**Лабораторна робота 19.**

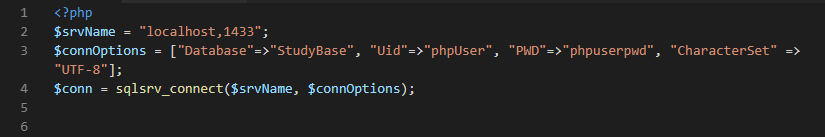
**Управління транзакціями та обробка помилок.**

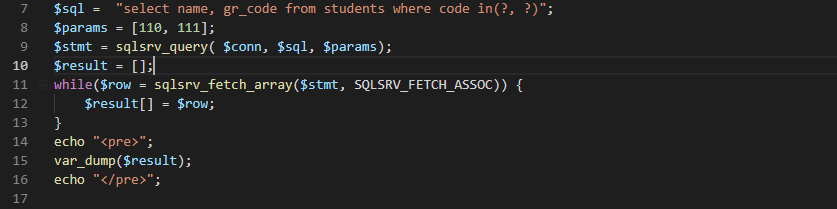
**Мета роботи:** Навчитись починати та завершувати транзакцію з боку клієнта та обробляти помилки, що повертаються від серверу БД.

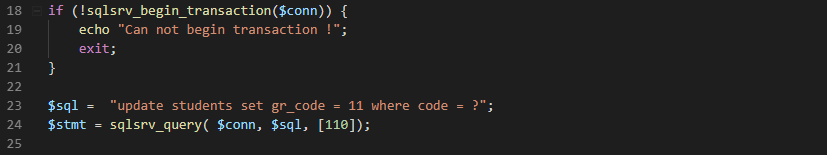
**Задачі:**

* Виконати декілька команд із клієнтського застосунку у межах однієї транзакції з наступною фіксацією та відкатом
* Із використанням команд управління транзакціями виконати імітацію «мертвого» блокування
* Обробити помилку, що може повернутись сервером у ході виконання команди
* Виконання індивідуального завдання згідно варіанту

1. У папці з проектом створимо файл «test-tran.php» та помістимо до нього код, що виконує підключення до БД та виконання декількох команд зміни даних у межах однієї транзакції (рис.1). Також у коді виконується виведення даних перед початком та після завершення транзакції.









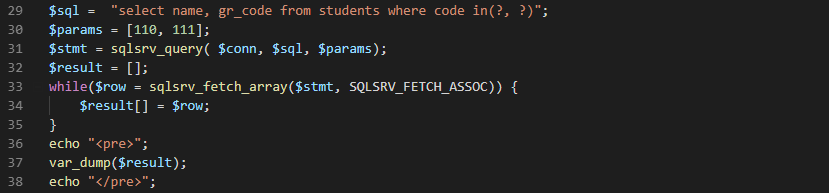




Рис.1. Вміст файлу «test-tran.php»

Далі наведемо вивід у вікно браузера при виконанні даного коду (http://sql-server/test-tran.php) до початку (рис.2) та після завершення (рис.3) транзакції, у межах якої виконувались операції зміни даних.

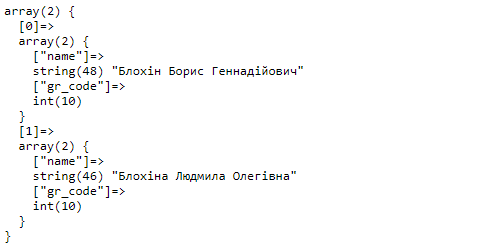


Рис.2. Вибірка до початку транзакції

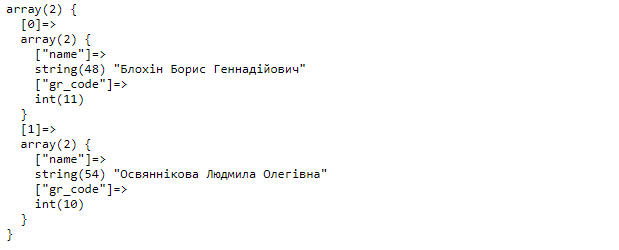


Рис.3. Вибірка після фіксації транзакції

Як бачимо, всі зміни були успішно виконані.

1. Тепер виконаємо ті ж дії, але із наступним відкатом транзакції. Також додамо ще одне звернення до даних в середині транзакції. Оскільки ми будемо виконувати одну і ту ж вибірку 3 рази, виділимо її в окрему функцію (рис.4).

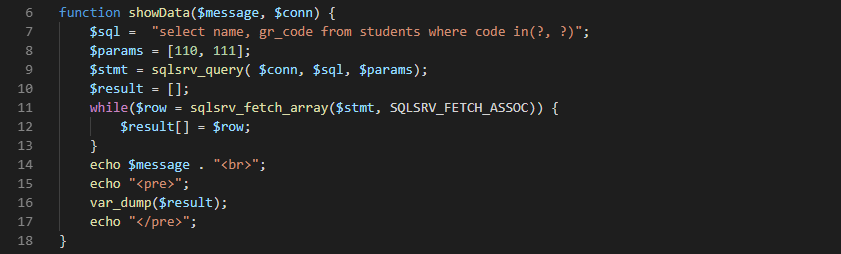


Рис.4. Функція для вибірки даних.

Тепер виконаємо самі зміни із наступним відкатом транзакції (рис.5)

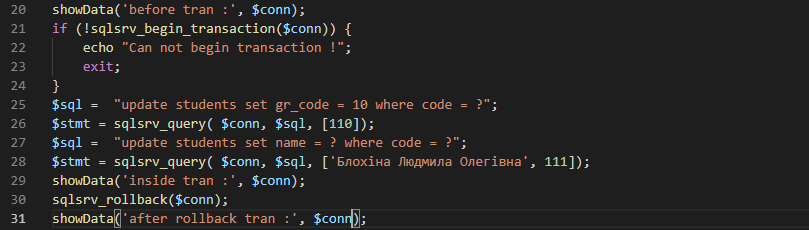


Рис.5. Виконання зміни даних із відкатом транзакції

Та наведемо результат виведення даних до початку (рис.6), всередині (рис.7) та після відкату транзакції (рис.8)

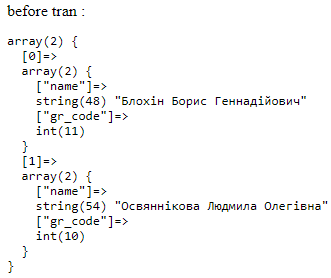


Рис.6. Вибірка до початку транзакції

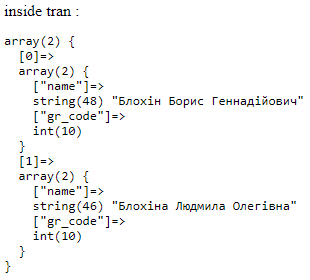


Рис.7. Вибірка всередині транзакції

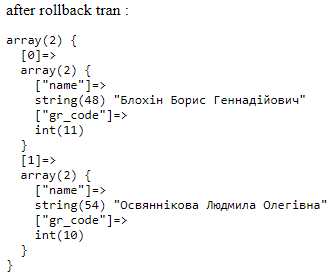


Рис.8. Вибірка після відкату транзакції

1. Поєднаємо 2 попередні приклади. Виконаємо 2 команди зміни у межах однієї транзакції, і якщо обидві завершаться успішно, зафіксуємо транзакцію. Якщо хоча б одна команда завершиться із помилкою, зробимо відкат транзакції (рис.9). Для забезпечення неуспішного виконання однієї з команд зробимо помилку у назві таблиці.

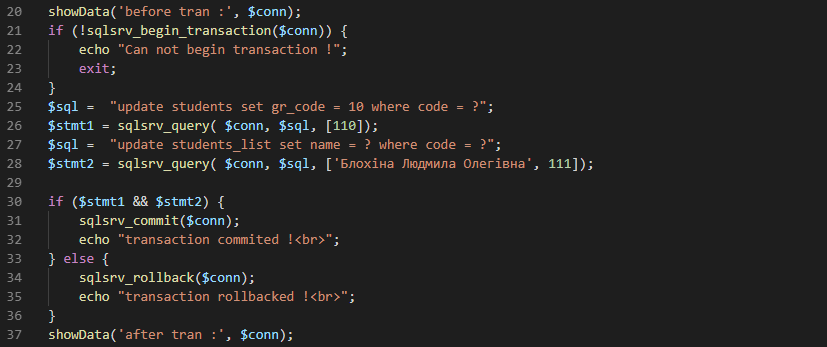


Рис.9. Фіксація або відкат транзакції при виконанні певних умов

Наведемо результат у браузері (рис.10)

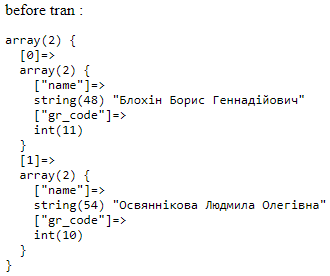
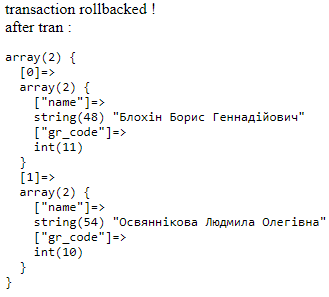
 

Рис.10. Результат виконання коду на рис.9.

Як і планувалось, транзакція була відкочена, оскільки 2-га команда завершується помилкою. Бачимо що не було застосовано також змін, що виконані першою командою. Для порівняння виконаємо той самий код із успішною та неуспішною командою, але без використання команд управління транзакціями (рис.11) та наведемо результат їх роботи (рис.12)

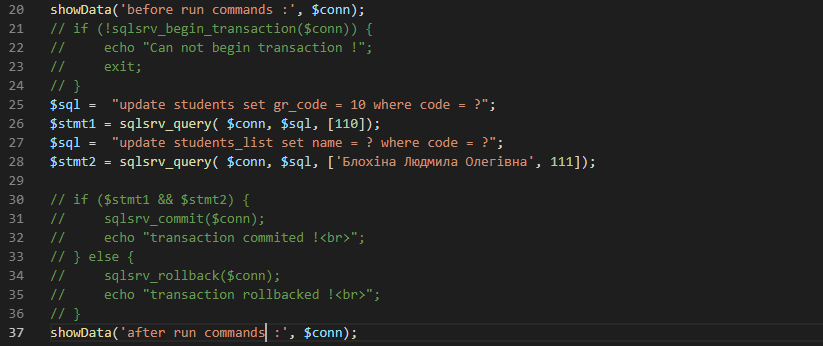


Рис.11. Послідовне виконання правильної та помилкової команди зміни даних

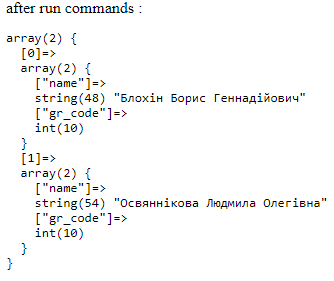
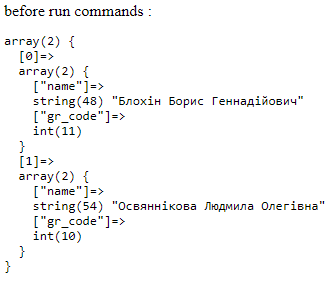
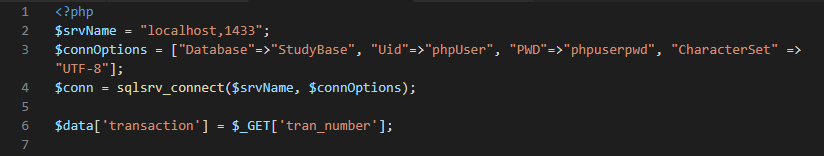
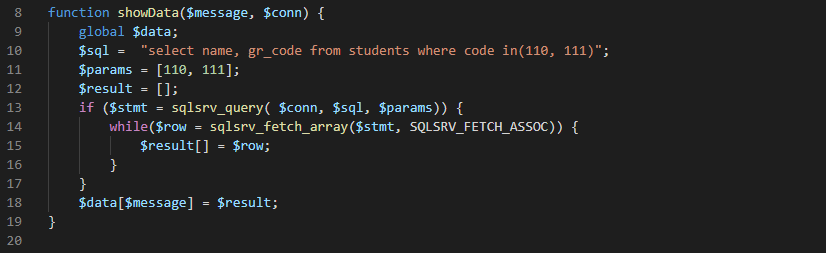


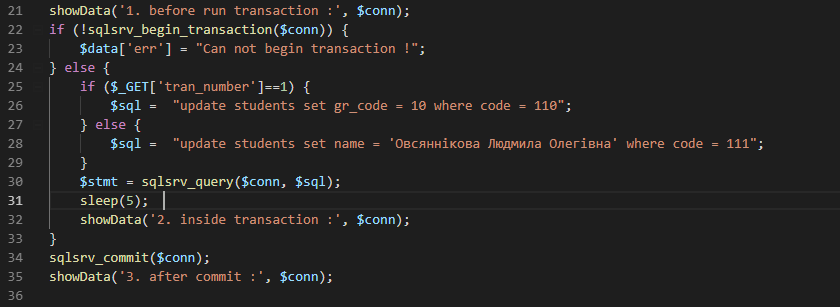
Рис.12. Результат роботи коду рис.11.

Як бачимо, зміни, виконані першою командою, успішно застосовано, не зважаючи на помилковість другої.

1. Володіючи командами управління транзакціями з боку клієнта, спробуємо штучно створити на клієнті «мертве» блокування, як ми це робили у попередніх роботах в SSMS. Для цього нам знадобиться 2 з’єднання із БД, та послідовне виконання команд зміни та вибірки даних, що перетинаються між собою у межах однієї транзакції. Реалізуємо PHP код, що буде виконувати зміну, а через невеликий таймаут читання даних. Будемо передавати у нього номер транзакції (1 чи 2) і в залежності від цього виконувати різні команди зміни даних (рис.13)







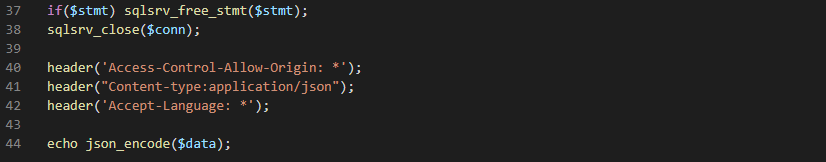
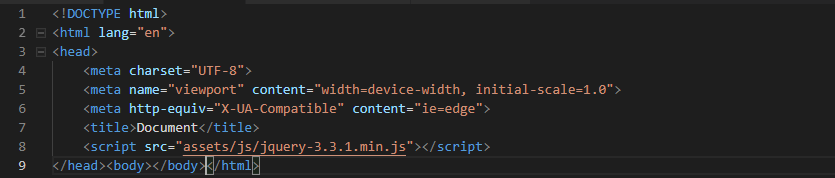


Рис.13. PHP файл «test-deadlock.php» для створення «мертвого» блокування

Тепер створимо deadlock.html, що ініціює паралельне виконання двох транзакцій (рис.14)



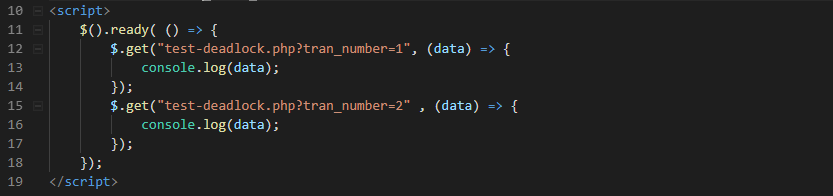


Рис.14. Виконання двох паралельних транзакцій

Запускаємо браузер та набираємо «http://sql-server/deadlock.html» (рис.15).



Рис.15. Запускаємо deadlock.html

В режимі розробника у браузері переглянемо результат, що повертається на консоль (через 5 секунд) (рис.16)

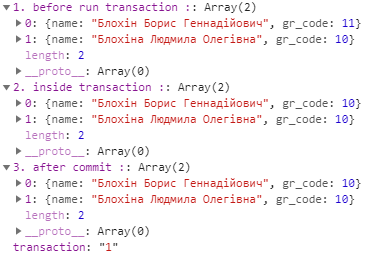
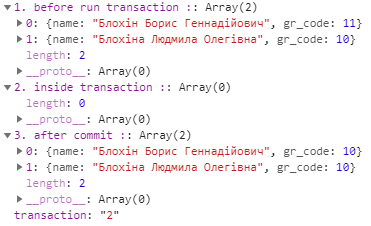
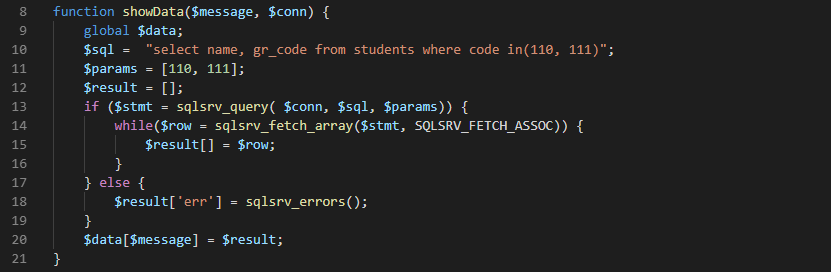
 

Рис.16. Результат роботи 2-х паралельних транзакцій

Бачимо, що зміни, які намагається виконати друга транзакція, не застосовуються. Крім того, можна побачити, що всередині другою транзакції не відпрацьовує вибірка даних. Це відбувається через відкат транзакції 2 сервером БД через виникнення мертвого блокування. Але бажано отримати від сервера якесь повідомлення та обробити його відповідним чином. Виконаємо цю обробку в test-deadlock.php (рис.17) та повернемо клієнту (deadlock.html) текст помилки (рис.18)



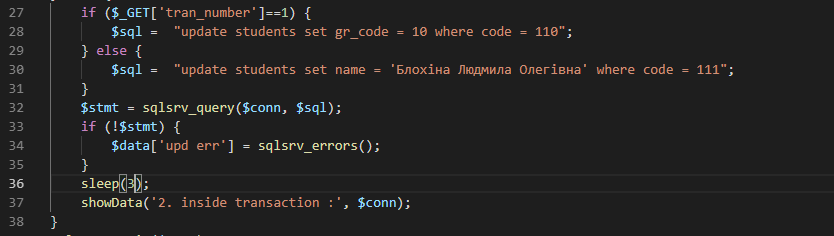


Рис.17. Обробка помилки виконання запиту

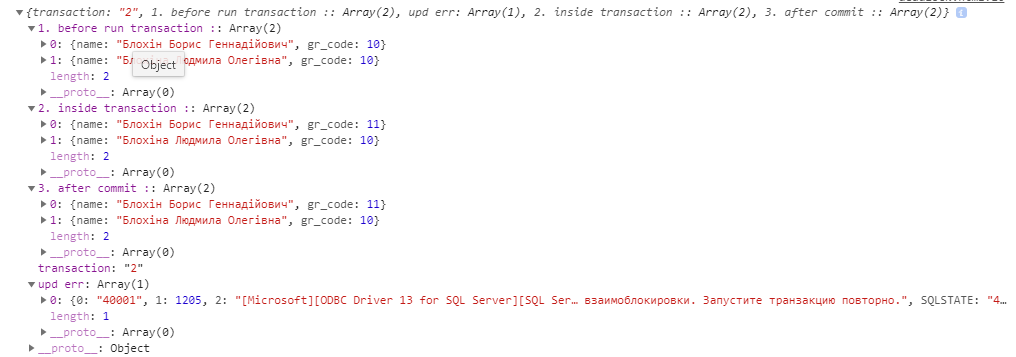


Рис.18. Інформація щодо 2-ї транзакції, що була відкочена сервером на етапі зміни даних

Як бачимо, транзакція була відкочена на етапі зміни даних, якщо зменшити таймаут між операціями зміни та читання, помилка буде виникати на етапі читання (рис.19)

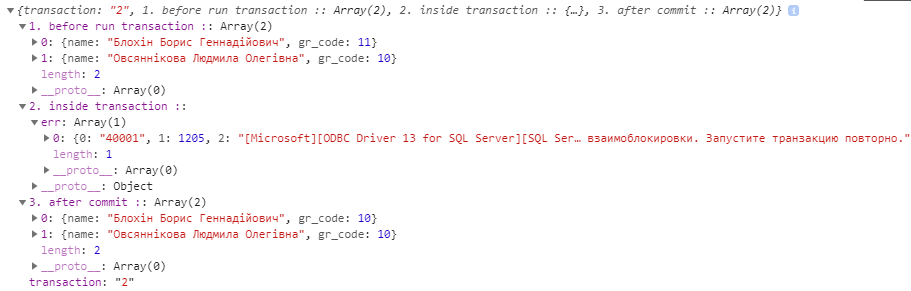


Рис.19. Інформація щодо 2-ї транзакції, що була відкочена сервером на етапі читання даних

1. Кінець роботи

**Завдання для індивідуального виконання.**

Для БД свого варіанту виконати в клієнтському застосунку декілька команд модифікації даних у межах однієї транзакції. У випадку невиконання однієї з команд виконувати відкат транзакції. Продемонструвати варіант успішного та неуспішного виконання транзакції.

Організувати обробку помилок виконання запиту та повернення користувачу повідомлення про неможливість виконання операції із наведенням помилки, що отримана від серверу БД.

Додатково. У власній БД із використанням клієнтського застосунку створити «мертве» блокування.