**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **ПЗ**

### ЗВІТ

до лабораторної роботи № 8

**На тему:** *“Транзакції і обробка помилок”*

**З дисципліни:** *“Бази даних”*

**Лектор:**

доцент каф. ПЗ

Павич Н.Я.

**Виконав:**

ст. гр. ПЗ-36

Матвіїв Т.Т.

**Прийняв:**

асист. каф. ПЗ

Ковела І.О.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2021 р.

∑= \_\_\_\_ .

Львів – 2021

**Тема роботи:** транзакції і обробка помилок.

**Мета роботи:** познайомитись з поняттям транзакції, навчитись планувати і керувати транзакціями для різних бізнес-процесів і навчитись відловлювати помилки під час транзакцій.

**Теоретичні відомості**

Транзакція є послідовністю операцій, що виконуються як одна логічна одиниця роботи. Логічна одиниця роботи повинна володіти чотирма властивостями: атомарністю, узгодженістю, ізоляцією і тривалістю (ACID), щоб називатися транзакцією. SQL Server надає різні способи обробки транзакцій, які можна визначити для кожного з'єднання з базою даних. Будь-яке з'єднання може використовувати той режим, який необхідний для виконання специфічних для цього з'єднання вимог.

Програмісти відповідальні за початок і завершення транзакцій в точках, де забезпечується логічна цілісність даних. Програміст повинен визначити послідовність змін даних, які залишають дані в цілісному стані по відношенню до ділових правилами організації. Програміст включає ці інструкції модифікації в одну транзакцію, щоб SQL Server Database Engine зміг забезпечити фізичну цілісність транзакції.

Екземпляр компоненти Database Engine відповідальний за забезпечення механізмів, що гарантують фізичну цілісність кожної транзакції. Компонента Database Engine забезпечує наступне:

• блокуючі засоби, які зберігають ізоляцію транзакцій.

• реєструючі засоби, які гарантують тривалість транзакції. Навіть якщо в обладнанні сервера, операційній системі або екземплярі Database Engine відбудеться збій, після перезапуску Database Engine використовує журнали транзакцій для автоматичного відкату будь-яких незавершених транзакцій до моменту збою системи.

• функції управління транзакціями, які реалізують атомарність і узгодженість транзакції. Після початку транзакції вона повинна бути успішно завершена, інакше екземпляр компоненти Database Engine скасовує всі зміни даних, зроблені з початку транзакції.

Управління транзакціями реалізується, головним чином, шляхом зазначення того, коли транзакція починається і закінчується. Це можна вказати або за допомогою інструкцій мови Transact-SQL, або використовуючи функції інтерфейсу прикладного програмування (API) для баз даних. У системі також повинна бути можливість правильної обробки помилок, що переривають транзакцію до її закінчення.

Транзакції в екземплярі компонента SQL Server Database Engine можна запускати як явні, автоматично фіксовані або неявні.

**Індивідуальне завдання**

1. Для кращого засвоєння виконайте всі вправи.

2. Визначте межі транзакцій для власної бази даних. Зауважимо, що розробник повинен визначити, де починається і де закінчується транзакція. Транзакція завжди повинна бути настільки короткою, наскільки це можливо, але її тривалість повинна бути достатньою, щоб транзакцію можна було зіставити вимогам бізнес-процесу.

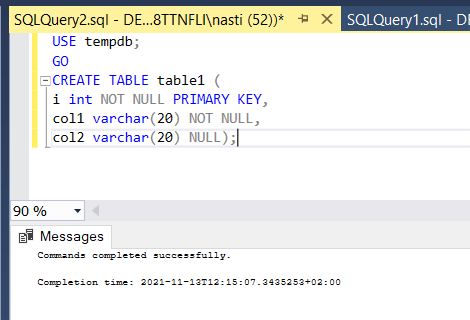
3. Розробіть механізм управління помилками у власній базі даних. Зауважимо, що не всі помилки викликають автоматичний відкат транзакцій. Розробка управління помилками є обов'язком розробника.

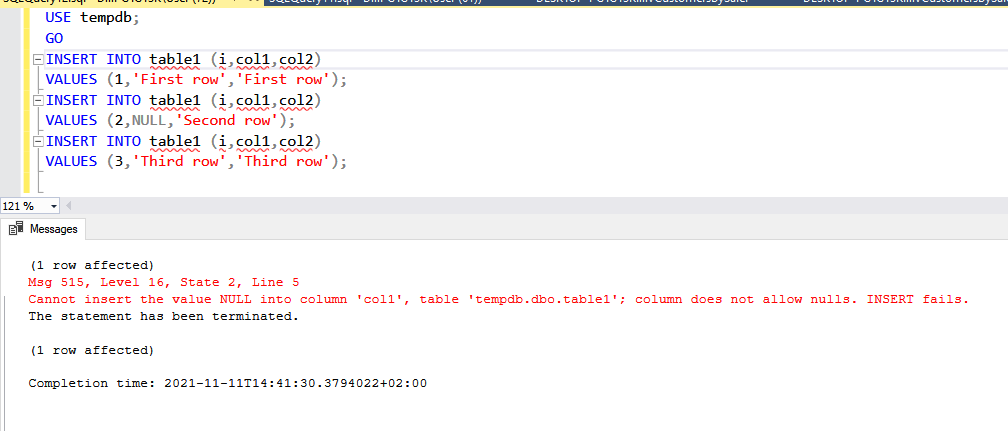
4. Визначте рівень ізольованості транзакції для вашої бази даних. Зауважимо, що ізольованість різних транзакцій завжди має негативні наслідки для паралелізму. Якщо ви повністю ізолюєте яку-небудь транзакцію, а іншій транзакції потрібно буде прочитати ті ж дані, але в попередньому стані, то друга транзакція буде блокована до завершення першої. Це може стати великою проблемою для систем баз даних з безліччю паралельних підключень. SQL Server реалізує різні варіанти рівнів ізольованості, які слід вибирати дуже ретельно. За загальним правилом, вибирати слід той рівень, який блокує якнайменший об'єм даних на найкоротший проміжок часу, але при цьому забезпечує необхідний рівень захищеності транзакції.

5. Продемонструйте на власній базі даних моніторинг блокувань.

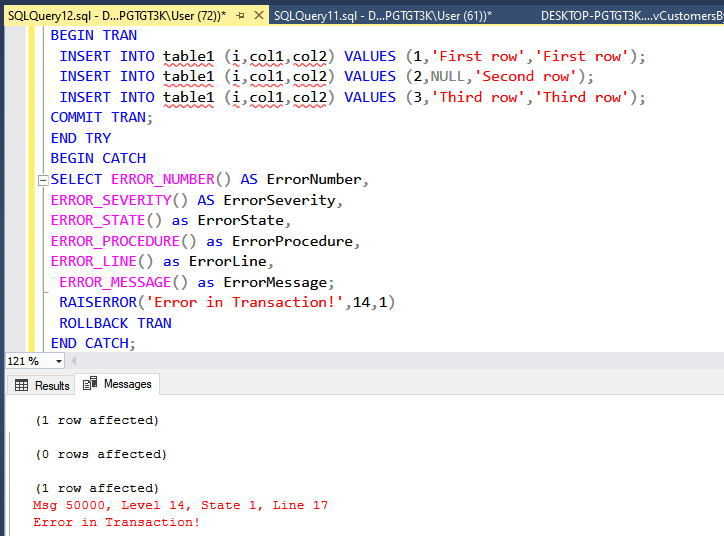
**Хід роботи**

1. Виконую вправи подані в теоретичних відомостях.

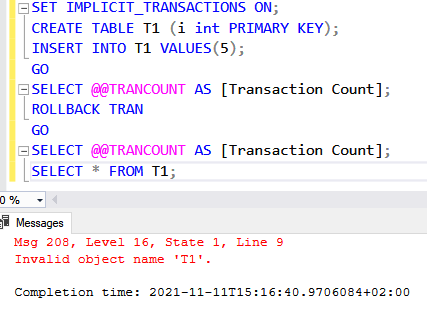




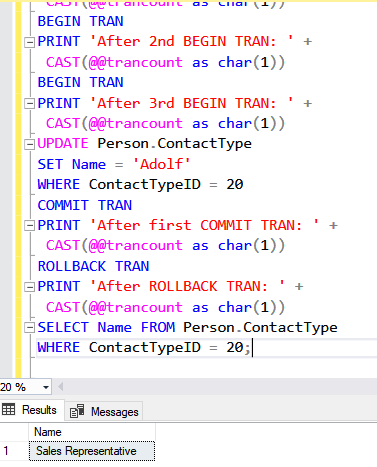
*Рис.1-2. Кожен запит на вставку рядка розглядається як окрема транзакція з автофіксацією, тому другий запит автоматично виконує відкат. Інші інструкції автоматично зафіксувалися*



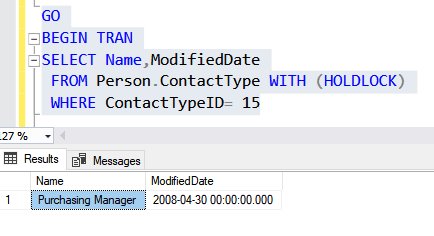
*Рис.3. Виконання явної транзакції з відкатом при помилці в одній з інструкцій, що міститься в ній*



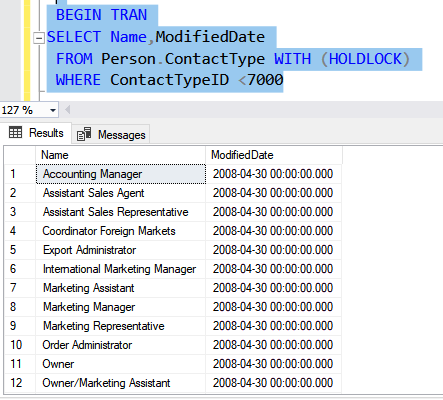
*Рис.4 Таблиці більше не існує, адже ми зробили відкат неявної транзакції*



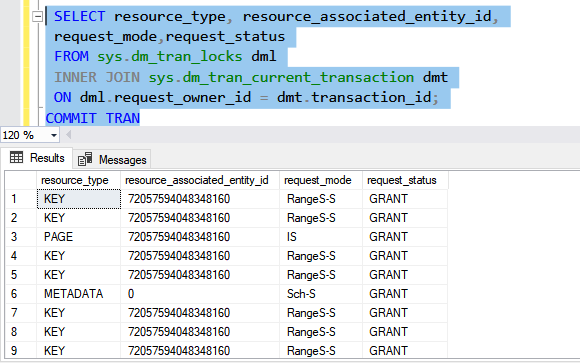
*Рис. 5 Перевіряю, що відбудеться у разі відкату транзакції*

**

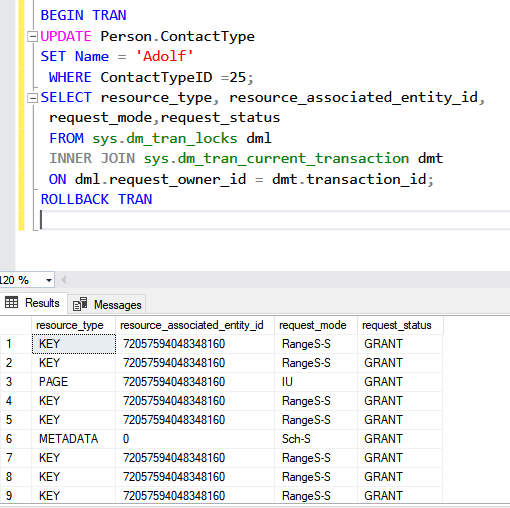
*Рис. 6 Запускаємо транзакцію і виконуємо запит до таблиці*

**

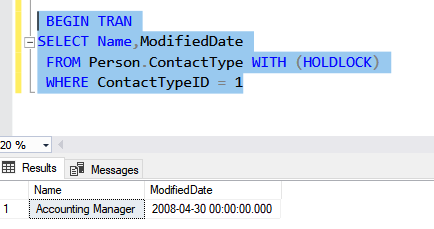
*Рис. 7 Змінюємо фразу WHERE так, щоб запит повертав більше рядків*

**

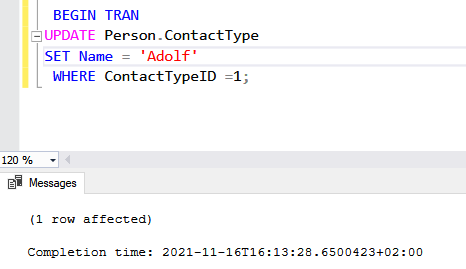
*Рис. 8 Вивчимо блокування. Видно, що роздільне блокування визначене для об'єктного типу ресурсів, в цьому випадку, для таблиці Person.Contact.*

**

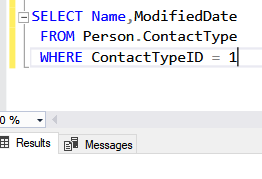
*Рис. 9 Видно, що SQL Server блокує ключ за допомогою монопольного блокування (request\_mode = X).*

**

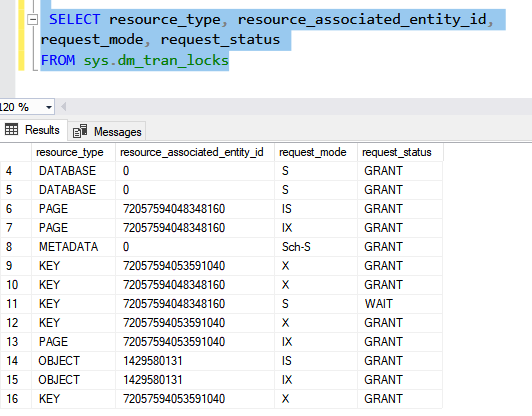
*Рис. 10 Створюємо запит на отримання імя та дати*



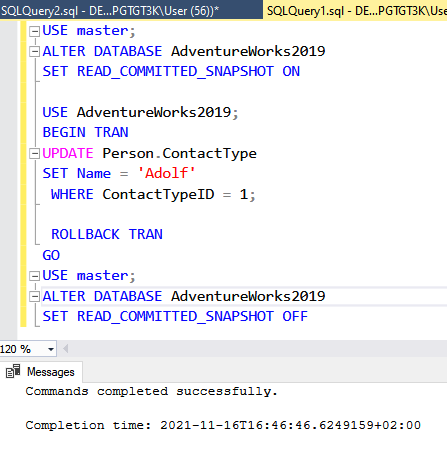
*Рис.11 У новому вікні змінюємо імя*

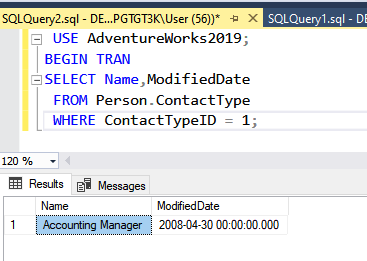


*Рис12 Повертаємся до минулого вікна і пробуємо знову виконати запит. Він не завершується оскільки інструкція SELECT заблокована*

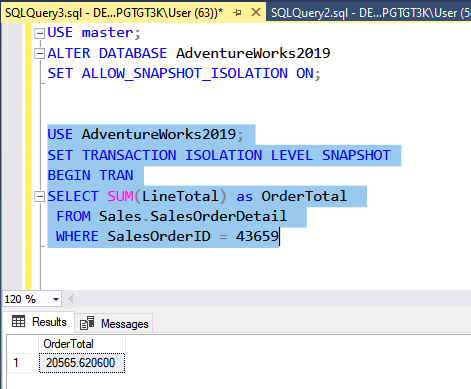
**

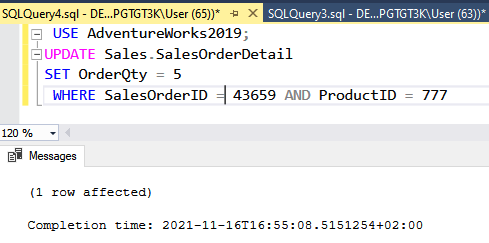
*Рис. 13 Одне з роздільних блокувань має статус WAIT. Це запит, виконуваний у вікні 1*

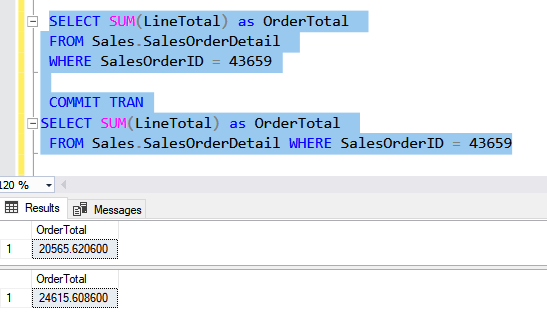
**

**

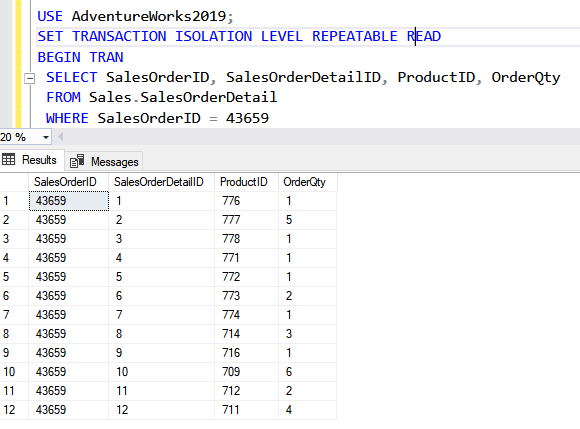
*Рис.14-15 Рівень ізоляції транзакції - Read Committed Snapshot*

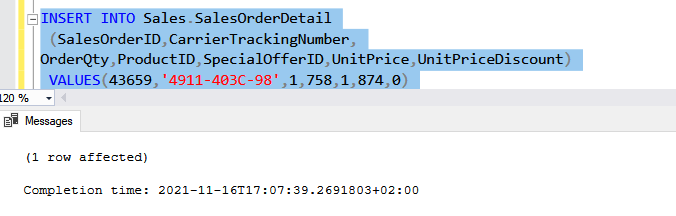
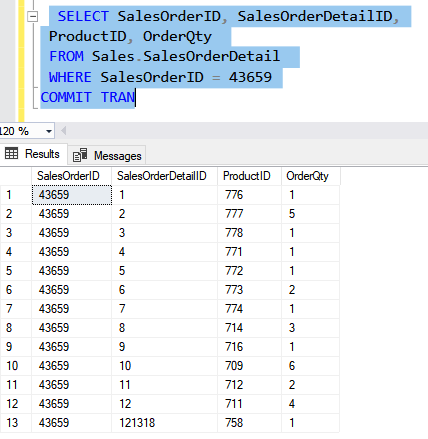
**

**

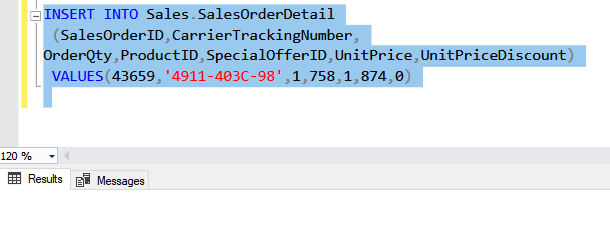
**

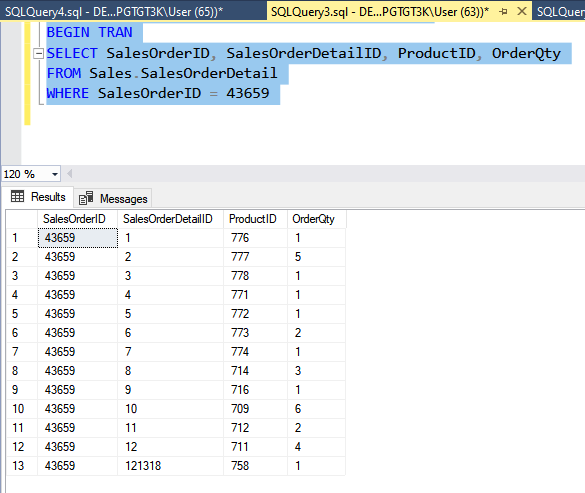
*Рис.16-18 Рівень ізоляції транзакції - Snapshot*

**

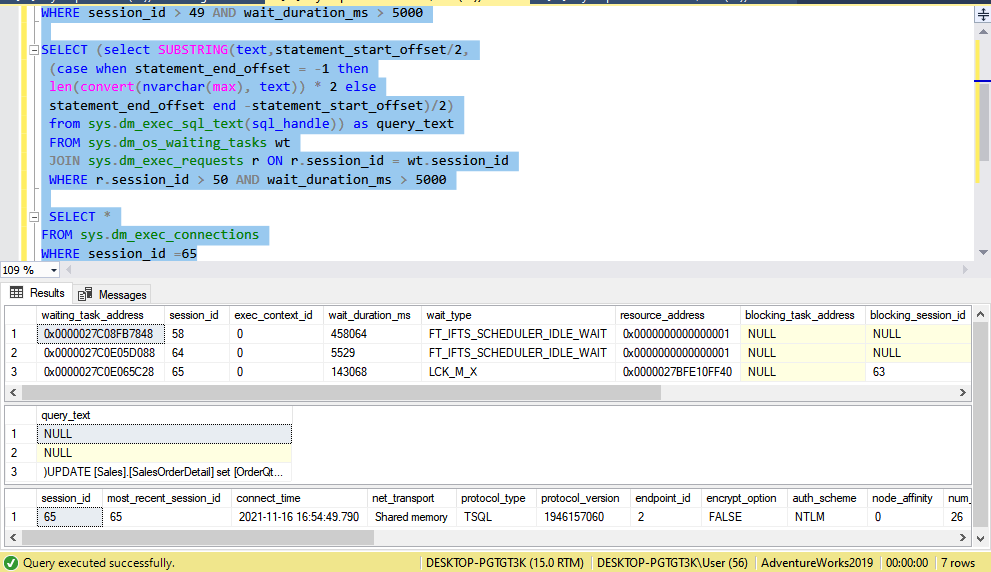
** **

*Рис.19-21. При рівні ізоляції REPEATABLE READ зміна даних впродовж транзакції заборонена, але вставка елементів дозволена, тому ми отримуємо ефект фантомних рядків при повторному читанні*

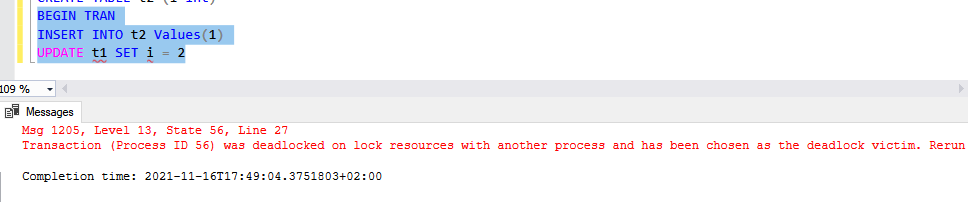
**

**

*Рис.22-23. При рівні ізоляції SERIALIZABLE блокується також діапазон даних, тому вставка за допомогою INSERT також не доступна*

**

*Рис. 24 моніторинг блокувань. Інструкція UPDATE заблокована*

**

*Рис.25. Виявлення SQL Server взаємоблокування та відкат однієї з транзакцій*

1. Визначив межі транзакції та розробив механізм обробки помилок.

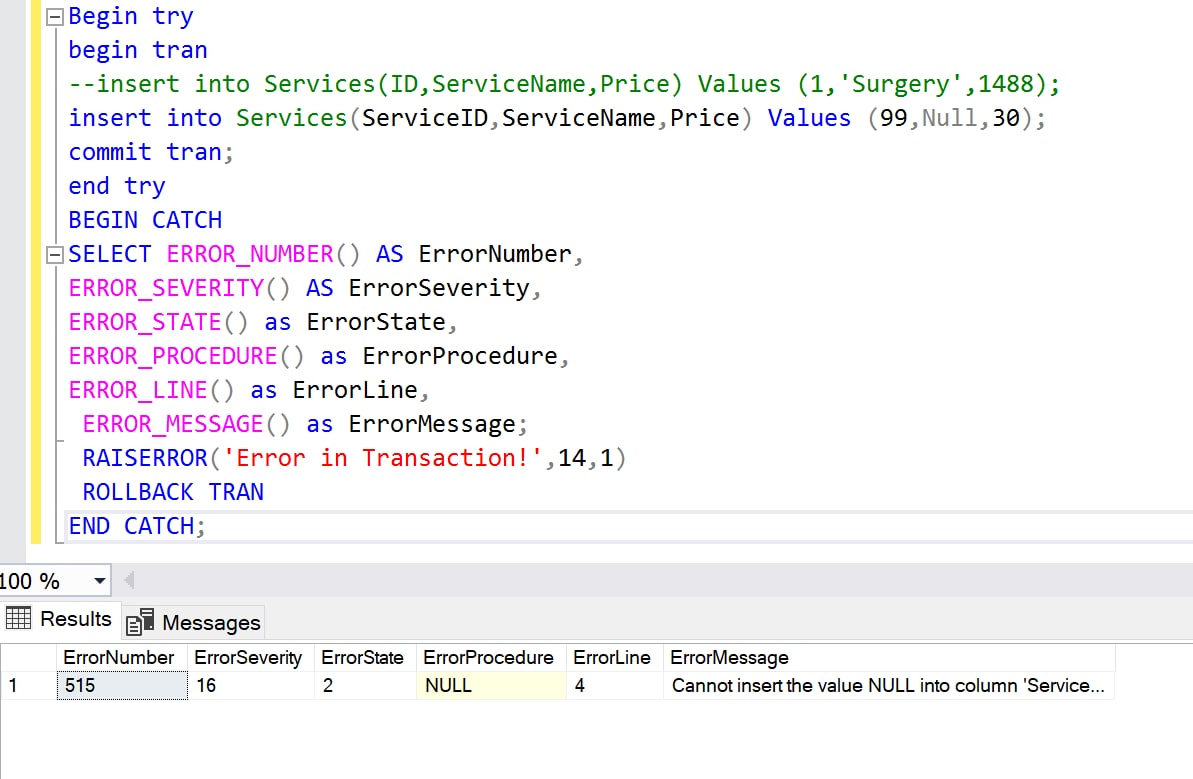


Рис. 26. *Вставка значеннь, що містить поле NULL*

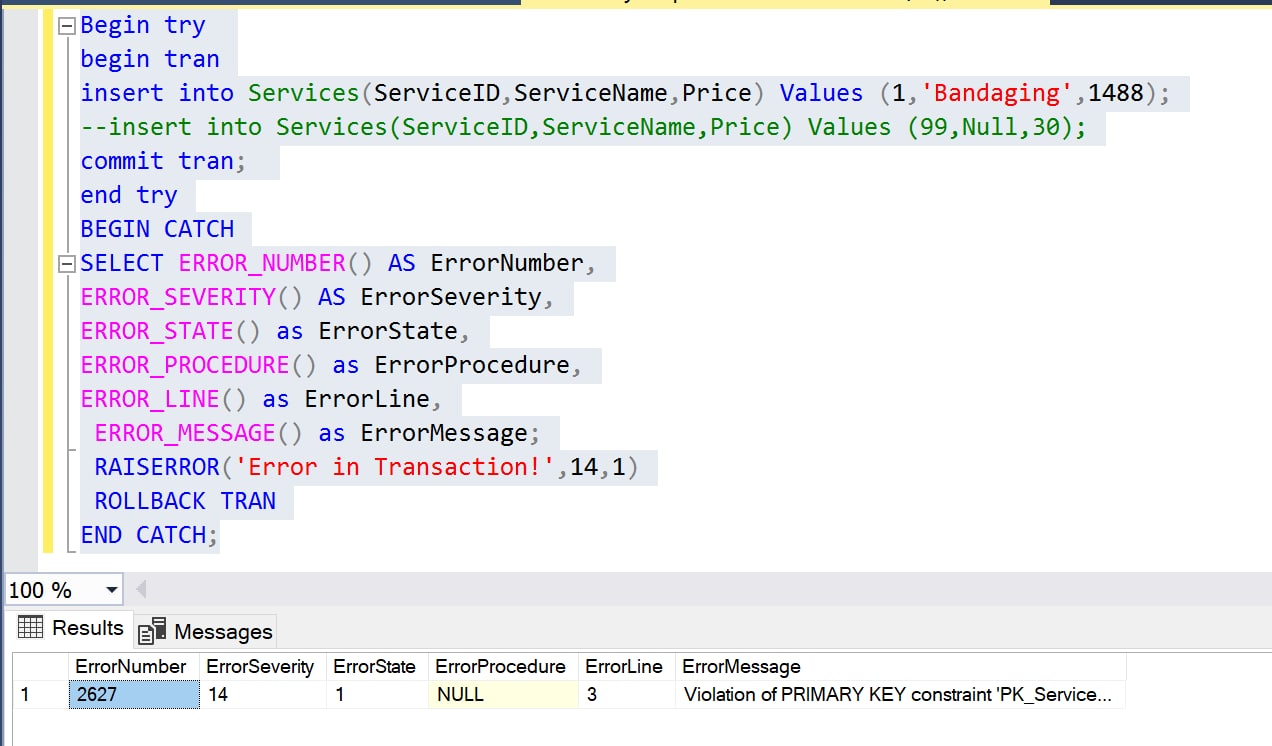
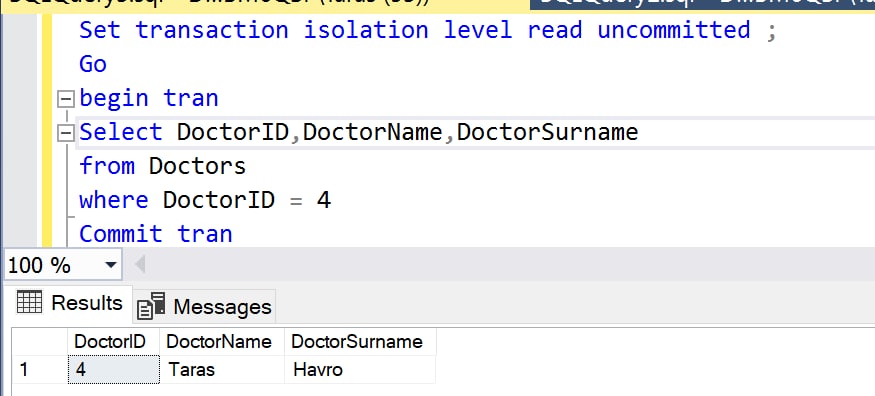
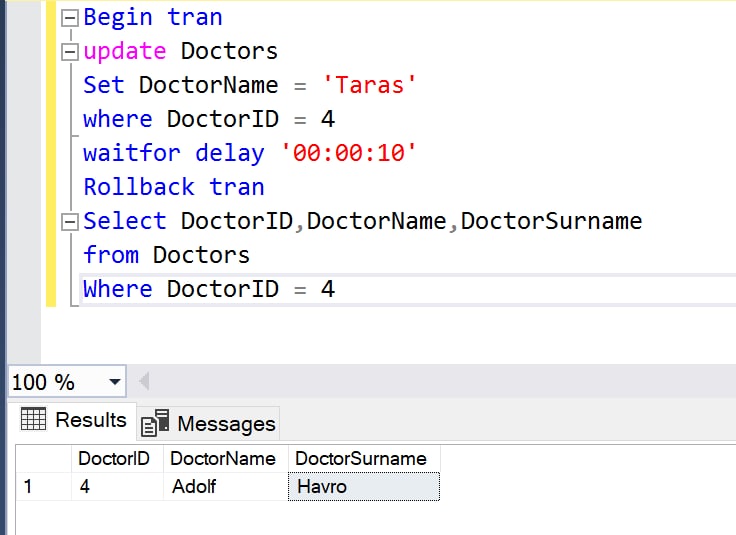


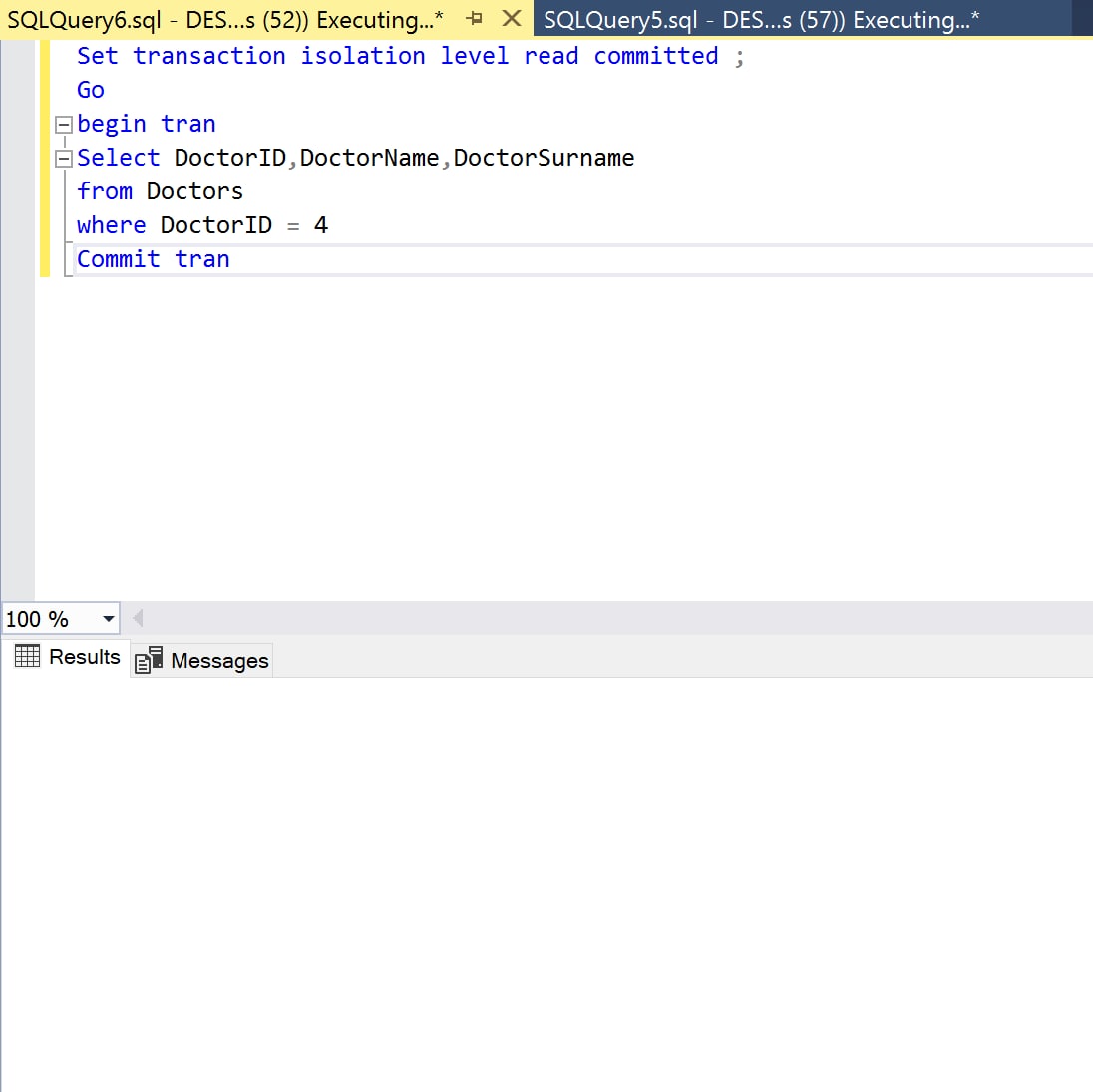
Рис. 27 *Вставка рядку, що містить вже існуючи первинний ключ*

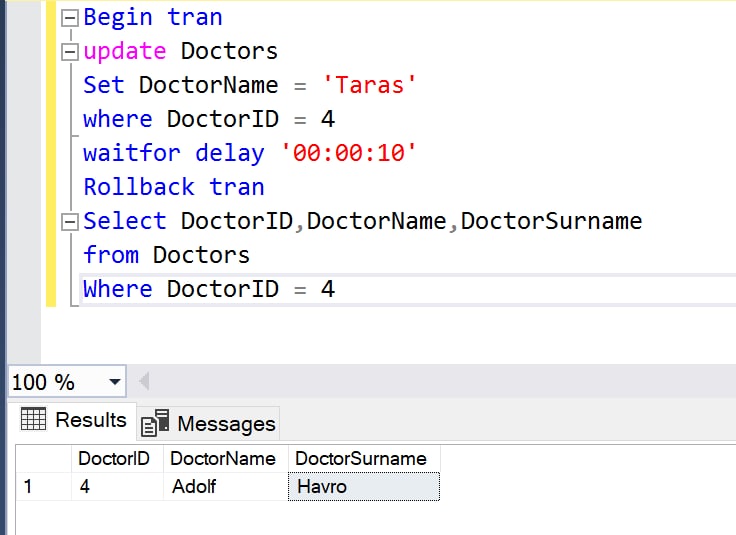
1. Встановив різні рівні ізоляції



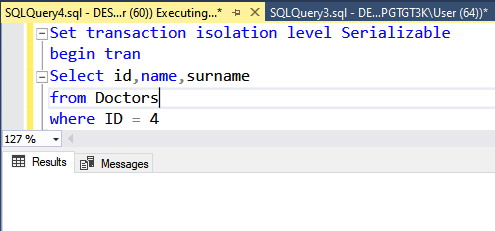


*Рис. 28-29. Рівень ізоляції READ UNCOMMITTED. Другий запит вивів не зафіксовані дані, тому що у першому запиті було відмінено зміни.*



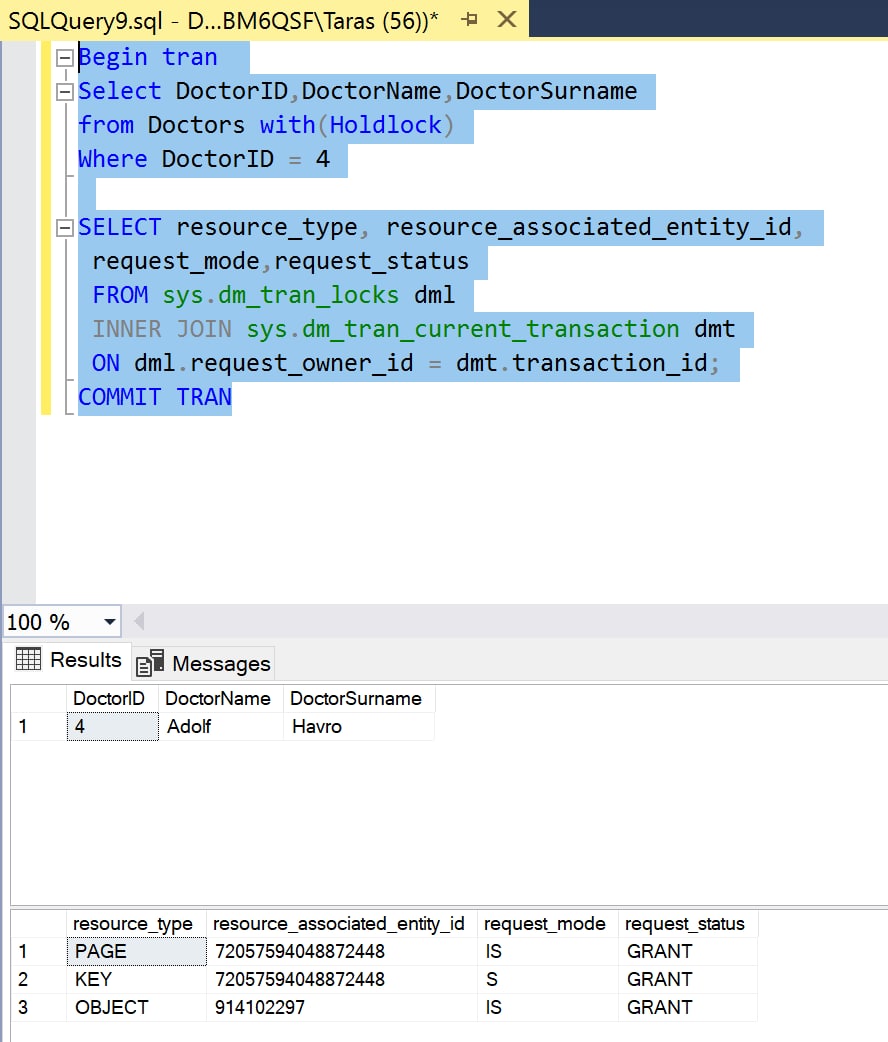


*Рис. 30-31 Рівень ізоляції READ COMMITTED. Другий запит чекає завершення транзакції запущеної у першому запиті.*



*Рис. 32. Встановлення рівня ізоляції SERIALIZABLE. Операція не закінчена*

1. Продемонстрував моніторинг блокувань



*Рис. 33. Моніторинг блокувань*

**Висновок**

У ході виконання цієї роботи я ознайомився з транзакціями. Навчився створювати межі транзакції, їх планувати та відновлювати помилки під час транзакцій. Також випробував різні рівні ізоляції для ізоляції транзакцій