

Лабораторна робота 5

СТВОРЕННЯ ТА ВИКОРИСТАННЯ ЗБЕРЕЖЕНИХ ПРОЦЕДУР ТА ТРИГЕРІВ

Мета роботи: навчитися створювати та застосовувати програмні об'єкти бази даних – збережені процедури та тригери, на прикладі СУБД MySQL.

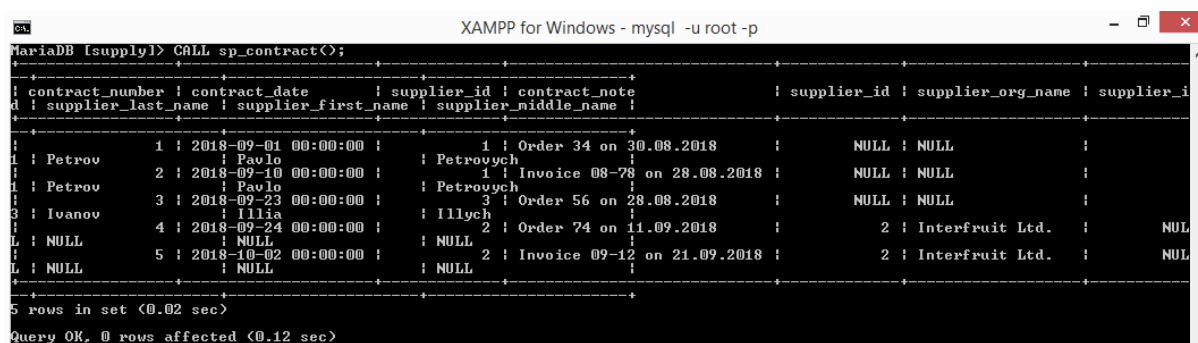
Хід роботи

1. Створення та використання збережених процедур

Створення збережених процедур реалізується оператором CREATE PROCEDURE. Таким чином, створити збережену процедуру, яка реалізує вибірку даних з таблиць contract, supplier_org, supplier_person, можна за допомогою наступної команди (рисунок 5.1).

```
DELIMITER //  
CREATE PROCEDURE sp_contract()  
BEGIN  
    SELECT *  
    FROM (contract LEFT JOIN supplier_org ON  
        contract.supplier_id = supplier_org.supplier_id)  
        LEFT JOIN supplier_person ON  
        contract.supplier_id = supplier_person.supplier_id;  
END //
```

Виклик процедури здійснюється за допомогою оператора CALL.



contract_number	contract_date	supplier_id	contract_note	supplier_last_name	supplier_first_name	supplier_middle_name	supplier_id	supplier_org_name	supplier_i
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018	Petrov	Paulo	Petrovych	NULL	NULL	
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018	Petrov	Paulo	Petrovych	NULL	NULL	
3	2018-09-23 00:00:00	3	Order 56 on 28.08.2018	Ivanov	Illia	Illych	NULL	NULL	
4	2018-09-24 00:00:00	2	Order 74 on 11.09.2018	NULL	NULL	NULL	2	Interfruit Ltd.	NUL
5	2018-10-02 00:00:00	2	Invoice 09-12 on 21.09.2018	NULL	NULL	NULL	2	Interfruit Ltd.	NUL

5 rows in set (0.02 sec)
Query OK, 0 rows affected (0.12 sec)

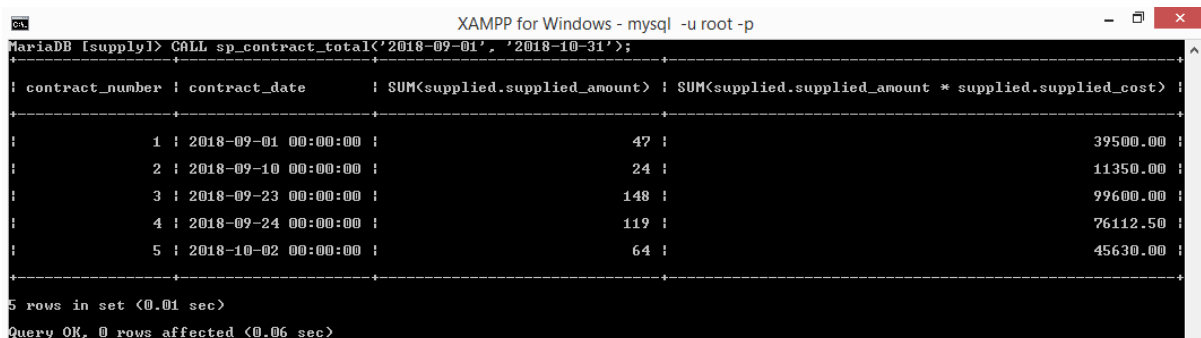
Рисунок 5.1

Для знайомства з особливостями створення та використання процедур з параметрами, необхідно створити збережену процедуру, яка забезпечує формування агрегатних даних за поставками для вказаного інтервалу календарних дат (рисунок 5.2).

```
DELIMITER //
CREATE PROCEDURE sp_contract_total(IN date_from timestamp,
                                   IN date_to timestamp)
BEGIN
    SELECT contract.contract_number, contract.contract_date,
           SUM(supplied.supplied_amount), SUM(supplied.supplied_amount * supplied.supplied_cost)
    FROM contract LEFT JOIN supplied ON contract.contract_number = supplied.contract_number
    WHERE contract.contract_date BETWEEN date_from AND date_to
    GROUP BY contract.contract_number, contract.contract_date;
END //
```

Здійснити виклик створеної процедури можна за допомогою наступного запиту.

```
CALL sp_contract_total('2018-09-01', '2018-10-31');
```



contract_number	contract_date	SUM(supplied.supplied_amount)	SUM(supplied.supplied_amount * supplied.supplied_cost)
1	2018-09-01 00:00:00	47	39500.00
2	2018-09-10 00:00:00	24	11350.00
3	2018-09-23 00:00:00	148	99600.00
4	2018-09-24 00:00:00	119	76112.50
5	2018-10-02 00:00:00	64	45630.00

5 rows in set (0.01 sec)
Query OK, 0 rows affected (0.06 sec)

Рисунок 5.2

Наступна збережена процедура призначена для виконання різних операцій модифікації даних для таблиці contract. Дана процедура використовує оператор умови IF, призначений для управління потоком даних.

```

DELIMITER //
CREATE PROCEDURE sp_contract_ops(IN op CHAR(1), IN c_num INT, IN c_date TIMESTAMP,
                                IN s_id INT, IN c_note VARCHAR(100))
BEGIN
    IF op = 'i' THEN
        INSERT INTO contract(contract_date, supplier_id, contract_note)
            VALUES(CURRENT_TIMESTAMP(), s_id, c_note);
    ELSEIF op = 'u' THEN
        UPDATE contract SET contract_date = c_date,
                            supplier_id = s_id,
                            contract_note = c_note
        WHERE contract_number = c_num;
    ELSE
        DELETE FROM contract WHERE contract_number = c_num;
    END IF;
END //

```

Наступний запит дозволяє створювати договір (рисунок 5.3).

```
CALL sp_contract_ops('i', 0, '2018-12-16', 2, 'contract inserted');
```

XAMPP for Windows - mysql -u root -p

```

MariaDB [supply]> CALL sp_contract_ops('i', 0, '2018-12-16', 2, 'contract inserted');
Query OK, 1 row affected (0.01 sec)

MariaDB [supply]> select * from contract;

```

contract_number	contract_date	supplier_id	contract_note
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018
3	2018-09-23 00:00:00	3	Order 56 on 28.08.2018
4	2018-09-24 00:00:00	2	Order 74 on 11.09.2018
5	2018-10-02 00:00:00	2	Invoice 09-12 on 21.09.2018
6	2018-12-27 13:10:43	2	contract inserted

6 rows in set (0.00 sec)

Рисунок 5.3

Наступний запит дозволяє модифікувати договір (рисунок 5.4).

```
CALL sp_contract_ops('u', 6, '2018-12-31', 2, 'contract updated');
```

XAMPP for Windows - mysql -u root -p

```

MariaDB [supply]> CALL sp_contract_ops('u', 6, '2018-12-31', 2, 'contract updated');
Query OK, 1 row affected (0.01 sec)

MariaDB [supply]> select * from contract;

```

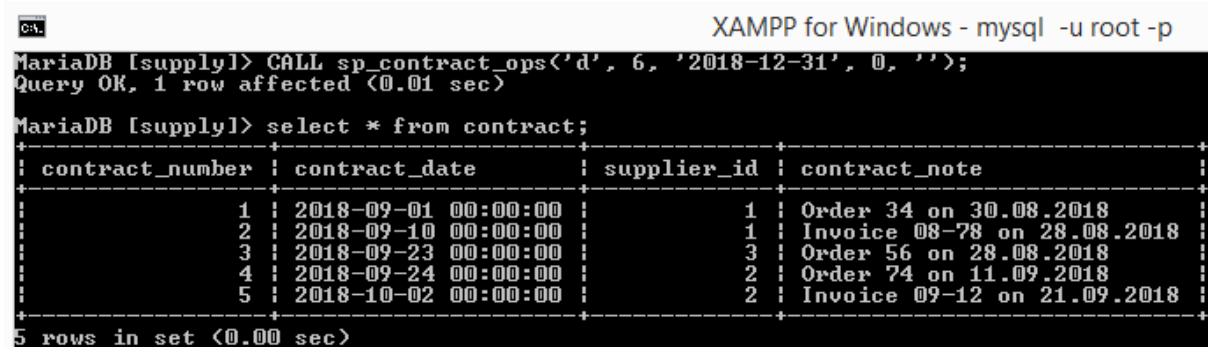
contract_number	contract_date	supplier_id	contract_note
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018
3	2018-09-23 00:00:00	3	Order 56 on 28.08.2018
4	2018-09-24 00:00:00	2	Order 74 on 11.09.2018
5	2018-10-02 00:00:00	2	Invoice 09-12 on 21.09.2018
6	2018-12-31 00:00:00	2	contract updated

6 rows in set (0.00 sec)

Рисунок 5.4

Наступний запит дозволяє видаляти договір (рисунк 5.5).

```
CALL sp_contract_ops('d', 6, '2018-12-31', 0, '');
```



The screenshot shows a MySQL command prompt window titled "XAMPP for Windows - mysql -u root -p". The user is logged in as root. The prompt shows the following commands and results:

```
MariaDB [supply]> CALL sp_contract_ops('d', 6, '2018-12-31', 0, '');
Query OK, 1 row affected (0.01 sec)

MariaDB [supply]> select * from contract;
```

contract_number	contract_date	supplier_id	contract_note
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018
3	2018-09-23 00:00:00	3	Order 56 on 28.08.2018
4	2018-09-24 00:00:00	2	Order 74 on 11.09.2018
5	2018-10-02 00:00:00	2	Invoice 09-12 on 21.09.2018

5 rows in set (0.00 sec)

Рисунок 5.5

2. Створення та використання тригерів

Припустимо, що при вводі даних у таблицю contract, у якій зберігається інформація про договори на постачання продукції, поле contract_date, у якому зберігається дата укладення договору, повинне бути обов'язково заповнене. При чому у випадку, якщо при вводі нового договору дане поле залишається незаповненим, в нього повинна бути автоматично записана поточна дата. Дану задачу можна вирішити за допомогою створення певного тригера, використовуючи відповідну команду CREATE TRIGGER (рисунк 5.6).

```
DELIMITER //
CREATE TRIGGER not_null_date BEFORE INSERT ON contract
FOR EACH ROW
BEGIN
    IF NEW.contract_date IS NULL THEN
        SET NEW.contract_date = CURRENT_TIMESTAMP();
    END IF;
END //
```

Для перевірки роботи тригера необхідно додати новий договір за допомогою наступного запиту.

```
INSERT INTO contract (supplier_id, contract_note) VALUES (1, '');
```

```

XAMPP for Windows - mysql -u root -p
MariaDB [supply]> INSERT INTO contract (supplier_id, contract_note) VALUES (1, '');
Query OK, 1 row affected (0.01 sec)

MariaDB [supply]> select * from contract;
+-----+-----+-----+-----+
| contract_number | contract_date | supplier_id | contract_note |
+-----+-----+-----+-----+
| 1 | 2018-09-01 00:00:00 | 1 | Order 34 on 30.08.2018 |
| 2 | 2018-09-10 00:00:00 | 1 | Invoice 08-78 on 28.08.2018 |
| 3 | 2018-09-23 00:00:00 | 3 | Order 56 on 28.08.2018 |
| 4 | 2018-09-24 00:00:00 | 2 | Order 74 on 11.09.2018 |
| 5 | 2018-10-02 00:00:00 | 2 | Invoice 09-12 on 21.09.2018 |
| 7 | 2018-12-27 13:30:04 | 1 | |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

Рисунок 5.6

В базі даних зберігається як загальна інформація про постачальників, так і інформація, яка відноситься тільки до фізичних або юридичних осіб. Одночасна наявність даних про постачальника у таблицях `supplier_org` та `supplier_person` не допускається з точки зору логіки управління бізнесом. Таким чином, виникає необхідність складного контролю відношень посилкової цілісності. Для вирішення даної задачі створимо тригер, який при введенні інформації у таблицю `supplier_person` буде контролювати наявність коду відповідного постачальника у таблиці `supplier_org` та блокувати введення даних про постачальника як про фізичну особу у тому випадку, якщо вже наявні дані про даного постачальника як про юридичну особу (рисунок 5.7).

```

DELIMITER //
CREATE TRIGGER check_supplier_org BEFORE INSERT ON supplier_person
FOR EACH ROW
BEGIN
    IF NEW.supplier_id IN (SELECT supplier_id FROM supplier_org) THEN
        SET @message = CONCAT('The person with id ', NEW.supplier_id,
            ' is already stored as the organization!');
        SIGNAL SQLSTATE '45001';
        SET MESSAGE_TEXT = @message;
    END IF;
END //

```

Для перевірки роботи тригеру необхідно спробувати додати дані про постачальника 2 (який вже зберігається у БД в якості юридичної особи) як про фізичну особу.

```

INSERT INTO supplier_person VALUES (2, 'Makarov', 'Oleg', 'Petrovych');

```

```
XAMPP for Windows - mysql -u root -p
MariaDB [supplyl]> INSERT INTO supplier_person VALUES (2, 'Makarov', 'Oleg', 'Petrovych');
ERROR 1644 (45001): The person with id 2 is already stored as the organization!
MariaDB [supplyl]> select * from supplier_person;
+-----+-----+-----+-----+
| supplier_id | supplier_last_name | supplier_first_name | supplier_middle_name |
+-----+-----+-----+-----+
| 1 | Petrov | Pavlo | Petrovych |
| 3 | Ivanov | Illia | Illych |
| 5 | Sydorov | Serhii | Stepanovych |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Рисунок 5.7

Для видалення збережених процедур та тригерів необхідно скористатися операторами DROP PROCEDURE та DROP TRIGGER відповідно.

3. Оформити звіт з лабораторної роботи

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

4. Питання для самоконтролю

1. Що таке збережена процедура?
2. Назвати переваги використання збережених процедур.
3. Який оператор використовується для створення збереженої процедури?
4. Яким чином можна визначити вхідні або вихідні параметри збереженої процедури?
5. Для чого використовується оператор IF?
6. Яке призначення операторів BEGIN та END?
7. Що таке тригер?
8. Назвати переваги використання тригерів.
9. За допомогою якого оператора тригер зв'язується з таблицею?
10. До яких подій, пов'язаних зі зміною вмісту таблиці, можна прив'язати тригер?
11. Яким чином можна визначити до чи після операції зміни вмісту таблиці повинен спрацьовувати тригер?
12. Для чого використовуються префікси NEW та OLD?
13. Яке призначення оператора SET?
14. За допомогою яких операторів виконується видалення процедур та тригерів?