ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №2

Транзакції в СКБД PostgreSQL

Виконав:

Ст. Яцуляк Андрій

Група ПМі-31

Оцінка

Перевірила:
доц. Малець Р.Б.

2023

Тема: Вивчення понять транзакції та управління конкурентним доступом в СКБД PostgreSQL.

Мета роботи: Ознайомлення з використанням транзакцій, їх розробкою та застосуванням, рівнями ізоляцій та механізмом управління конкурентним доступом в СКБД PostgreSQL.

Завдання

Розробити базу даних для системи автоматизації шкільної бібліотеки. Система веде облік читачів, які реєструються в бібліотеці і можуть позичати книги. Для читачів зберігається прізвище, ім'я, контактні дані (адреси та телефони) та рейтинг (наскільки вчасно повертають позичені книги). Крім того база даних містить інформацію про книги, які зберігаються в бібліотеці, їх статус (доступна, кому позичена, не видається, в ремонті і т.д.) Бібліотекар може додавати нових читачів, книги і їх деталі, а також здійснювати пошук книг за авторами, видавництвами, назвами, статусом і т.д.

Хід роботи

1. Опрацював теоретичний матеріал.
2. Написав першу транзакцію, використовуючи SAVEPOINT та ROLLBACK. Приклад роботи:

Query | Query History

```
1  SELECT * FROM borrowing
2  WHERE date > '2023-03-01'
```

Data Output | Messages | Notifications

	borrowing_id [PK] integer	form_id integer	book_id integer	date date
1	10	30	19	2023-03-11
2	13	5	5	2023-04-01
3	16	44	25	2023-04-01

```sql
BEGIN;
UPDATE borrowing SET date = '2023-05-01' WHERE borrowing_id = 10 AND form_id = 30;
SAVEPOINT pnt;
UPDATE borrowing SET date = '2023-05-01' WHERE borrowing_id = 13 AND form_id = 5;
ROLLBACK TO pnt;
UPDATE borrowing SET date = '2023-05-01' WHERE borrowing_id = 16 AND form_id = 44;
COMMIT;

SELECT * FROM borrowing
WHERE date > '2023-03-01';
```

Data Output | Messages | Notifications

	borrowing_id [PK] integer	form_id integer	book_id integer	date date
1	13	5	5	2023-04-01
2	10	30	19	2023-05-01
3	16	44	25	2023-05-01

Як ми бачимо, взята книга з borrowing_id = 13 не змінила дату взяття через ролбек.

3. Написав транзакції, щоб продемонструвати як вони допомагають вирішити різні проблеми під час паралельних запитів:
    - dirty read - відбувається, коли одна транзакція зчитує дані, які були змінені, але ще не зафіксовані іншою транзакцією:

Покажу це на таблиці AUTOR:

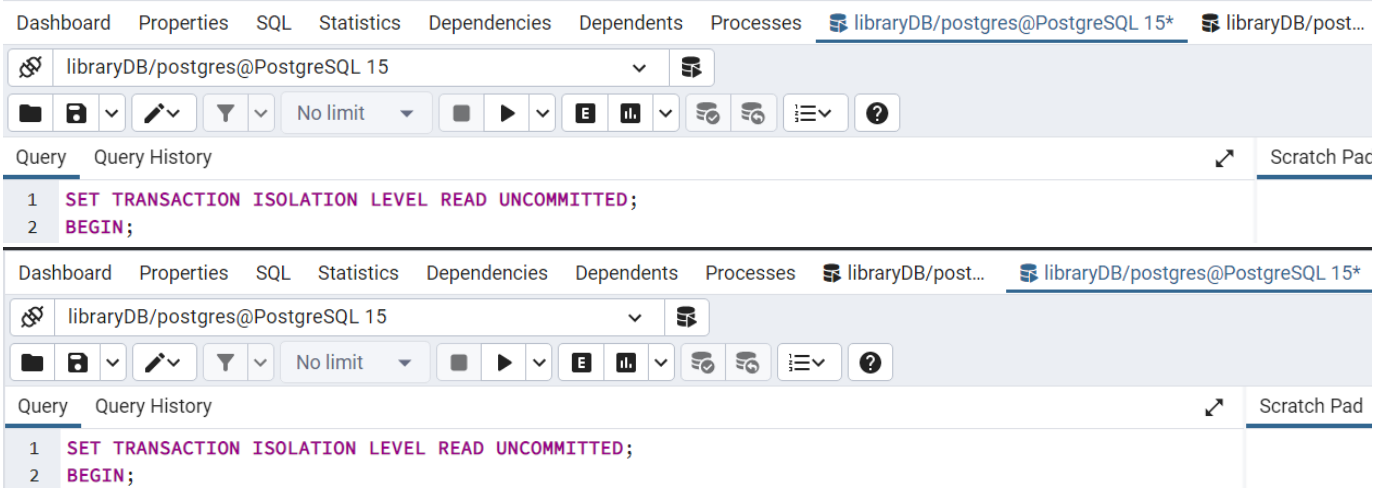
Query Query History

```
1  SELECT * FROM autor;
```

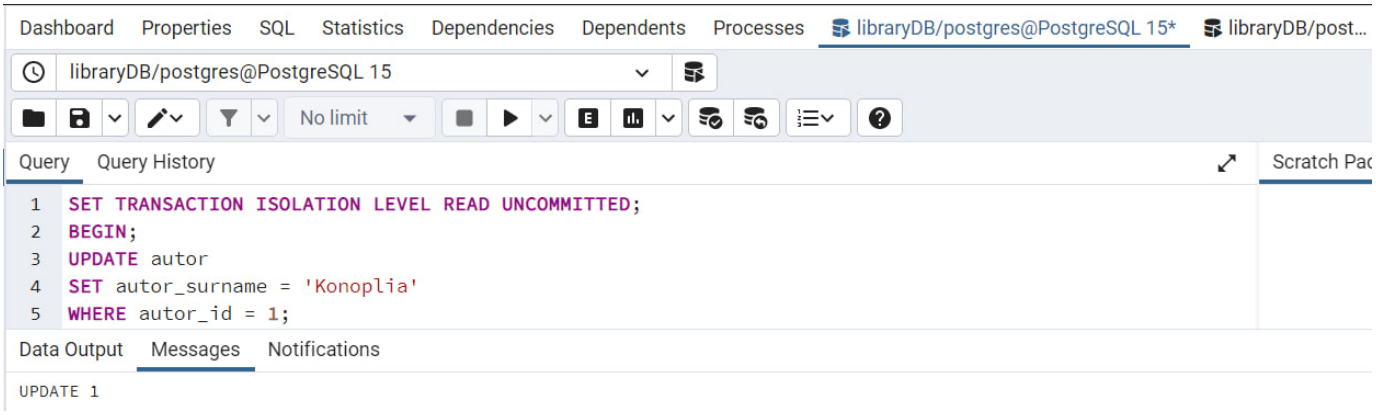
Data Output Messages Notifications

	autor_id [PK] integer	autor_name character varying (30)	autor_surname character varying (30)
1	1	Petro	Shchur
2	2	Mychailo	Dovzenko
3	3	Taras	Shevchenko
4	4	David	Shram
5	5	Adam	Davies
6	6	Dmytro	Vyshnevetskii
7	7	Mykola	Dniprovskii
8	8	Ivan	Franko
9	9	Lesia	Ukrainka
10	10	Adam	Kopernyk
11	11	Sophie	Miller
12	12	Robert	Frost
13	13	Maya	Angelou
14	14	David	Rowling

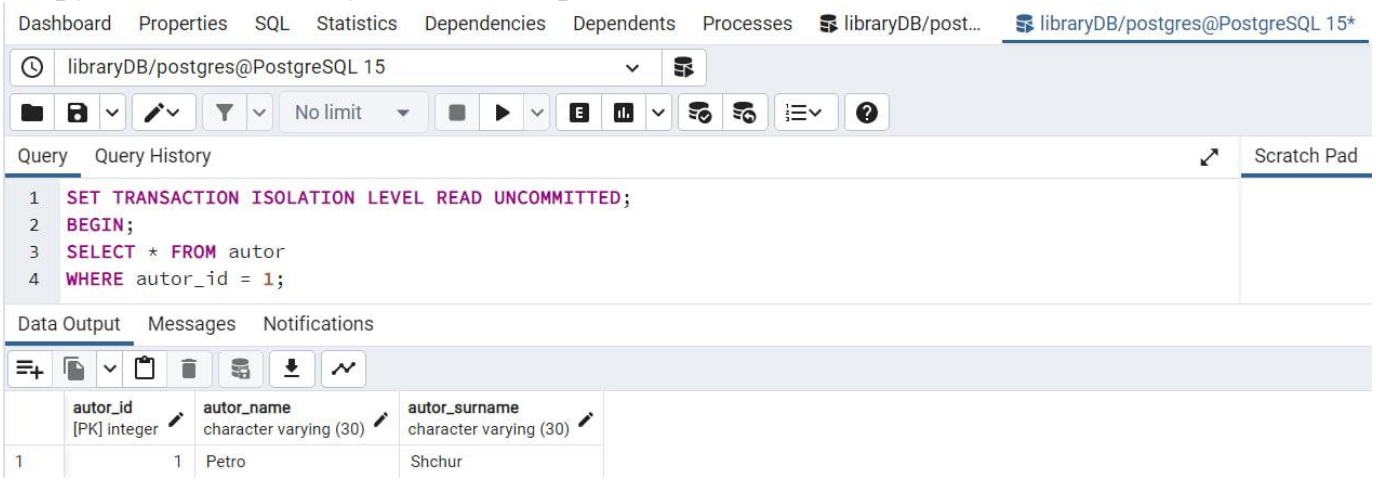
В середовищі pgAdmin для своєї бази даних відкрив дві вкладки Query Tool. В обох вказав рівень ізоляції ‘Read Uncommitted’:



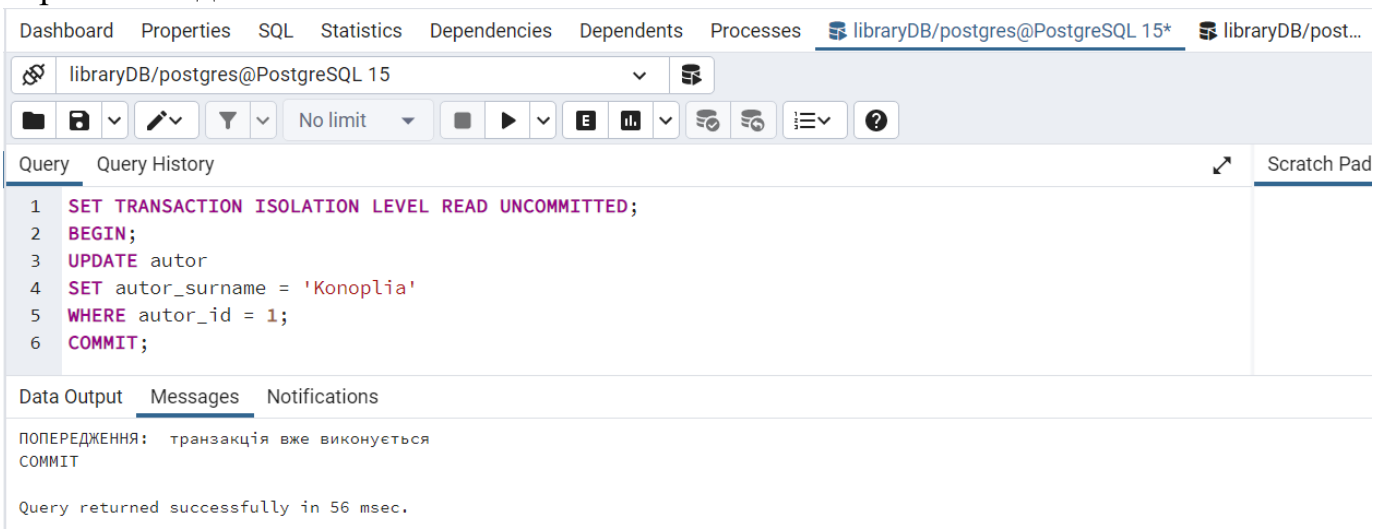
В першій вкладці оновлюю рядок, але не підтверджую



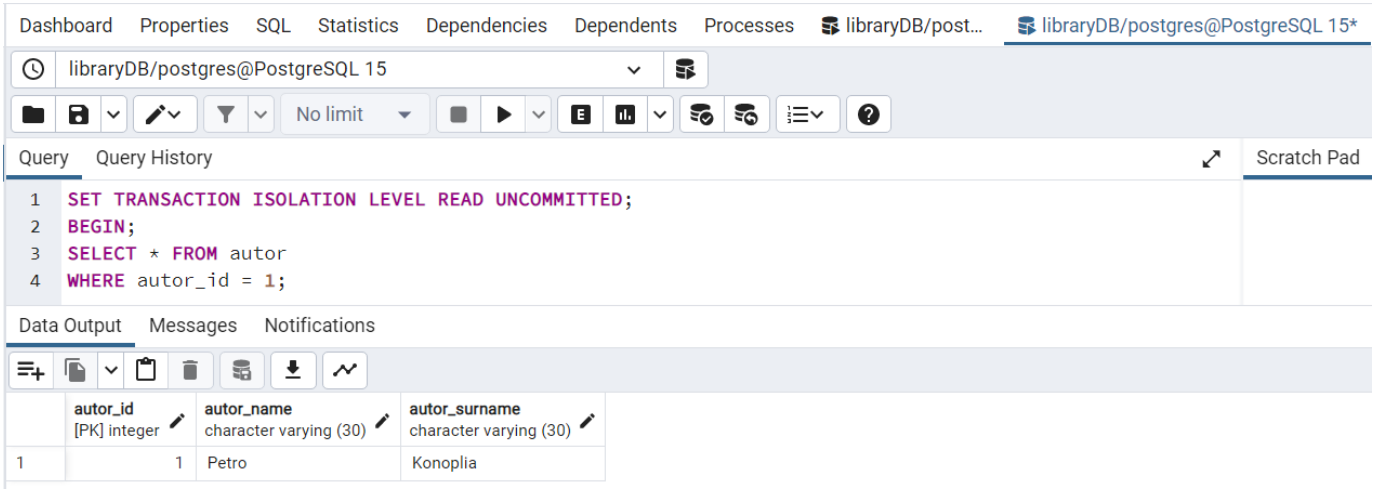
В другій вкладці пишу SELECT вираз



Отже, друга вкладка не помітила незафіксованої зміни. Дописую COMMIT до першої вкладки



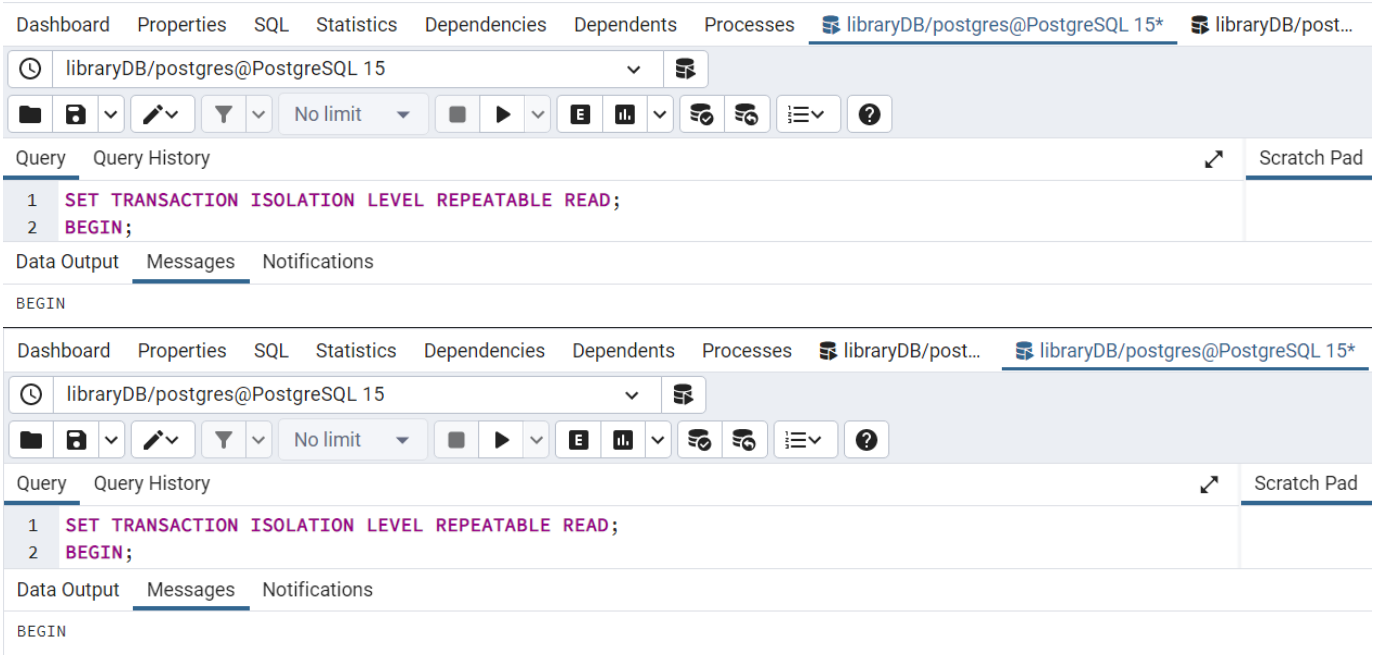
і запускаю аналогічний запит у 2-й



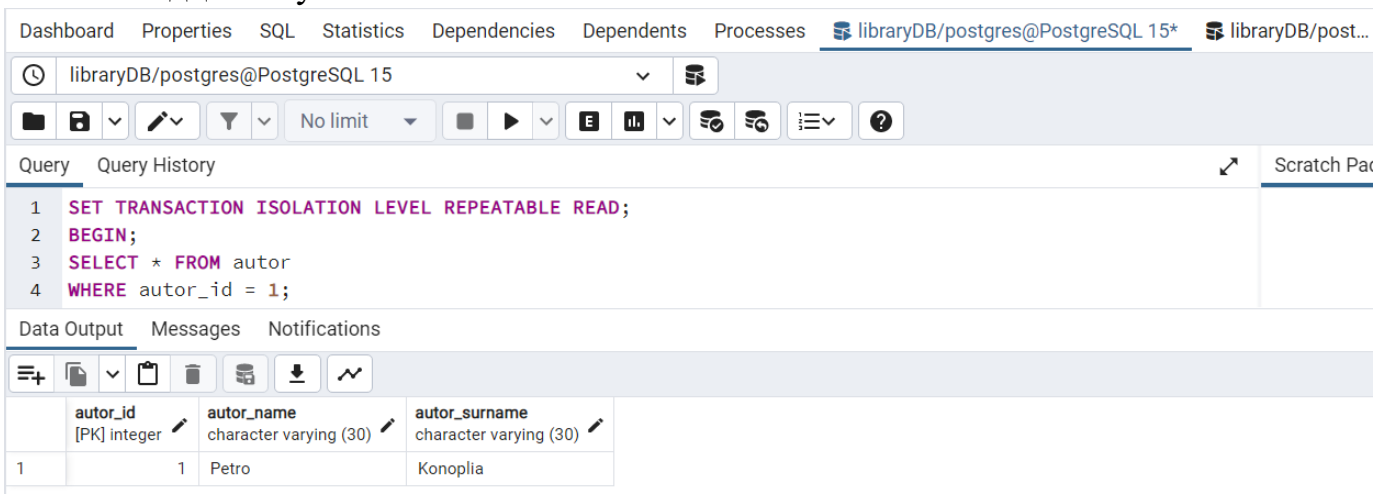
Як ми бачимо, зміни відбулися у другій вкладці

- Nonrepeatable Read - відбувається, коли транзакція зчитує ті самі дані кілька разів, але отримує різні результати, оскільки інша транзакція змінила дані між ними:

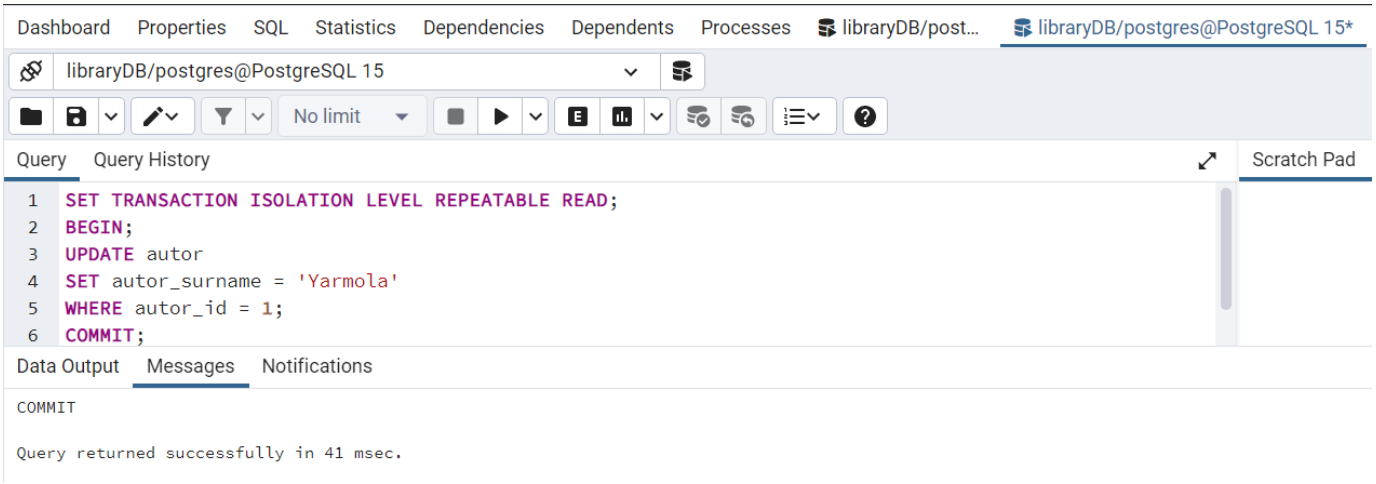
В обох вкладках встановив рівень ізоляції ‘Repeatable read’



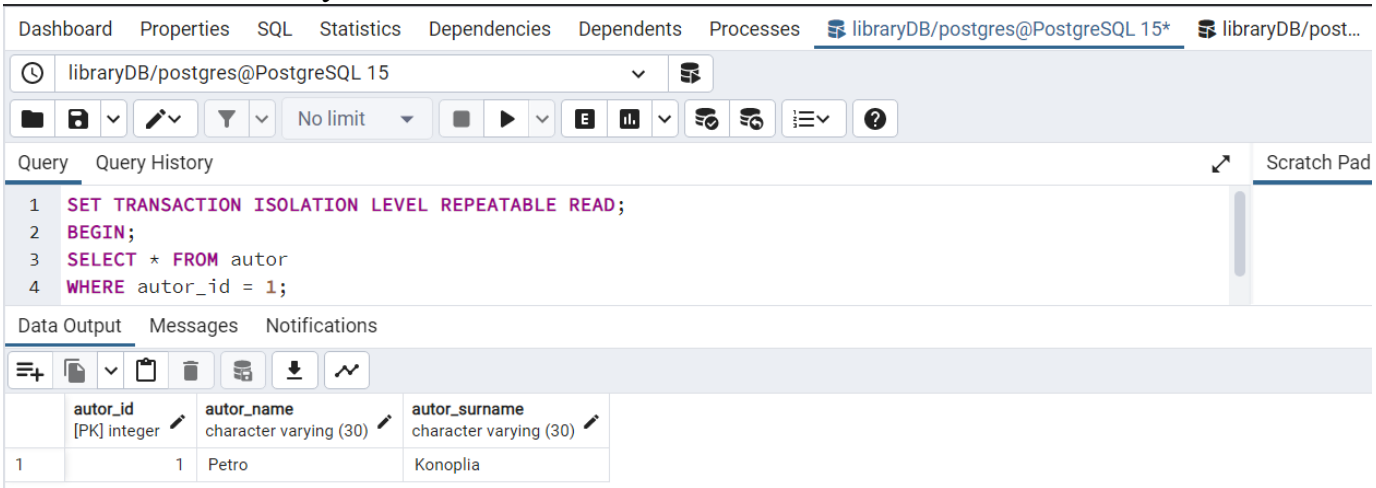
В 1-й вкладці пишу SELECT запит



В 2-й вкладці оновлюю той самий рядок



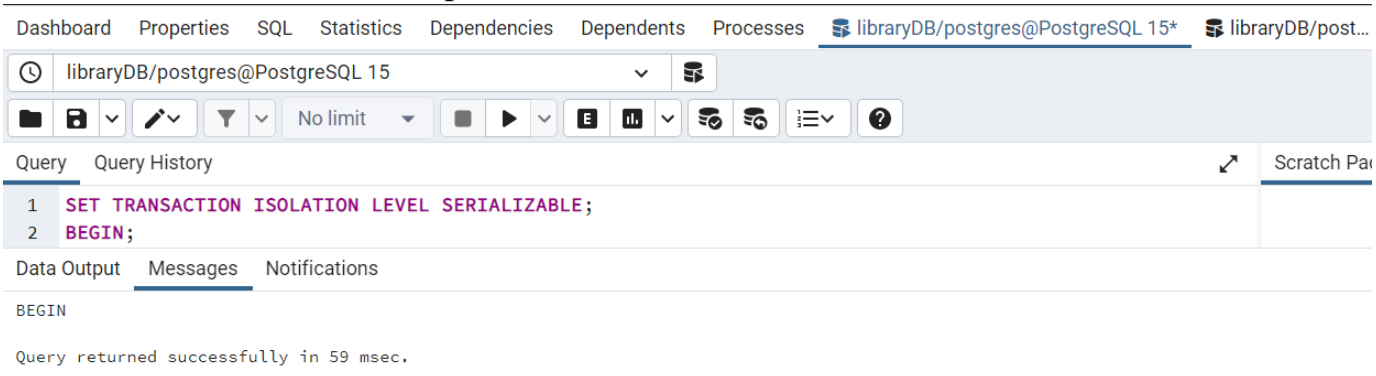
В 1-й вкладці виконую той самий SELECT запит

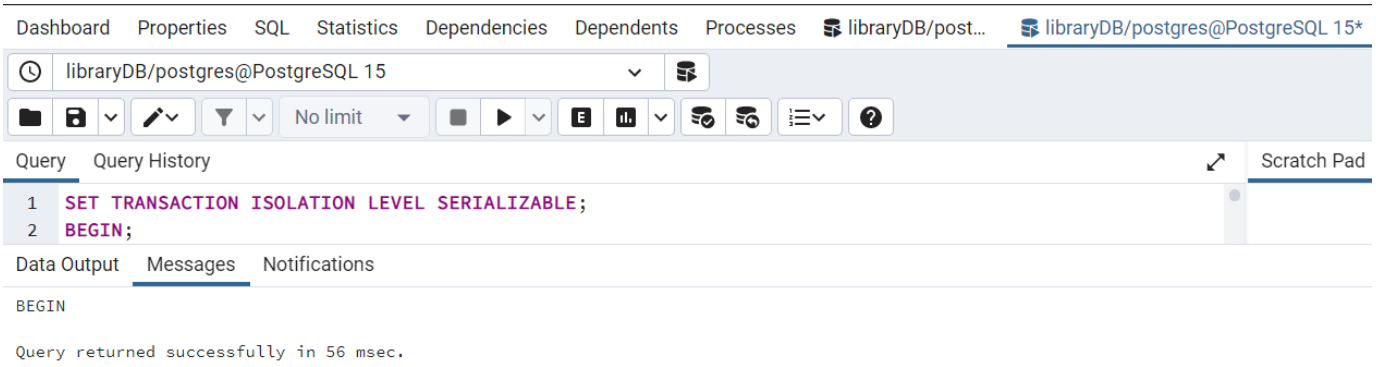


Отже, дані не змінилися. Це демонструє, що ‘Repeatable read’ запобігає ‘Nonrepeatable Read’, надаючи послідовний порядок даних протягом усієї транзакції, гарантуючи, що та сама операція читання дає той самий результат, навіть якщо інші транзакції змінюють дані.

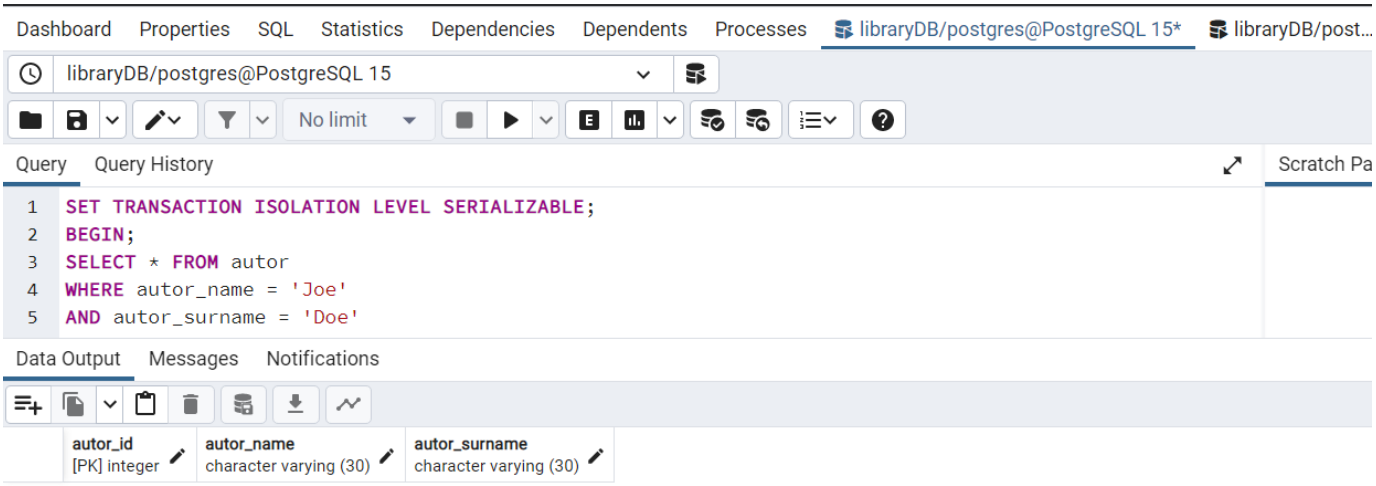
- Phantom read - відбувається, коли транзакція читає набір рядків, які задовольняють умову, але інша транзакція вставляє або видаляє рядки, які також задовольняють цю умову між читаннями.

В обох вкладках встановив рівень ізоляції ‘Serializable’

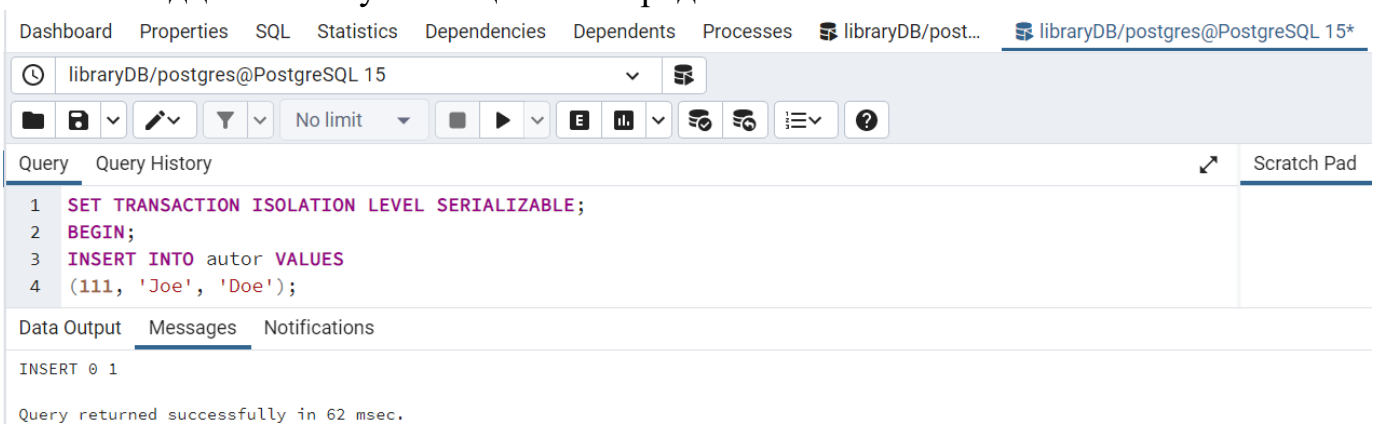




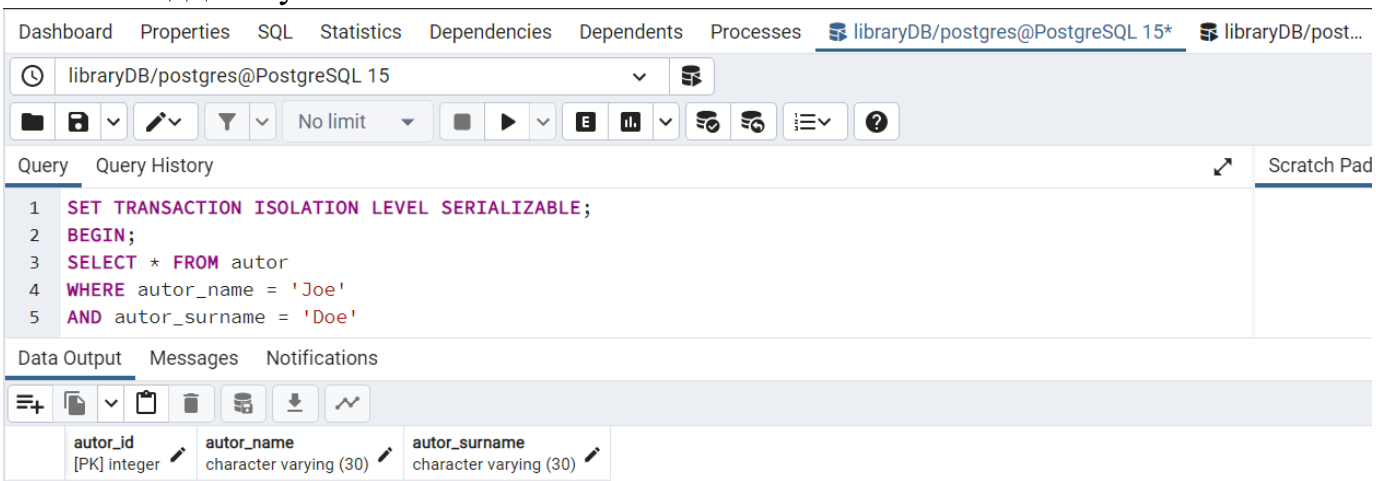
В 1-й вкладці написав SELECT запит



В 2-й вкладці вставив у таблицю новий рядок



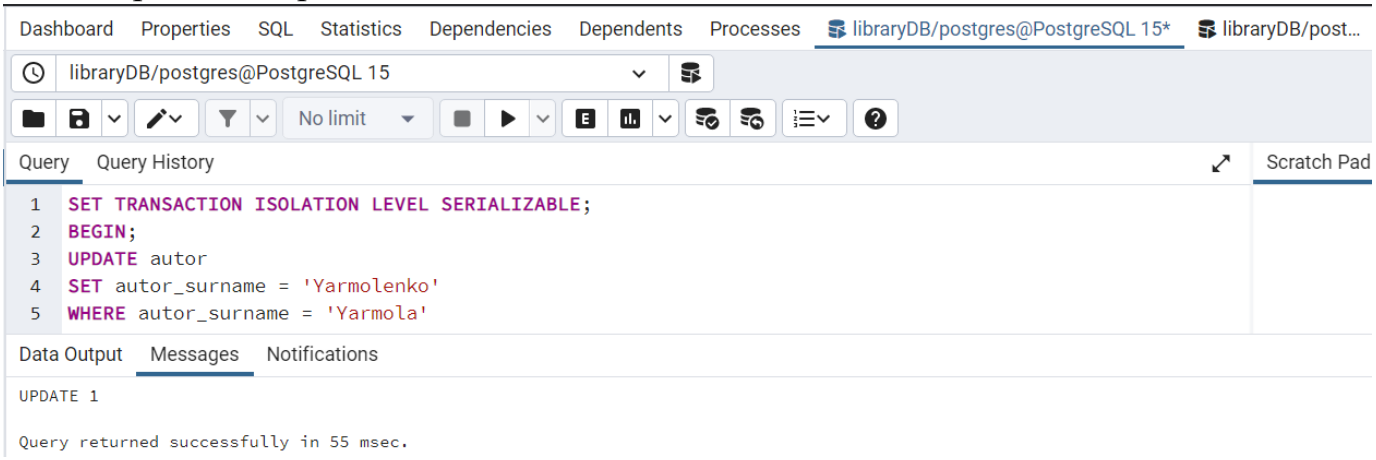
В 1-й вкладці запускаю аналогічний запит SELECT



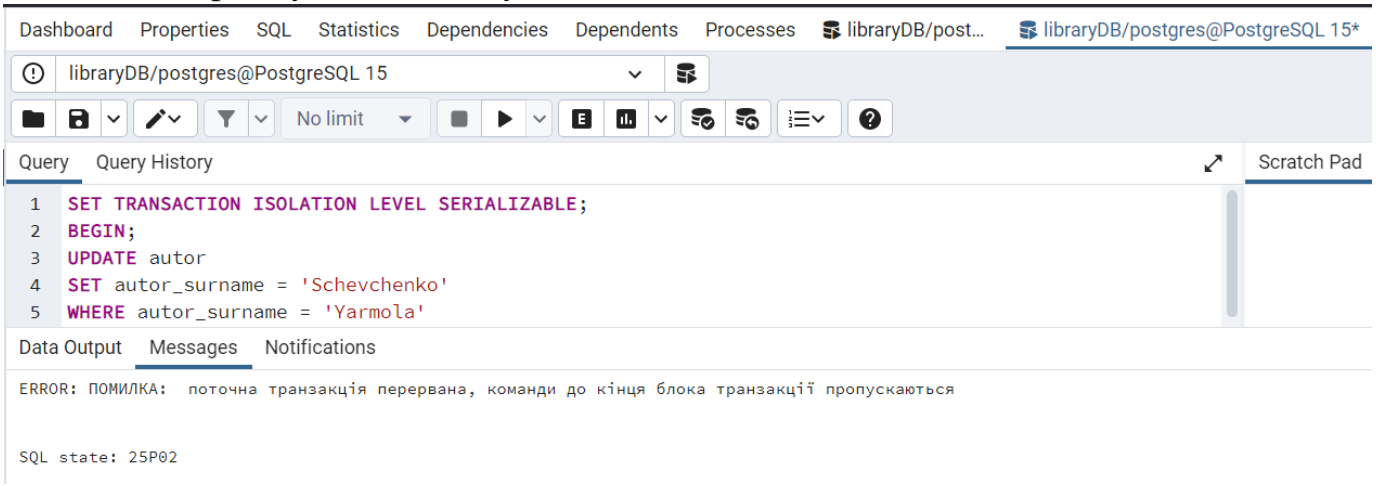
Нічого не змінилося, тому що ‘Serializable’ запобігає явищу ‘Phantom Read’.

- Serialization anomaly - виникає, коли транзакції, які здаються серіалізованими, дають результати, які неможливі під час послідовного виконання тих самих транзакцій:

Оновив рядок в першій вкладці



Написав конфліктуючий запит у 2-й вкладці



Цей запит не виконався, оскільки ‘Serializable’ запобігає явищу ‘Serialization Anomaly’.

Висновок: під час виконання лабораторно роботи я ознайомився з використанням транзакцій, їх розробкою та застосуванням, рівнями ізоляцій та механізмом управління конкурентним доступом в СКБД PostgreSQL.