МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра систем штучного інтелекту

Звіт до практичної роботи №10

з дисципліни “ ОБДЗ ”

На тему: «Написання збережених процедур на мові SQL»

Виконав:
ст. гр. КН-211
Качмарик Віктор
Викладач:
Якимишин Х. М.

Львів – 2020

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

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/дешифрування із заданим ключем.

```
CREATE FUNCTION myPass_encode(password CHAR(100))
RETURNS BLOB
DETERMINISTIC
RETURN AES_ENCRYPT(password, 'pharmacy_pass');


CREATE FUNCTION myPass_decode(password BLOB)
RETURNS CHAR(100)
DETERMINISTIC
RETURN AES_DECRYPT(password, 'pharmacy_pass');
```

2. Процедура повинна обраховувати, скільки покупець витрачає в певних аптеках за пивний проміжок часу.
   Перед основними директивами додамо перевірку коректності задання початкової і кінцевої дати (IF date1<=date2 THEN…). Результати обчислень будуть записуватись у таблицю pharInfo, яку процедура завжди очищує (командою TRUNCATE pharInfo) і заповнює з нуля.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `myPhar`(my_login varchar(255), date_1 DATE, IN date_2 DATE)
    DETERMINISTIC
BEGIN
    if date_1 < date_2 then
    begin
        create table if not exists pharInfo(first_name varchar(255), last_name varchar(255), pharmacy varchar(255), cost int);
        TRUNCATE pharInfo;
        INSERT INTO pharInfo SELECT customer.first_name AS first_name,
        customer.last_name AS last_name,
        pharmacy.pharmacy_name AS pharmacy,
        max(order_item.quantity*medicine.price) as order_id
        FROM (((customer INNER JOIN `order`) INNER JOIN order_item) INNER JOIN medicine) INNER JOIN pharmacy
        ON customer.login = my_login
        AND customer.customer_id = `order`.customer_id
        AND `order`.order_id = order_item.order_id
        AND medicine.medicine_id = order_item.medicine_id
        AND pharmacy.pharmacy_id = `order`.pharmacy_id
        WHERE `order`.order_date BETWEEN date_1 AND date_2;
    end;
    end if;
END
```

3. Після створення функцій і процедури перевіримо їх роботу:

```
SELECT first_name, login, myPass_encode(customer.password) AS password
FROM customer;
```

Результат виконання функції кодування:

first_name	login	password
Viktor	user_vitya	0x68EDDE9CF34DAA53A2C175568E952966
Andrew	andreeew	0x28E6C070B4BA0AA41D5C3847FD1A733C
Vlad	user05_vlad	0x1EA9E26AD8988C9BB4B01F4F1216B399
Max	max_kachmaryk	0x2A9F42330DE85A6E405151B658495E4E
Olesya	olesia.marko	0xB978A4CCB2AF898618A44F0BD1F2FB67
Yuliana	userq_lavrino	0xEE93E619BD2AF9F8F2FAE4BA6BF7A63B
Roman	25user_roman	0x7BB739C2F6B0D1FB170C60FF65D02C2B
Petro	klloiak	0xDBF8DF7CDEAE7E890E1EAE486A443136
Sasha	user001_sasha	0x20525237C13156C225591537F9133258
Yuriy	superuser_yuriy	0x4DB2172CE2E65E63C0599A88EE6DD637

```
SELECT first_name, login, myPass_decode(password) AS password
FROM customer1;
```

Результат виконання функції декодування:

first_name	login	password
Viktor	user_vitya	viktor
Andrew	andreeew	andrew
Vlad	user05_vlad	vlad
Max	max_kachmaryk	max
Olesya	olesia.marko	jlesya
Yuliana	userq_lavrino	yuliana
Roman	25user_roman	roman
Petro	klloiak	petro
Sasha	user001_sasha	sasha
Yuriy	superuser_yuriy	yuriy

```
CALL myPhar('andreeew', '20180101', '20200401');

SELECT * FROM pharInfo;
+------------+-----------+----------+------+
| first_name | last_name | pharmacy | cost |
+------------+-----------+----------+------+
| Andrew     | Ilkiv     | Phar3    |  244 |
+------------+-----------+----------+------+
```

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