

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра систем штучного інтелекту



ЗВІТ № 10 з

курсу “ОБДЗ”

на тему:

«Написання збережених процедур на мові SQL»

Виконав:

студент групи КН-208

Фіняк М.В

Викладач:

Якимишин Х.М.

Лабораторна робота № 10

Мета роботи: навчитися розробляти та виконувати збережені процедури та функції у MySQL.

Короткі теоретичні відомості

Більшість СУБД підтримують використання збережених послідовностей команд для виконання часто повторюваних, однотипних дій над даними. Такі збережені процедури дозволяють спростити оброблення даних, а також підвищити безпеку при роботі з базою даних, оскільки в цьому випадку прикладні програми не потребують прямого доступу до таблиць, а отримують потрібну інформацію через процедури. СУБД MySQL підтримує збережені процедури і збережені функції. Аналогічно до вбудованих функцій (типу COUNT), збережену функцію викликають з деякого виразу і вона повертає цьому виразу обчислене значення. Збережену процедуру викликають за допомогою команди CALL. Процедура повертає значення через вихідні параметри, або генерує набір даних, який передається у прикладну програму.

Синтаксис команд для створення та збережених процедур :

CREATE

[DEFINER = { користувач | CURRENT_USER }]

FUNCTION назва_функції ([параметри_функції ...])

RETURNS тип

[характеристика ...] тіло_функції

CREATE

[DEFINER = { користувач | CURRENT_USER }]

PROCEDURE назва_процедури ([параметри_процедури ...])

[характеристика ...] тіло_процедури

Аргументи:

DEFINER

Задає автора процедури чи функції. За замовчуванням – це CURRENT_USER.

RETURNS

Вказує тип значення, яке повертає функція.

тіло_функції, тіло_процедури

Послідовність директив SQL. В тілі процедур і функцій можна оголошувати локальні змінні, використовувати директиви BEGIN ... END, CASE, цикли тощо. В тілі процедур також можна виконувати транзакції. Тіло функції обов'язково повинно містити команду RETURN і повертати значення.

- **параметри_процедури:**

[IN | OUT | INOUT] ім'я_параметру тип Параметр, позначений як IN, передає значення у процедуру. OUT-параметр передає значення у точку виклику процедури. Параметр, позначений як INOUT, задається при виклику, може бути змінений всередині процедури і зчитаний після її завершення. Типом параметру може бути будь-який із типів даних, що підтримується MySQL.

- **параметри_функції:**

ім'я_параметру тип

У випадку функцій параметри використовують лише для передачі значень у функцію.

При створенні процедур і функцій можна вказувати їхні додаткові характеристики.

- **характеристика:**

LANGUAGE SQL

| [NOT] DETERMINISTIC

| {CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA}

| SQL SECURITY {DEFINER | INVOKER}

| COMMENT 'короткий опис процедури'

DETERMINISTIC

Вказує на те, що процедура обробляє дані строго визначеним (детермінованим) чином. Тобто, залежно від вхідних даних, процедура повертає один і той самий результат. Недетерміновані процедури містять функції типу NOW() або RAND(), і результат їх виконання не можна передбачити. За замовчуванням всі процедури і функції є недетермінованими.

CONTAINS SQL | NO SQL

Вказує на те, що процедура містить (за замовчуванням), або не містить директиви SQL.

READS SQL DATA

Вказує на те, що процедура містить директиви, які тільки зчитують дані з таблиць.

MODIFIES SQL DATA

Вказує на те, що процедура містить директиви, які можуть змінювати дані в таблицях.

SQL SECURITY

Задає рівень прав доступу, під яким буде виконуватись процедура.

DEFINER – з правами автора процедури (задано за замовчуванням),

INVOKER – з правами користувача, який викликає процедуру. Щоб запускати збережені процедури і функції, користувач повинен мати права EXECUTE.

При створенні процедур і функцій у командному рядку клієнта MySQL, потрібно перевизначити стандартний символ завершення вводу директив ";", щоб мати можливість ввести всі директиви процедури. Це робиться за допомогою команди DELIMITER. Наприклад,
DELIMITER | означає, що завершення вводу процедури буде позначатись символом "|".

Нижче наведено синтаксис додаткових директив MySQL, які дозволяють розробляти нескладні програми на мові SQL.

DECLARE *назва_змінної* *тип_змінної*
[DEFAULT *значення_за_замовчуванням*]
Оголошення змінної заданого типу.

SET *назва_змінної* = *вираз*
Присвоєння змінній значення.

IF умова THEN директиви

[ELSEIF умова THEN директиви] ...

[ELSE директиви2]

END IF

Умовний оператор. Якщо виконується вказана умова, то виконуються відповідні їй директиви, в протилежному випадку виконуються директиви2.

CASE вираз

WHEN значення1 THEN директиви1

[WHEN значення2 THEN директиви2] ...

[ELSE директиви3]

END CASE

Оператор умовного вибору. Якщо вираз приймає значення1, виконуються директиви1, якщо приймає значення2 – виконуються директиви2, і т.д.

Якщо вираз не прийме жодного зі значень, виконуються директиви3.

[мітка:] LOOP директиви

END LOOP

Оператор безумовного циклу. Вихід з циклу виконується командою LEAVE мітка.

REPEAT

директиви UNTIL

умова

END REPEAT

WHILE умова DO директиви

END WHILE

Оператори REPEAT і WHILE дозволяють організувати умовні цикли, які завершуються при виконанні деякої умови.

Хід роботи

1. Функції шифрування і дешифрування із заданим ключем.

```
CREATE FUNCTION encode (password CHAR(48)) RETURNS  
TINYBLOB RETURN AES_ENCRYPT(password, 'key-key');
```

```
CREATE FUNCTION decode (password TINYBLOB) RETURNS  
CHAR(48) RETURN AES_DECRYPT(password, 'key-key');
```

2. Перевіримо роботу створених функцій.

```
SELECT email, password, encode(password) AS encode,  
decode(encode(password)) AS decode FROM restaurant.customer ;
```

	email	password	encode	decode
►	yulilav@gmail.com	yulilav123	BLOB	yulilav123
	iryna123@gmail.com	iryna27	BLOB	iryna27
	olena111@gmail.com	mydream111	BLOB	mydream111
	olexandra17@gmail.com	oleksadyp10	BLOB	oleksadyp10
	anna31@gmail.com	annaKvittkova12	BLOB	annaKvittkova12

3. Створимо процедуру повинна рахувати кількість замовлень клієнта зроблених за певний проміжок часу кожним з існуючих працівників. Для цього потрібно відібрати всі замовлення та працівників, що їх виконували за клієнтом та часом оформлення.

Потім згрупувати вибрані замовлення за працівниками та порахувати кількість замовлень.

У процедуру потрібно передати ім'я клієнта, а також першу і другу дату. Перед основними директивами додамо перевірку коректності задання початкової і кінцевої дати (IF date1<=date2 THEN...). Результати обчислень будуть записуватись у таблицю stats, яку процедура завжди очищує (командою TRUNCATE restaurant.stats) і заповнює з нуля.

```
DELIMITER //

CREATE PROCEDURE or_count (IN name VARCHAR(100), IN
date1 DATE, IN date2 DATE)
BEGIN
    DECLARE error VARCHAR(30);
    SET error = 'Invalid date entered';
    IF (date1<=date2) THEN
        BEGIN
            CREATE TABLE IF NOT EXISTS restaurant.stats
(name_employee VARCHAR(100), amount INT UNSIGNED);
            TRUNCATE restaurant.stats;
            INSERT INTO restaurant.stats SELECT staff.first_name AS
name_employee,
            COUNT(restaurant.order.id) AS amount
            FROM (customer INNER JOIN restaurant.order) INNER JOIN
staff
            ON customer.first_name=name
            AND customer.id=restaurant.order.customer_id
            AND restaurant.order.employee_id=staff.id
            WHERE restaurant.order.date BETWEEN date1 AND date2
            GROUP BY name_employee;
        END
    ELSE
        SET error = 'Invalid date entered';
    END IF;
END
```



```

END;
ELSE SELECT error;
END IF;
END//
DELIMITER ;

```

4.Перевіримо роботу створеної процедури :

```

CALL or_count('Olena', '2020-03-10', '2020-03-22');
SELECT * FROM stats;

```

Результат роботи процедури – таблиця stats:

	name_employee	amount
▶	Viktor	1
	Anastasia	2
	Andriy	1

```
CALL or_count('Olena', '2020-04-10', '2020-03-22');
```

Результат виклику процедури:

	error
▶	Invalid date entered

5. Створимо процедуру, яка буде рахувати кількість інгредієнтів для вказаної страви. Для цього згрупуємо інгредієнти за стравою та порахуємо їх кількість.

У процедуру потрібно передати номер потрібної страви.

```

DELIMITER //
CREATE PROCEDURE ing (IN id1 INT )
BEGIN
    DECLARE error VARCHAR(30);
    SET error = 'Invalid date entered';
    BEGIN
        CREATE TABLE IF NOT EXISTS restaurant.ing (id_dish INT
UNSIGNED, amount INT UNSIGNED);
        TRUNCATE restaurant.ing;
        INSERT INTO restaurant.ing SELECT dish_id1 AS id_dish,
COUNT(ingredient_dish.ingredient_id) AS amount
FROM restaurant.ingredient_dish
        WHERE ingredient_dish.dish_id1 = id1
        GROUP BY id_dish;
    END;
END//
DELIMITER ;

```

6. Перевіримо роботу створеної процедури :

```

CALL ing(7);
SELECT * FROM ing;

```

Результат виклику процедури:

	id_dish	amount
▶	7	4

Висновок : під час виконання даної лабораторної роботи я навчився розробляти та виконувати збережені процедури та функції у MySQL.

