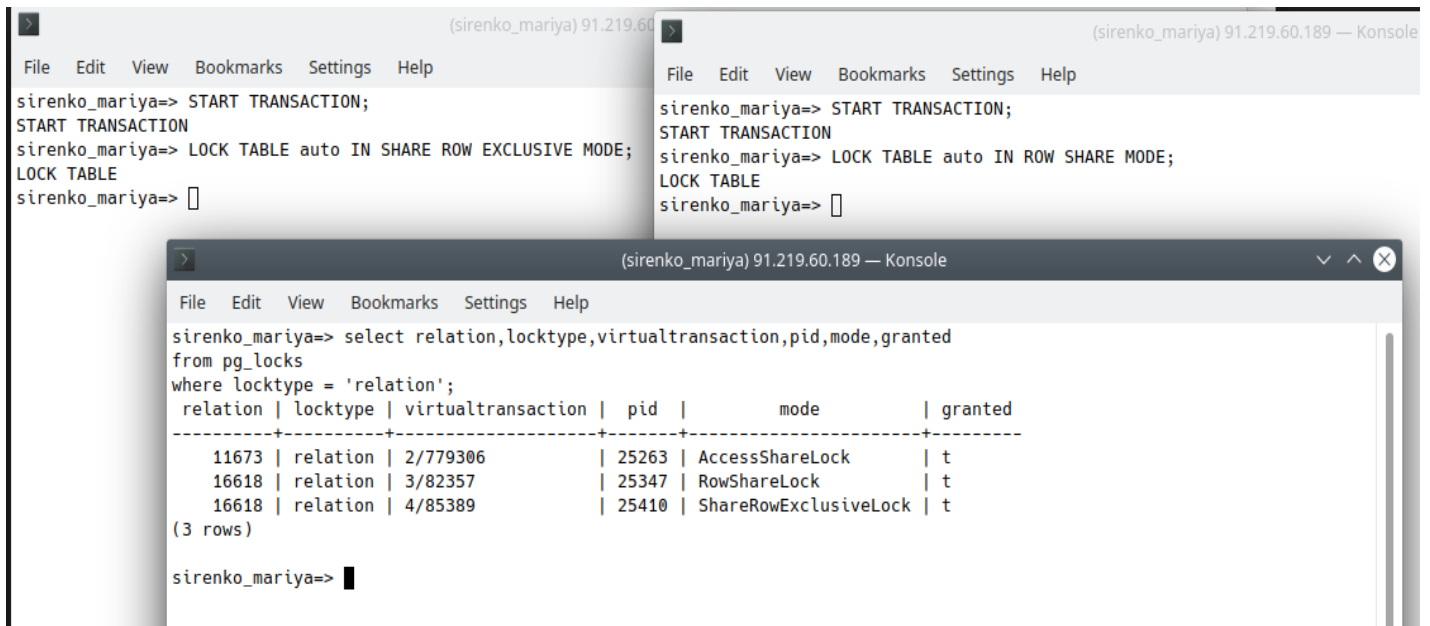
```
(sirenko_mariya) 91.219.60.189 — Konsole
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> LOCK TABLE auto IN SHARE ROW EXCLUSIVE MODE;
LOCK TABLE
sirenko_mariya=> []

(sirenko_mariya) 91.219.60.189 — Konsole
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> LOCK TABLE auto IN ROW EXCLUSIVE MODE;
LOCK TABLE
sirenko_mariya=> []

(sirenko_mariya) 91.219.60.189 — Konsole
File Edit View Bookmarks Settings Help
sirenko_mariya=> select relation,locktype,virtualtransaction,pid,mode,granted
from pg_locks
where locktype = 'relation';
relation | locktype | virtualtransaction | pid | mode | granted
-----+-----+-----+-----+-----+-----
11673 | relation | 2/779304 | 25263 | AccessShareLock | t
16618 | relation | 3/82356 | 25347 | RowExclusiveLock | f
16618 | relation | 4/85388 | 25410 | ShareRowExclusiveLock | t
(3 rows)

sirenko_mariya=> []
```

1) Блокування SIX-IS є сумісними ,тому транзакції проходять без проблем. У колонці `granted` стоїть `t`



The image shows three terminal windows from the PostgreSQL command-line interface. The top-left window shows a transaction starting and a table lock being acquired. The top-right window shows a similar transaction. The bottom window shows the output of a query to check the `pg_locks` system catalog.

```
(sirenko_mariya) 91.219.60.189 — Konsole
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> LOCK TABLE auto IN SHARE ROW EXCLUSIVE MODE;
LOCK TABLE
sirenko_mariya=> 
```

```
(sirenko_mariya) 91.219.60.189 — Konsole
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> LOCK TABLE auto IN ROW SHARE MODE;
LOCK TABLE
sirenko_mariya=> 
```

```
(sirenko_mariya) 91.219.60.189 — Konsole
File Edit View Bookmarks Settings Help
sirenko_mariya=> select relation,locktype,virtualtransaction,pid,mode,granted
from pg_locks
where locktype = 'relation';
 relation | locktype | virtualtransaction | pid | mode | granted
-----+-----+-----+-----+-----+-----
 11673 | relation | 2/779306 | 25263 | AccessShareLock | t
 16618 | relation | 3/82357 | 25347 | RowShareLock | t
 16618 | relation | 4/85389 | 25410 | ShareRowExclusiveLock | t
(3 rows)
sirenko_mariya=> 
```

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) рівень ізоляції READ COMMITTED

Реалізує безпеку від брудного читання. Транзакції бачуть лише зафіксовані зміни (командою commit), тому update 2 транзакції не бачить змінених даних. Допоки транзакція 1 не завершиться, T2 перебуває у стані WAIT

```
(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> SET TRANSACTION ISOLATION
LEVEL READ COMMITTED;
SET
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
a_id | name | year
-----+-----+-----
1 | Porche | 2021
(1 row)

sirenko_mariya=> UPDATE auto SET name = 'Mazda 3' WHERE a_id = 1;
UPDATE 1
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
a_id | name | year
-----+-----+-----
1 | Mazda 3 | 2021
(1 row)

sirenko_mariya=> COMMIT;
COMMIT
sirenko_mariya=> 
```

```
(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> SET TRANSACTION ISOLATION
LEVEL READ COMMITTED;
SET
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
a_id | name | year
-----+-----+-----
1 | Porche | 2021
(1 row)

sirenko_mariya=> UPDATE auto SET year=1999 WHERE a_id=1;
UPDATE 1
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
a_id | name | year
-----+-----+-----
1 | Mazda 3 | 1999
(1 row)

sirenko_mariya=> COMMIT;
COMMIT
sirenko_mariya=> 
```

2) рівень ізоляції REPEATABLE READ

Реалізує безпеку від неповторного читання. Коли зміни першої транзакції були зафіксовані, то REPEATABLE READ відмінює другу транзакцію з помилкою
ERROR: could not serialize access due to concurrent update
тому що, REPEATABLE READ не може змінювати данні змінені іншою транзакцією

```
(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> SET TRANSACTION ISOLATION
LEVEL REPEATABLE READ;
SET
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
a_id | name | year
-----+-----+-----
1 | Lada | 2000
(1 row)

sirenko_mariya=> UPDATE auto SET name = 'Mazda 3' WHERE a_id = 1;
UPDATE 1
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
a_id | name | year
-----+-----+-----
1 | Mazda 3 | 2000
(1 row)

sirenko_mariya=> COMMIT;
COMMIT
sirenko_mariya=> 
```

```
(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> SET TRANSACTION ISOLATION
LEVEL REPEATABLE READ;
SET
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
a_id | name | year
-----+-----+-----
1 | Lada | 2000
(1 row)

sirenko_mariya=> UPDATE auto SET year=1999 WHERE a_id=1;
ERROR: could not serialize access due to concurrent update
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
ERROR: current transaction is aborted, commands ignored until end of transaction block
sirenko_mariya=> COMMIT;
ROLLBACK
sirenko_mariya=> 
```


2) рівень ізоляції Serializable

Транзакції працюють так ,якби нічого окрім них не існує.Так як 1 транзакція вже зафіксувала свої зміни , 2 завершилася без фіксації своїх змін.Та рівні ізоляції Serializable заборонено виконувати паралельно зміни одних даних

```
(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> SET TRANSACTION ISOLATION
sirenko_mariya-> LEVEL SERIALIZABLE;
SET
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
 a_id |      name      | year
-----+-----+-----
  1   | Mazda 3        | 2000
(1 row)

sirenko_mariya=> UPDATE auto SET name = 'Lada' WHERE a_id = 1;
UPDATE 1
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
 a_id |      name      | year
-----+-----+-----
  1   | Lada           | 2000
(1 row)

sirenko_mariya=> COMMIT;
COMMIT
sirenko_mariya=> []

(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> SET TRANSACTION ISOLATION
sirenko_mariya-> LEVEL SERIALIZABLE;
SET
sirenko_mariya=> SELECT * FROM auto WHERE a_id = 1;
 a_id |      name      | year
-----+-----+-----
  1   | Mazda 3        | 2000
(1 row)

sirenko_mariya=> UPDATE auto SET year=1999 WHERE a_id=1;
ERROR:  could not serialize access due to concurrent update
sirenko_mariya=> COMMIT;
ROLLBACK
sirenko_mariya=> []
```

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.

Дві транзакції зависають і СКБД сама вирішує яку з них видалити. Транзакція 1(процес 4214) заблокована та транзакція 2 (номер 4190) заблокована(стан процесу Ss) . Кожна з них чекає на завершення попередньої ,тому отримуємо цикл.

```
(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> SELECT * FROM auto;
 a_id |      name      | year
-----+-----+-----
  2   | Lanos DEO      | 2010
  3   | Lada            | 2010
  1   | Audi            | 2000
(3 rows)

sirenko_mariya=> UPDATE auto SET name ='Mazda-3' WHERE a_id=1;
UPDATE 1
sirenko_mariya=> UPDATE auto SET name ='Mazda-3' WHERE a_id=2;
UPDATE 1
sirenko_mariya=> []

(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
sirenko_mariya=> START TRANSACTION;
START TRANSACTION
sirenko_mariya=> UPDATE auto SET year=2021 WHERE a_id=2;
UPDATE 1
sirenko_mariya=> UPDATE auto SET name='Lada' WHERE a_id=1;
ERROR:  deadlock detected
DETAIL:  Process 27503 waits for ShareLock on transaction 3013; blocked by process 27418.
Process 27418 waits for ShareLock on transaction 3014; blocked by process 27503.
HINT:  See server log for query details.
CONTEXT:  while updating tuple (0,19) in relation "auto"
sirenko_mariya=> []
```

```
[sirenko_mariya@vpsj3IeQ ~]$ ps -u postgres -o pid,ppid,stat,cmd | egrep "sirenko_mariya"
27418  8763 Ss   postgres: sirenko_mariya sirenko_mariya [local] idle in transaction
27503  8763 Ss   postgres: sirenko_mariya sirenko_mariya [local] idle in transaction (aborted)
[sirenko_mariya@vpsj3IeQ ~]$ █
```

Висновок : було отримано практичні навички з роботи з БД. Було досліджено поведінку процесів-транзакцій в базах даних та засоби керування ними через механізм блокування з використанням сучасних систем керування базами даних.