Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

**Звіт**

**до лабораторної роботи № 4**

**з дисципліни «Проектування та використання баз даних»**

**Тема : «Вивчення механізмів формування транзакцій, рівні ізоляції транзакцій і взаємні блокування»**

Виконав:

студент групи ТР-23

Ровний Григорій

*Перевірив :*

асистент

Головакін Микита Андрійович

КИЇВ – 2024

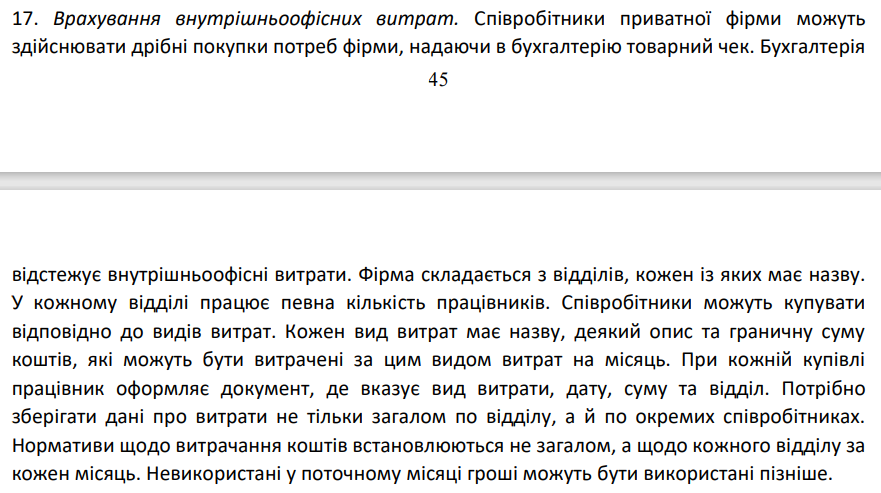
**Мета лабораторної роботи :** Вивчити поняття транзакція, блокування.

**Завдання:**

Для заданої предметної області :

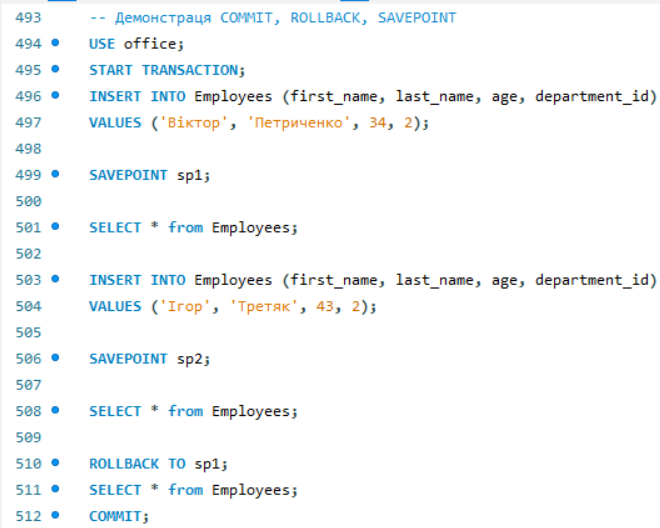
1. Продемонструвати роботу транзакції (підтвердження та відкат транзакції, точки збереження SAVEPOINT транзакцій);
2. Продемонструвати таку властивість транзакції як блокування доступу до даних;
3. Продемонструвати робту інших команд блокування даних (lock / unlock та ін.);

**Варіант**

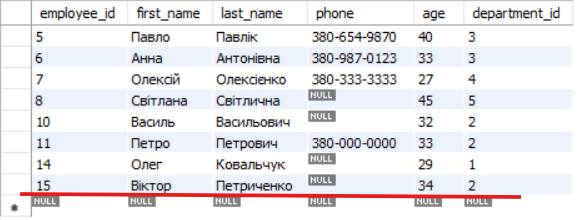


**Результати виконання завдань:**

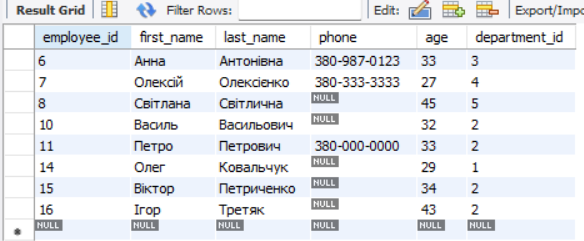
**Завдання 1:**



Спробуємо виконати команди в порядку черги. Запустимо транзакцію **START TRANSACTION;** після чого спробуємо вставити в таблицю Employees нові дані і виконати команду **SAVEPOINT sp1.** Перевіряємо нашу таблицю, виконаємо SELECT для вибірки даних з таблиці:



Як бачимо до таблиці було доданого нового робітника. Заповнимо цю саму таблицю іншими даними, знову виконаємо **SAVEPOINT sp2.** Як бачимо, в таблицю було доданого нового користувача

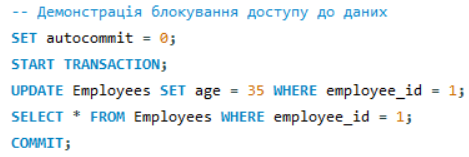


Спробуємо зробити відкат до точки sp1 за допомогою **ROLLBACK TO sp1;** Як бачимо, останні додані дані зникли з таблиці.



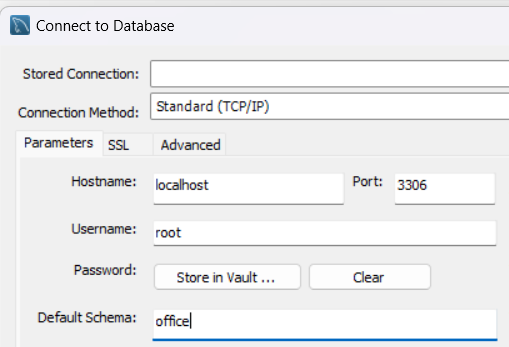
Після виконаня команди **COMMIT;** відкочуватись до сейфпоінтів буде не можливо, оскільки транзакція повноцінно завершує свою роботу після коміту.

**Завдання 2:**

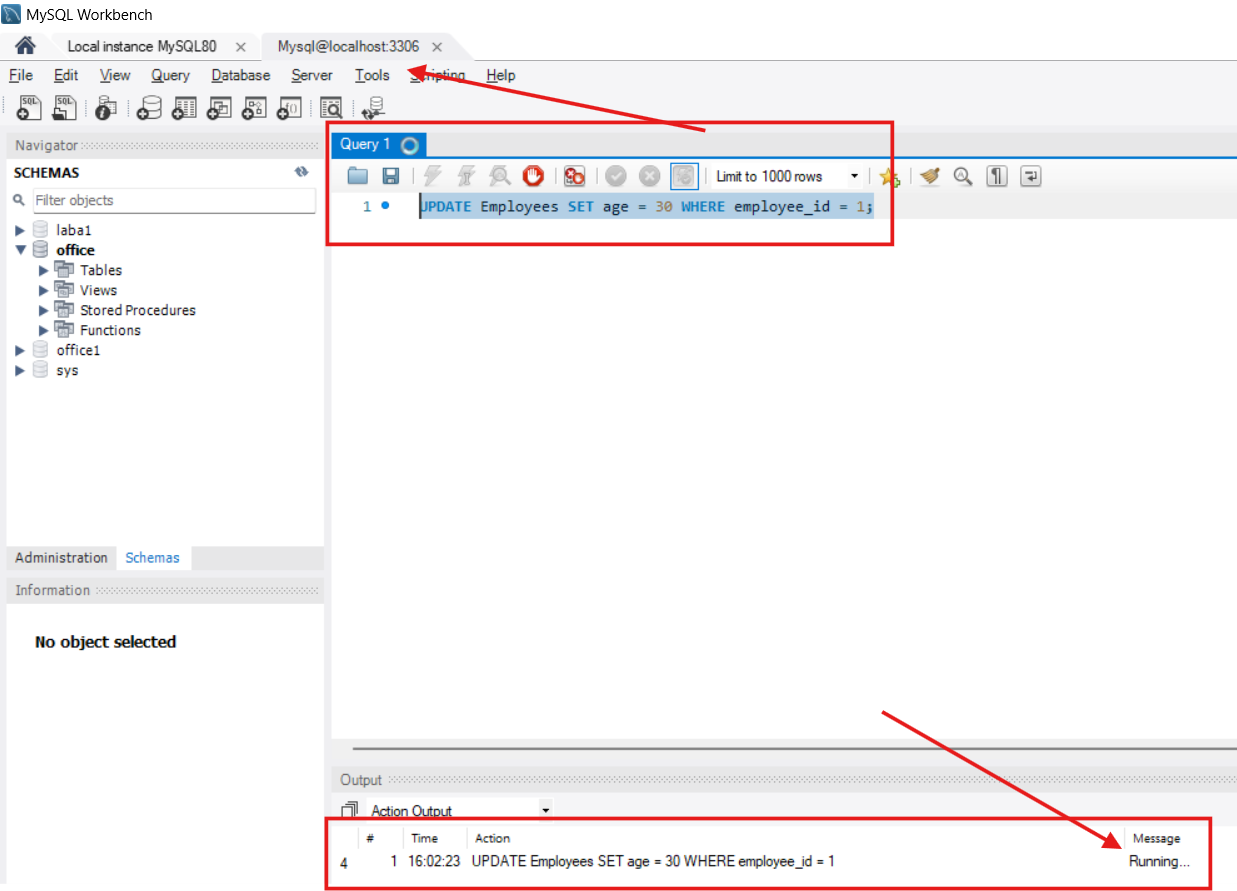


До початку роботи з наступним завданням треба виконати **SET autocommit = 0;** щоб вимкнути автоматичне підтвердження транзакцій, що може завадити блокуванню.

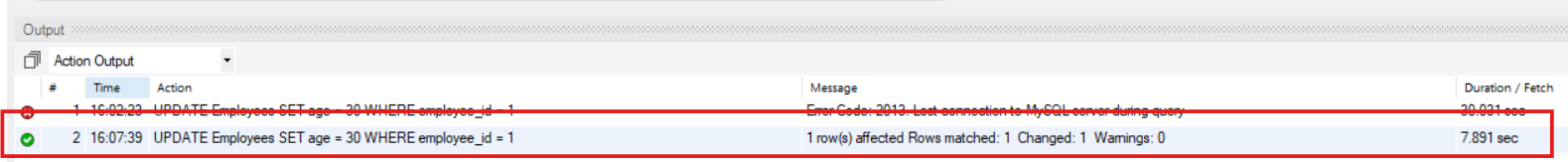
Запускаємо транзакцію та виконуємо апдейт будь яких даних. Після чого необхідно законектитись до цієї ж бази даних через іншу сесію (Connect to database):

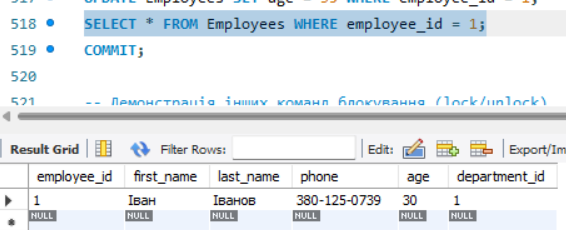


В ції сесії спробуємо змінити ці самі поля за допомогою команди **update**. Як бачимо наша транзакція «зависла». Цей запит заблокувався і не виконається, доки в нашій поточній сесії, де запущена транзакція, не завершиться транзакція (тобто не виконається команда **COMMIT;**).

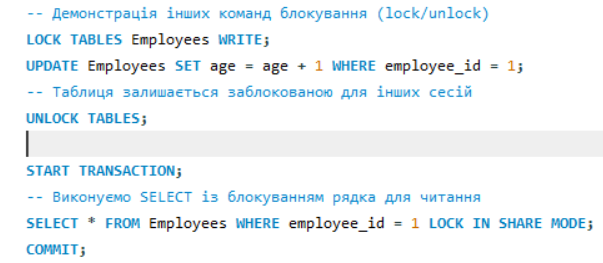


Спробуємо виконати команду **COMMIT;** щоб завершити нашу транзакцію і поврнутися сюди. Як бачимо, команда була виконана, і дані в таблиці біли змінені саме на ті, які виконались після завершення транзакії. Така властивість транзакій дозволяє забезпечити цілісність і непошкодженість даних у разі різних аварійних ситуацій.





**Завдання 3:**

****

Спробуємо заблокувати таблицю на запис (**LOCK TABLES Employees WRITE;**) Таблиця залишається заблокованою для інших сесій. Наприклад, якщо в іншій сесії виконати команду апдейту, то вона знову зависне, доки не буде виконано розблокування таблиці (**UNLOCK TABLES;**). Аналогічно, якщо виконати **LOCK IN SHARE MODE** або **FOR UPDATE;** то отримаємо такий самий «завис» доки не буде зроблено **COMMIT;**

