МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота №5**

з дисципліни « Реляційні бази даних »

*назва дисципліни*

на тему: «Тригери Баз даних»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Я.І.

(прізвище й ініціали студента)

Прийняв: канд.тех.наук, каф. 603

Манжос Ю.С.

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2024

ПОСТАНОВКА ЗАДАЧІ

1. Створити тригери, які виконують завдання варіанту.
2. Для кожної таблиці, що має автоінкрементний первинний ключ, створити тригер для автоматичної генерації цього первинного ключа під час внесення даних
3. Здійснити денормализацію «уверх» однієї з таблиць, додавши у неї підсумкове поле; забезпечити коректність інформації в цьому полі за допомогою тригерів. Наприклад, для моделі «Студент-Група- Оцінка» у таблицю «Група» додати поле «середній бал групи» ; забезпечити коректність інформації цього поля за допомогою тригерів.
4. Здійснити денормализацію «униз» однієї з таблиць, додавши у неї поле ( Наприклад, у таблицю «Студент» додати поле «Номер факультету»); забезпечити коректність інформації цього поля за допомогою тригерів.
5. За допомогою тригерів до однієї з таблиць додати журнал, в якому повинні бути зафіксовані відомості про дії користувачів над цією таблицею.
6. Протестувати створені тригери.
7. Оформити звіт.

**Варіант завдань – 10**

Створити тригер, який НЕ буде дозволяти клієнту автозаправної станції в один день заправлятися кілька разів.

Виконання роботи

Створення тригера за варіантом

Створити тригер, який НЕ буде дозволяти клієнту автозаправної станції в один день заправлятися кілька разів.

Код створення тригера

DELIMITER //

CREATE TRIGGER `before\_insert\_sale`

BEFORE INSERT ON `продаж`

FOR EACH ROW

BEGIN

DECLARE cnt INT;

SET cnt = (

SELECT COUNT(\*)

FROM `продаж`

WHERE `НомерДисконту` = NEW.`НомерДисконту`

AND DATE(`ДатаПродажу`) = DATE(NEW.`ДатаПродажу`)

);

IF cnt > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Клієнт вже заправлявся цього дня!';

END IF;

END //

DELIMITER ;

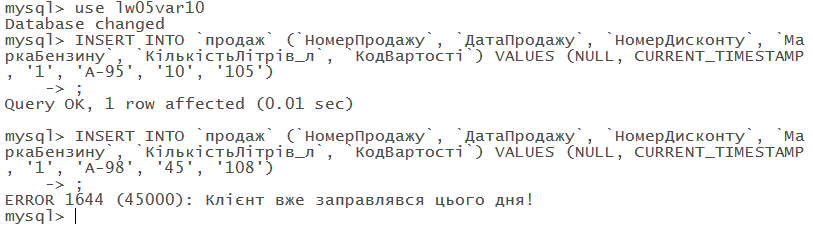


Рисунок 1 – перевірка роботи тригера разової заправки на день

Cтворення тригерів для автоматичної генерації первинних ключів під час внесення даних в таблиці з автоінкрементним первинним ключем.

Код створення тригера для таблиці **клієнт**

DELIMITER //

CREATE TRIGGER `auto\_num\_client`

BEFORE INSERT ON `клієнт`

FOR EACH ROW

BEGIN

DECLARE next\_id INT;

SET next\_id = (

SELECT IFNULL(MAX(`idAccount`), 0) + 1

FROM `клієнт`

);

SET NEW.`idAccount` = next\_id;

END //

DELIMITER ;

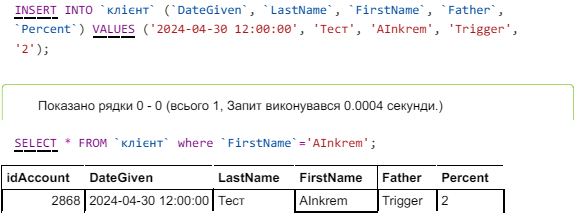


Рисунок 2 – перевірка роботи тригера на авто встановлення номеру id клієнта

Код створення тригера для таблиці **продаж**

DELIMITER //

CREATE TRIGGER `auto\_num\_sale`

BEFORE INSERT ON `продаж`

FOR EACH ROW

BEGIN

DECLARE next\_id INT;

SET next\_id = (

SELECT IFNULL(MAX(`НомерПродажу`), 0) + 1

FROM `продаж`

);

SET NEW.`НомерПродажу` = next\_id;

END //

DELIMITER ;

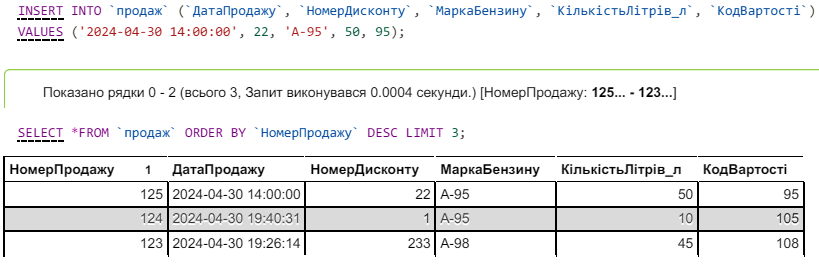


Рисунок 3 – перевірка роботи тригера на авто встановлення номеру продажу

Здійснити денормализацію «уверх» однієї з таблиць, додавши у неї підсумкове поле; забезпечити коректність інформації в цьому полі за допомогою тригерів

DELIMITER $$

CREATE PROCEDURE DeleteOldSales()

BEGIN

DECLARE cutoffDate DATE; -- Дата обрізки

DECLARE rowsAffected INT; -- Кількість змінених рядків

-- Обчислюємо дату, яка знаходиться на рік назад від поточної дати

SET cutoffDate = DATE\_SUB(CURRENT\_DATE(), INTERVAL 1 YEAR);

-- Видаляємо записи про продажі, зроблені більше року тому

DELETE FROM продаж

WHERE ДатаПродажу < cutoffDate; -- Отримуємо кількість видалених записів

SET rowsAffected = ROW\_COUNT(); -- Перевіряємо, чи були видалені записи

IF rowsAffected > 0 THEN -- Якщо видалені записи, виводимо повідомлення про успішне видалення

SELECT CONCAT('Успішно видалено ', rowsAffected, ' записів') AS Message;

ELSE

-- Якщо записи не були видалені, виводимо повідомлення про відсутність записів для видалення

SELECT 'Записів для видалення не знайдено' AS Message;

END IF;

END$$

DELIMITER ;

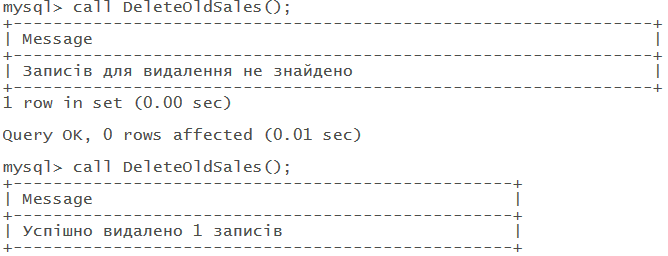


Рисунок 3 – результат виконання процедури з видалення старих записів продажів

Для кожної таблиці з автоінкрементним первинним ключем, написати функцію, яка приймає значення полів запису, крім первинного ключа. Ця функція має отримувати новий первинний ключ, вставляти новий запис в таблицю з цим ключем і з зазначеними у вигляді параметрів значеннями полів. Функція має повертати первинний ключ вставленого запису або NULL , якщо запис НЕ був вставлений.

**для таблиці клієнт:**

DELIMITER $$

CREATE FUNCTION `InsertClient`(p\_DateGiven DATETIME, p\_LastName VARCHAR(50), p\_FirstName VARCHAR(20), p\_Father VARCHAR(50), p\_Percent ENUM('0','1','2','3','4','5'))

RETURNS INT DETERMINISTIC

BEGIN

DECLARE v\_idAccount INT;

-- Встановлюємо p\_DateGiven в поточний час, якщо воно NULL або '0000-00-00 00:00:00'

IF p\_DateGiven IS NULL OR p\_DateGiven = '0000-00-00 00:00:00' THEN

SET p\_DateGiven = CURRENT\_TIMESTAMP;

END IF;

INSERT INTO клієнт (DateGiven, LastName, FirstName, Father, Percent)

VALUES (p\_DateGiven, p\_LastName, p\_FirstName, p\_Father, p\_Percent);

-- Отримуємо ID вставленого запису

SET v\_idAccount = LAST\_INSERT\_ID();

-- Повертаємо ID вставленого запису або NULL, якщо запис не був вставлений

IF ROW\_COUNT() = 0 THEN

RETURN NULL;

ELSE

RETURN v\_idAccount;

END IF;

END$$

DELIMITER ;

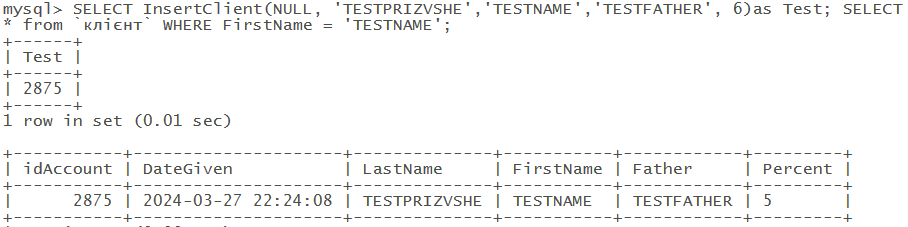
****

Рисунок 1 – створення нового клієнта без введення Id

**для продажу:**

 DELIMITER $$

CREATE FUNCTION InsertSale(

p\_DateGiven DATETIME,

p\_DiscountNumber INT,

p\_GasolineBrand VARCHAR(10),

p\_LitersAmount INT,

p\_CostCode INT

)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE v\_SaleId INT;

-- Встановлюємо p\_DateGiven в поточний час, якщо воно NULL або '0000-00-00 00:00:00'

IF p\_DateGiven IS NULL OR p\_DateGiven = '0000-00-00 00:00:00' THEN

SET p\_DateGiven = CURRENT\_TIMESTAMP;

END IF;

-- Додаємо запис про продаж

INSERT INTO продаж (ДатаПродажу, НомерДисконту, МаркаБензину, КількістьЛітрів\_л, КодВартості)

VALUES (p\_DateGiven, p\_DiscountNumber, p\_GasolineBrand, p\_LitersAmount, p\_CostCode);

-- Отримуємо ID вставленого запису

SET v\_SaleId = LAST\_INSERT\_ID();

-- Повертаємо ID вставленого запису або NULL, якщо запис не був вставлений

RETURN IF(v\_SaleId IS NULL, NULL, v\_SaleId);

END$$

DELIMITER ;

PREPARE t3 FROM 'SELECT DATE(ДатаПродажу) AS Дата\_продажу, SUM(КількістьЛітрів\_л) AS Сумарна\_кількість\_проданих\_літрів FROM продаж

WHERE МаркаБензину = ? AND НомерДисконту IN (SELECT idAccount FROM клієнт WHERE Percent = ?) GROUP BY DATE(ДатаПродажу)'; SET @N = 'А-95'; SET @M = '3'; EXECUTE t3 USING @N, @M;

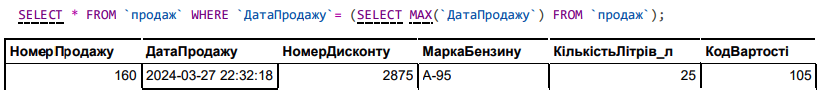
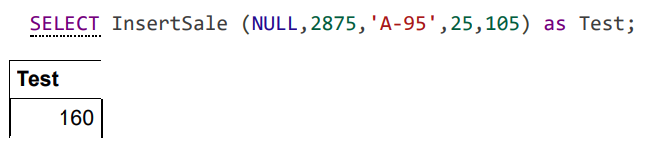


Рисунок 5 – створення нового продажу без введення НомеруПродажу

ВИСНОВОК

У ході виконання лабораторної роботи було створено та використано процедури, функції, курсори до створеної на першій лабораторній роботі бази даних.

Навчилися використовувати та створювати функції, процедури та курсори в базах даних для виконання повторюваних дій з певними умовами, що дозволяє більш ефективно використовувати бази даних.