**Лабораторна робота №9** Розроблення тригерів

**Мета:** Набуття практичних навичок створення тригерів до БД

***Методичні рекомендації***

*При вивченні теми слід звернути увагу на створення та збереження структури бази даних(БД), редагування структури, визначення типів полів, зв’язування таблиць у БД.*

1. Повторити лекції №7, 8, теоретичну частину ЛР № 4-8 та ПР№6-7. Ознайомитися з теоретичною частиною цієї ЛР.

1. Реалізувати тригери, що обробляють внесення змін до таблиць БД з використанням оператора INSERT.

2. Реалізувати тригери, що забороняють внесення змін до таблиць БД з використанням оператора INSERT.

3. Реалізувати тригери, що обробляють внесення змін до таблиць БД з використанням оператора UPDATE.

4. Реалізувати тригери, що забороняють внесення змін до таблиць БД з використанням оператора UPDATE.

5. Реалізувати тригери, що обробляють внесення змін до таблиць БД з використанням оператора DELETE.

6. Реалізувати тригери, що забороняють внесення змін до таблиць БД з використанням оператора DELETE.

Таблиці обрати самостійно.

1. Результати оформити звітом, в якому надаються сформовані SQL-запити та опис дій, що вони виконують.
2. Оформити результати відповідно до стандарту подання лабораторних/ практичних робіт.
3. Результати надсилати на електронну адресу викладача [t.i.lumpova@gmail.com](mailto:t.i.lumpova@gmail.com)

Файл повинен мати назву в такому форматі:

**DB<Номер групи><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище англійською>**. Наприклад, **DB3101R**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт -"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи ІПЗ-31, ІПЗ-32 - 02.12.2023**

Контрольні запитання

1. Поняття процедури та функції, призначення, властивості.

2. Переваги процедури у порівнянні з іншими об’єктами бази (запитами, представленнями).

3. Поняття локальної та глобальної процедури.

4. Призначення тригеру, типи тригерів.

5. Команди щодо створення процедури, вхідні та вихідні параметри.

6. Команди СКБД щодо перевірки стану роботи тригерів та їх заборони.

7. Тригери бази даних. Призначення, особливості застосування

**Теоретична частина**

Створення тригерів в БД SQL-Lite: <https://coderlessons.com/tutorials/bazy-dannykh/vyuchit-sqlite/sqlite-triggery>

<https://www.w3big.com/ru/sqlite/sqlite-trigger.html#gsc.tab=0>

<https://club.shelek.ru/viewart.php?id=353>

<https://coderlessons.com/tutorials/bazy-dannykh/vyuchit-sqlite/sqlite-kratkoe-rukovodstvo>

<https://www.researchgate.net/profile/Alexei-Piskunov/publication/344750300_SQLite_i_vvedenie_v_razrabotku_klient_-_servernyh_prilozenij_na_azyke_C/links/600ec22845851553a06b4521/SQLite-i-vvedenie-v-razrabotku-klient-servernyh-prilozenij-na-azyke-C.pdf>

<https://spec-zone.ru/sqlite/foreignkeys>

SQLite Triggers – функції зворотного виклику бази даних, які виконуються/ викликаються автоматично при виникненні певної події бази даних. Синтаксис:



Тригер SQLite може бути створений для запуску кожного разу, коли спрацьовує команда DELETE, INSERT або UPDATE до конкретної таблиці бази даних, або коли відбувається UPDATE на одному або декількох вказаних стовпцях таблиці.

Наразі SQLite підтримує тільки тригери FOR EACH ROW, а не FOR EACH STATEMENT. Отже, чітке визначення FOR EACH ROW є необов’язковим.

Ключові слова BEFORE або AFTER визначають, коли буде виконуватися тригер, відносно вставки, модифікації або видалення відповідного рядка.

Тригери автоматично видаляються, коли видаляється таблиця, з якою вони пов’язані.

Таблиця, яку потрібно змінити, має існувати в тій самій базі даних, що і таблиця або представлення, до яких прикріплений тригер, звернення до таблиці має виглядати, як tablename, а не database.tablename.

Спеціальна функція SQL RAISE() може бути використана в тілі тригерів для ініціювання винятків (exception).

Event\_name може приймати значення операцій БД INSERT, DELETE та UPDATE до таблиці table\_name. За бажанням можна вказати FOR EACH ROW після назви таблиці.

Приклад тригера тригеру операції оновлення на одному або більше заданих стовпцях таблиці



Ці оператори розберемо он-лайн.

Прикладивиконання операцій щодо створення процедур та тригерів надані з використанням операторів **MS SQL Server**.

***Процедури***

Збережені процедури представляють собою набір команд, що складається з одного або кількох операторів SQL, або функцій та зберігається в базі даних в відкомпілювався вигляді. У SQL Server є кілька типів збережених процедур.

Системні процедури призначені для виконання різних адміністративних дій. Практично всі дії з адміністрування сервера виконуються з їх допомогою. Можна сказати, що системні збережені процедури є інтерфейсом, що забезпечує роботу з системними таблицями, яка, в кінцевому рахунку, зводиться до зміни, додавання, видалення і вибірці даних із системних таблиць як для користувача, так і системних баз даних. Системні процедури мають префікс «sp\_», зберігаються в системній базі даних і можуть бути викликані в контексті будь-якої іншої бази даних.

Користувальницькі процедури реалізують ті чи інші дії. Збережені процедури - повноцінний об'єкт бази даних. Внаслідок цього кожна збережена процедура розташовується в конкретній базі даних, де і виконується.

Тимчасові процедури існують лише деякий час, після чого автоматично знищуються сервером. Вони діляться на локальні і глобальні.

Локальні тимчасові процедури можуть бути викликані тільки з того з'єднання, в якому створені. При створенні такої процедури їй необхідно дати ім'я, що починається з одного символу **#.** Як і всі тимчасові об'єкти, збережені процедури цього типу автоматично видаляються при відключенні користувача, перезапуск або зупинці сервера.

Глобальні тимчасові процедури доступні для будь-яких з'єднань сервера, на якому є така ж процедура. Для її визначення досить дати їй ім'я, що починається з символів **#** #. Видаляються ці процедури при перезапуску або зупинці сервера, а також при закритті з'єднання, в контексті якого вони були створені.

Приклад створення процедури, що змінює оклад співробітника.

**Use sample;**

**go**

**Create procedure FFF (@oklad varchar(20) =5)**

**As**

**Begin**

**Update dbo.sotrydnik Set oklad =@oklad;**

**End**

**go**

Запуск процедури здійснюється за допомогою команди **ЕХЕС.** Приклад запуску процедури:

**ЕХЕС increase\_budget;**

Приклад: Створення процедури для зменшення ціни товару першого сорту на 10%

**CREATE PROC my\_proc2**

**AS**

**UPDATE Товар SET Ціна = Ціна \* 0.9**

WHERE Сорт = 'перший'

Для звернення до процедури можна використовувати команду:

EXEC my\_proc2

Приклад створення процедури з вхідним параметром.

Створити процедуру для отримання назв і вартості товарів, які придбав заданий клієнт.

CREATE PROC my\_proc3

@ K VARCHAR (20)

**AS**

**SELECT Товар.Названіе,**

**Товар.Цена \* Сделка.Колічество**

AS Вартість, Кліент.Фамілія

FROM Клієнт INNER JOIN

(Товар INNER JOIN Угода

ON Товар.КодТовара = Сделка.КодТовара)

ON Кліент.КодКліента = Сделка.КодКліента

WHERE Кліент.Фамілія = @ k

Для звернення до процедури можна використовувати команду:

**EXEC my\_proc3 'Іванов'**

***Тригери***

При роботі БД повинна забезпечуватися цілісність даних. при видаленні записів з первинних таблиць автоматично повинні віддалятися пов'язані з ними записи з вторинних таблиць. У разі недотримання цього принципу з часом в БД накопичиться велика кількість записів у вторинних таблицях, пов'язаних з неіснуючими записами в первинних таблицях, що призведе до збоїв в роботі БД і її засмічення. Для забезпечення цілісності даних в SQL Server використовують тригери.

Тригер - це поєднання збереженої в базі даних процедури і події, яке змушує її виконуватися. Такими подіями можуть бути: введення нового рядка таблиці, зміна значень одного або декількох її стовпців і (або) видалення рядки таблиці. При будь-якому з цих подій автоматично запускаються один або кілька заздалегідь створених тригерів, які проводять перевірку запрограмованих в них умов, і якщо вони не виконуються, скасовують введення, зміна або видалення, посилаючи про це заздалегідь підготовлене повідомлення користувачеві.

Існує два види тригерів:

- тригери DML, які запускаються при виконанні команд по внесенню змін до таблиці БД.

- тригери DDL, які запускаються при виконанні команд: CREATE, ALTER, DROP.

Тригери DML представлені такими видами:

**INSERT TRIGGER –** тригери цього виду запускаються при спробі вставки даних за допомогою команди INSERT.

**UPDATE TRIGGER -** тригери цього виду запускаються при спробі вставки даних за допомогою команди UPDATE.

**DELETE TRIGGER -** тригери цього виду запускаються при спробі вставки даних за допомогою команди DELETE.

Також тригери можна класифікувати за їх поведінкою.

**AFTER –** тригер виконується після успішного виконання команди, що змінює дані в таблиці. Якщо команда з якихось причин не може бути успішно завершена, то тригер також не виконується. Зміна даних в результаті виконання запиту користувача і виконання тригера виконується в тілі однієї трансакції. Тобто, якщо відбудеться відкіт тригера (в ньому буде виконана команда ROLLBACK TRAN), то також будуть відкочені і користувацькі зміни. Тригери типу AFTER неможна виконати для представлень. Для кожної таблиці можна створити більше одного тригера AFTER для кожної операції (INSERT, UPDATE, DELETE).

Якщо для таблиці передбачено виконання кількох AFTER-тригерів, то за допомогою системної збереженої процедури «**SP\_SETTRIGGERORDER»** можна вказати, який з них буде виконуватися першим, а який останнім. За умовчанням в SQL Server всі тригери є AFTER-тригерами.

**INSTEAD OF** тригер викликається замість команд, назначених для запуску тригера. Тригери INSTEAD OF можуть працювати як з таблицями так і представленнями. За замовчуванням в SQL Server 2008 всі тригери є тригерами типу AFTER.

Кожен тригер пов’язаний з певною таблицею БД і розглядається як одна транзакція, яка починається командою BEGIN і завершується командою END.

Якщо не потрібно тригеру передавати записи потрібно добавити команду ROLLBACK, яка дає наказ серверу призупинити оброблення модифікації і заборонити трансакцію.

Команда **RAISEERROR**  - відправляє повідомлення користувачу про помилку.

Для отримання інформації про кількість рядків, що будуть змінені при успішному завершенні тригера, можна використовувати функцію @@*ROWCOUN.*

Розглянемо роботу тригерів, що обробляють операції внесення змін до таблиці за допомогою команд INSERT, UPDATE чи DELETE.

Внесення змін до БД виконується операторами мови маніпулювання даними **DML** (Data Manipulation Language), яка є складовою мови SQL.

Основними командами DML є команди – дозапису, модифікації і вилучення даних INSERT, UPDATE и DELETE.

**Тригер INSERT**

Тригери INSERT запускаються при кожній спробі створити новий запис в таблиці командо INSERT. При спробі вставити новий запис в таблицю SQL Server копіює цей запис в таблицю тригерів БД і спеціальну таблицю **inserted.** За допомогою тригера можливий контроль для реалізації обмежень на значення окремих полів.

**Тригер DELETE**

Тригери DELETE запускаються при кожній спробі вилучити запис з таблиці командою DELETE. Вилучений запис з таблиці SQL Server копіює в таблицю тригерів БД і спеціальну таблицю **deleted.**

Тригери можливо створювати за допомогою мови програмування C# або Visual Basic.

**INSTEAD OF.** Тригер викликається замість виконання команд. На відміну від AFTER-тригера INSTEAD OF-тригер може бути визначений як для таблиці, так і для представлення.

Для кожної операції **INSERT, UPDATE, DELETE** можна визначити тільки один INSTEAD OF-тригер.

Приклад використання тригера з оператором INSERT для таблиці **sotrydnik** наведений:

**create TRIGGER prim1**

**ON sotrydnik**

**after INSERT**

**AS**

**DECLARE @oklad int**

select @oklad =(select oklad from inserted)

if (@oklad)<2000

**BEGIN**

ROLLBACK TRANSACTION

**end**

**if not (@oklad)<2000**

**BEGIN**

**select @oklad=(select oklad from inserted)**

**end;**

**Робота на занятті**

**Доповнимо таблицю Students полем "BegDate" TEXT дата модифікації**

**Доповнимо таблицю Teachers полем "BegDate" TEXT дата модифікації**

**CREATE TABLE Teachers (**

**ID\_T smallint NOT NULL ,**

**Name\_T text, -- Teacher's Name**

**ID\_G integer**

**, "Birth" TEXT DEFAULT '01-01-2000', "BegDate" TEXT)**

**CREATE TABLE AUDIT(**

**log\_ID INTEGER PRIMARY KEY AUTOINCREMENT,**

**Tab\_N TEXT NOT NULL,**

**ENTRY\_DATE TEXT NOT NULL);**

**CREATE TRIGGER mody\_log AFTER INSERT ON Teachers**

**BEGIN**

**INSERT INTO AUDIT(Tab\_N, ENTRY\_DATE )**

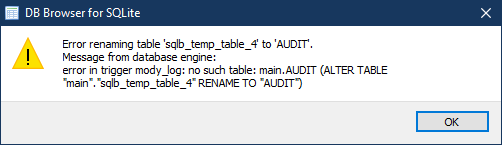
**VALUES (' Teachers ',datetime('now'));**

**END;**

**Insert into Teachers (ID\_T,**

**Name\_T, ID\_G, Birth, BegDate)VALUES(1, 'New Teacher',1, '01-01-1995', '28-11-2023')**

**Додаємо нове поле он-лайн**



**Drop TRIGGER mody\_log**

**Після цього додаємо поле OP для фіксації операції**

**CREATE TABLE "AUDIT" ( "log\_ID" INTEGER, "Tab\_N" TEXT NOT NULL, "ENTRY\_DATE" TEXT NOT NULL, "OP" TEXT, PRIMARY KEY("log\_ID" AUTOINCREMENT) )**

**CREATE TRIGGER mody\_log AFTER INSERT ON Teachers**

**BEGIN**

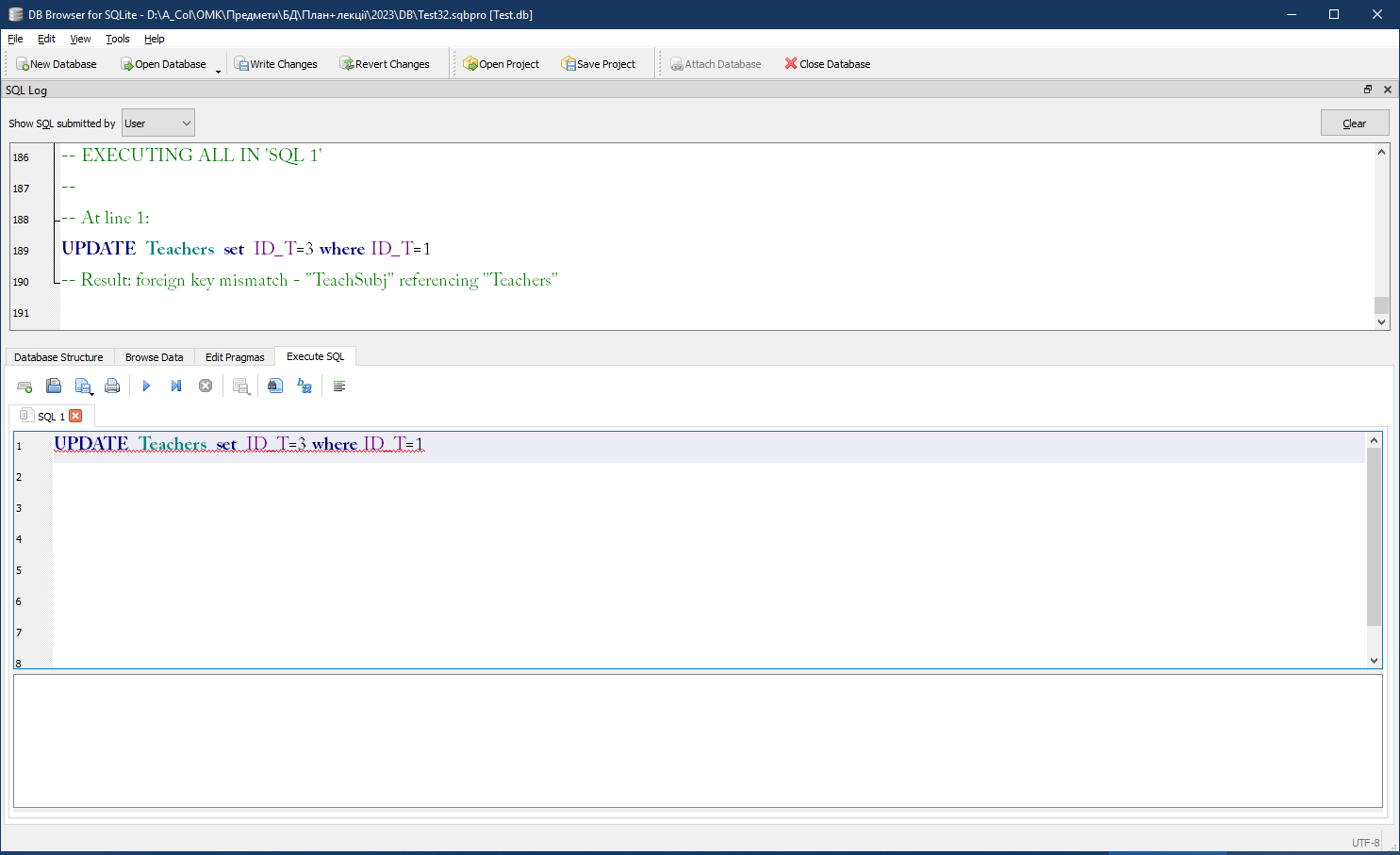
**INSERT INTO AUDIT(Tab\_N, ENTRY\_DATE,OP )**

**VALUES (' Teachers ',datetime('now'), 'Insert' );**

**END;**

**Insert into Teachers (ID\_T,**

**Name\_T, ID\_G, Birth, BegDate)VALUES(1, 'New Teacher',1, '01-01-1995', '28-11-2023')**



**CREATE TRIGGER mody\_Teach AFTER UPDATE ON Teachers**

**BEGIN**

**INSERT INTO AUDIT(Tab\_N, ENTRY\_DATE,OP )**

**VALUES (' Teachers ',datetime('now'), 'UPDATE' );**

**END;**

**CREATE TABLE "AUDIT\_col" ( "log\_ID" INTEGER, "Tab\_N" TEXT NOT NULL, "OP" TEXT, "Col" TEXT, "ENTRY\_DATE" TEXT NOT NULL, PRIMARY KEY("log\_ID" AUTOINCREMENT) )**

**CREATE TRIGGER mody\_col AFTER Update of Sub\_code ON TeachSubj**

**BEGIN**

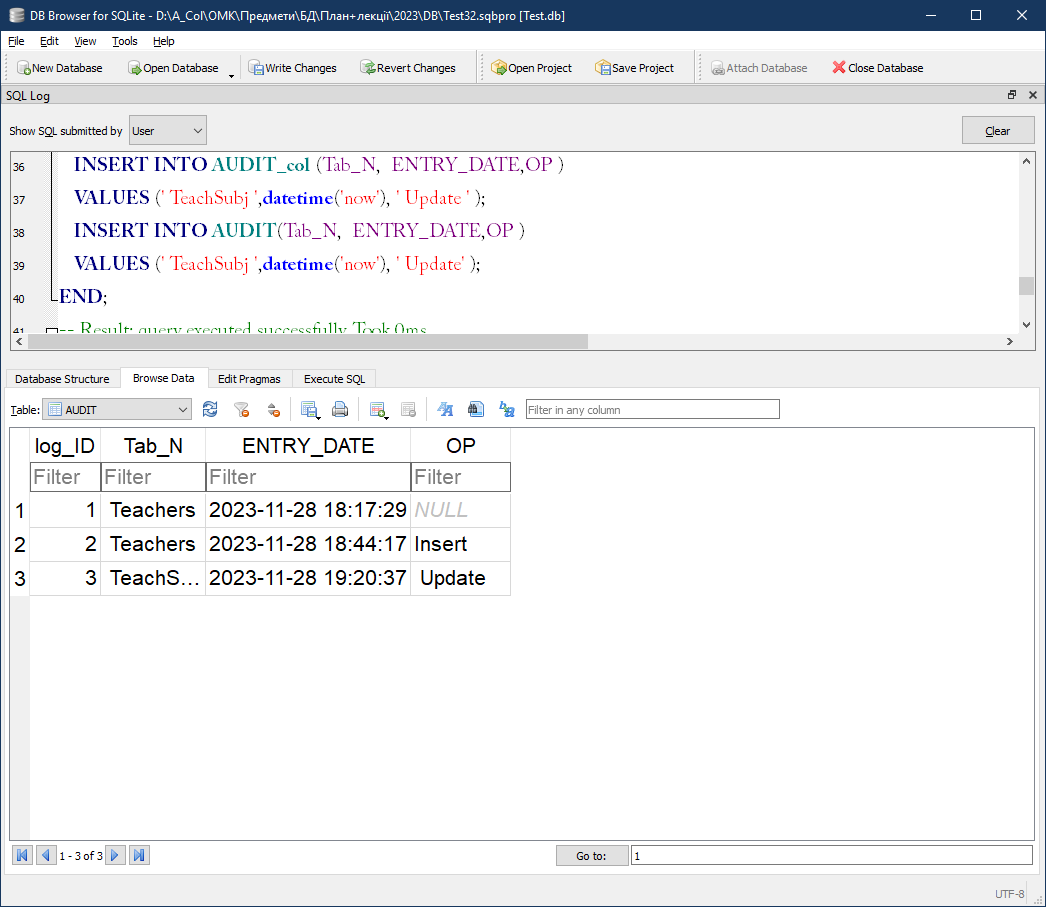
**INSERT INTO AUDIT\_col (Tab\_N, ENTRY\_DATE,OP )**

**VALUES (' TeachSubj ',datetime('now'), ' Update ' );**

**INSERT INTO AUDIT(Tab\_N, ENTRY\_DATE,OP )**

**VALUES (' TeachSubj ',datetime('now'), ' Update' );**

**END;**



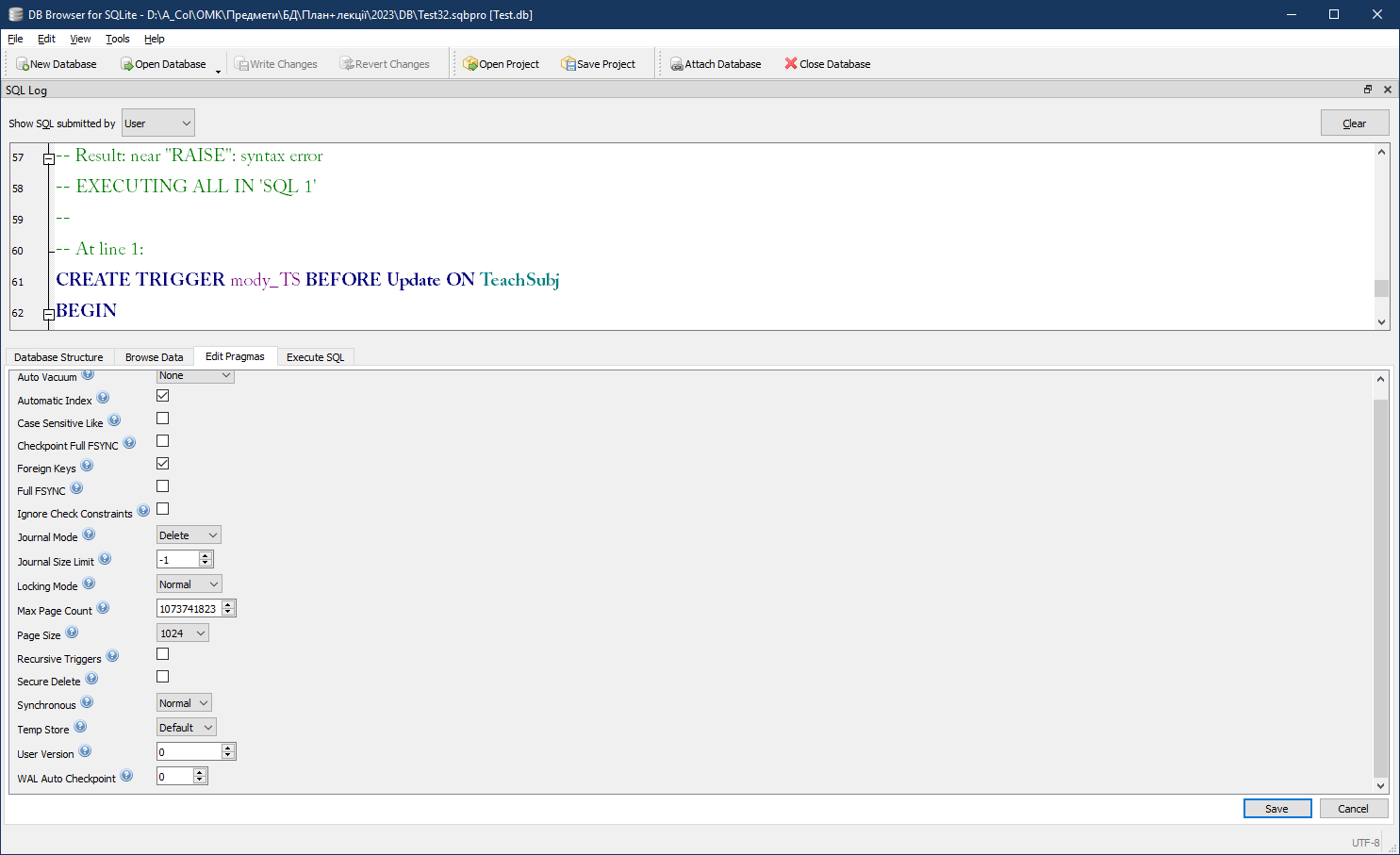
**Update TeachSubj set Sub\_code=12 where id\_ts=3**

**Предмета з кодом 7 не існує**

**Update TeachSubj set Sub\_code=7 where id\_ts=4**

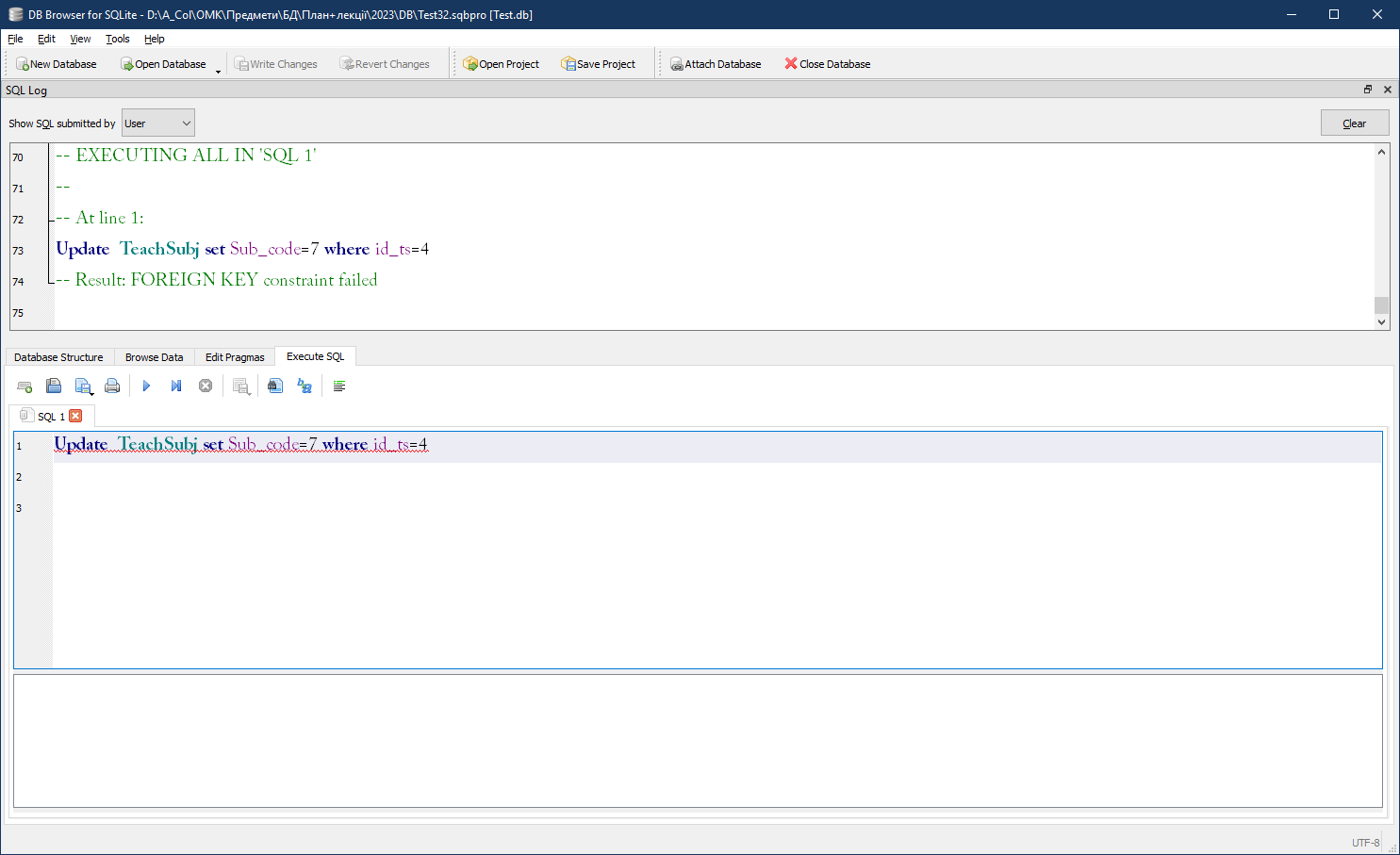
**Але повідомлення** SQL error: foreign key constraint failed

**Не видається**



**Після встановлення прапорця у прагмі**

**отримуємо**



**CREATE TRIGGER mody\_TS BEFORE UPDATE ON TeachSubj**

**BEGIN**

**UPDATE Subject set Sub\_code=new.Sub\_code where Sub\_code= old.Sub\_code;**

**--RAISE(FAIL)**

**END;**

**Update TeachSubj set Sub\_code=2 where id\_ts=4**