МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"
ІНСТИТУТ ПІСЛЯДИПЛОМНОЇ ОСВІТИ
КАФЕДРА СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №10

Підготувала:

Студентка групи КН-209

Кульчицька Олена

Викладач:

Мельникова Н.І.

2020

Лабораторна робота №10

на тему:

“Написання збережених процедур на мові SQL”

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 збережених процедур описано нижче.

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 транзакії. 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.

Запит:

```
USE Confectionary;

CREATE FUNCTION pass_encode (password VARCHAR(20))
RETURNS TINYBLOB
RETURN AES_ENCRYPT(password, 'key-key');

CREATE FUNCTION pass_decode (password TINYBLOB)
RETURNS VARCHAR(20)
RETURN AES_DECRYPT(password, 'key-key');
```

Перевіримо роботу створених функцій.

Запит:

```
SELECT email, password,  pass_encode(password) AS encode,
pass_decode(pass_encode(password)) AS decode
FROM Confectionary.customer ;
```

Результат запиту:

email	password	encode	decode
olink@gmail.com	olena123	BLOB	olena123
linavoo@gmail.com	lina123	BLOB	lina123
lina@gmail.com	boor123	BLOB	boor123
olickv@gmai.com	olia3	BLOB	olia3
robrun@gmai.com	robrob	BLOB	robrob
lina@gmail.com	larala1	BLOB	larala1
olickv@gmai.com	markhu3	BLOB	markhu3
robrun@gmai.com	ian8	BLOB	ian8
lilv@gmail.com	lilvraf9	BLOB	lilvraf9

2. 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 status, яку процедура завжди очищує (командою TRUNCATE confectionary.status) і заповнює з нуля.

```
DELIMITER |
CREATE PROCEDURE order_count (IN name VARCHAR(45), 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 confectionary.status (name_employee VARCHAR(100), amount INT UNSIGNED);
    TRUNCATE confectionary.status;
    INSERT INTO confectionary.status SELECT staff.first_name AS name_employee, COUNT(confectionary.order.id) AS amount
    FROM (confectionary.customer INNER JOIN confectionary.order) INNER JOIN confectionary.staff
    ON customer.first_name=name
    AND customer.id=confectionary.order.customer_id
    AND confectionary.order.staff_id=staff.id
    WHERE confectionary.order.date BETWEEN date1 AND date2
    GROUP BY name_employee;
  END;
  ELSE SELECT error;
  END IF;
END|
DELIMITER ;
```

Перевіримо роботу створеної процедури .

Запит виклику процедури:

```
CALL order_count('Olena', '2020-03-10', '2020-03-22');
SELECT * FROM confectionary.status;
```

Результат виклику процедури:

	name_employee	amount
	Alex	1

Запит виклику процедури:

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

Результат виклику процедури:

	error
	Invalid date entered

3. 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 confectionary.ing (dish_id INT UNSIGNED, amount INT UNSIGNED);
    TRUNCATE confectionary.ing;
    INSERT INTO confectionary.ing SELECT id_dish AS dish_id, COUNT(ingredient_dish.ingredient_id) AS amount
    FROM confectionary.ingredient_dish
    WHERE ingredient_dish.id_dish = id1
    GROUP BY dish_id;
  END;
END//
DELIMITER ;
```

Запит виклику процедури:

```
CALL ing(1);
SELECT * FROM ing;
```

Результат виклику процедури:

	dish_id	amount
	1	3

Висновок: на цій лабораторній роботі я навчився розробляти та використовувати збережені процедури і функції у СУБД MySQL.