

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту



Звіт до лабораторної роботи №10

з дисципліни

“Організація Баз Даних та знань”

Виконала:

ст. гр. КН-210
Заремба Вікторія

Викладач:

Мельникова Н.І.

Лабораторна робота №10
з курсу “ОБДЗ”
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 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

Для досягнення мети роботи, реалізуємо 3 запити до бази даних.

1. Написати функції шифрування/дешифрування із заданим ключем
2. Написати процедури виклику кількості завдань по департаментах на обраному проекті за певний проміжок часу (по дедлайнах).
3. Написати процедуру всіх активних завдань заданого учасника.

1. Написати функції шифрування/дешифрування із заданим ключем

```
#функції кодування і декодування паролю
CREATE FUNCTION pm_system_encode (pass VARCHAR(99))
  RETURNS TINYBLOB
  RETURN AES_ENCRYPT(pass, 'key-key');

CREATE FUNCTION pm_system_decode (pass TINYBLOB)
  RETURNS VARCHAR(48)
  RETURN AES_DECRYPT(UNHEX(pass), 'key-key');

SET GLOBAL log_bin_trust_function_creators = 1;

#кодування паролю таблиці user
UPDATE user SET password = HEX(pm_system_encode(password));
SELECT surname, password FROM user;

#запит декодованого паролю
SELECT surname, pm_system_decode(UNHEX(password)) FROM user;
SELECT surname, password FROM user;

#декодування паролю
UPDATE user SET password = pm_system_decode(UNHEX(password));
SELECT surname, password FROM user;
```

Результат запиту після кодування:

	surname	password
1	Zaremba	8FE54287AE09F25DFAF6B80F05752BD6
2	Vyshnevskia	8938E74E8E3F9BB7E42F027745FBA166
3	Boyechko	2A61C1D37D4BFD111CA984169028E1C8
4	Popova	8D2A74A65DB515594944FC6619717453
5	Kondratyk	FA8E95D0B06EA517D94FFB0CC69AEBEC
6	Rogynski	F8874C54DA952B916052F4247A020D0E
7	Osovska	22AA5173F8CBF81FA37CCBF15C93D70C
8	Мацків	82EA6905C33682E5634261E171F9EFF7
9	Корінь	9F8409EC30FD221A1506E1D421F51661
10	Осташевська	092FC3C24C2F3602450798E0CDC83017
11	Драбик	2DF2ECBA8D258E8771438C3FC48127DA
12	Зубик	085E1109B58B142AC5A7B0EC9BF630EF

Результат запита з декодуванням:

	surname	`mycms_decode(UNHEX(password))`
1	Zaremba	12345678
2	Vyshnevskia	jlkdc
3	Boyechko	vdsvzc
4	Popova	f8uejiosdak
5	Kondratyk	cdjvx
6	Rogynski	jocs,lm
7	Osovska	uhdosj.k
8	Мацків	ujdscdosj.k
9	Корінь	udjsksj.k
10	Осташевська	uhvdjkj.k
11	Драбик	uhdosj.kjdj
12	Зубик	uhdokdjv.k

2. Написати процедури виклику кількості завдань по департаментах на обраному проєкті за певний проміжок часу (по дедлайнах).

```

DELIMITER //
CREATE PROCEDURE task_count(IN name VARCHAR(30), IN date1 DATE, IN date2 DATE)
BEGIN
    DECLARE error VARCHAR(25);
    SET error = 'Неправильні вхідні дані';
    IF (date1 <= date2) THEN
        BEGIN
            CREATE TABLE IF NOT EXISTS pm_system.dep_tasks (department_name VARCHAR(20), number
INT UNSIGNED) char set UTF8MB4;
            TRUNCATE pm_system.dep_tasks;
            INSERT INTO pm_system.dep_tasks SELECT pm_system.department.dep_name AS department_name,
COUNT(pm_system.task.id_task) AS number
FROM (project INNER JOIN department) INNER JOIN task
ON project_name = name
AND project.id_project = department.id_project
AND department.id_department = task.id_department
WHERE deadline BETWEEN date1 AND date2
GROUP BY department_name;
        END;
    ELSE SELECT error;
    END IF;
END//
DELIMITER;

```

Запит 1:

```

CALL task_count('Summer Camp', '2020-05-01', '2020-07-15');
SELECT * FROM pm_system.dep_tasks;

```

Результат запиту 1:

	department_name	number
1	звязківство	1
2	інтендант	2
3	бунчужна	1
4	господарник	1
5	писар	1

Запит2:

```
CALL task_count('Summer Camp', '2020-08-01', '2020-07-15');
```

Результат запиту 2:

error
1 Неправильні вхідні дані

3. Написати процедуру всіх активних завдань заданого учасника.

```
DELIMITER //
CREATE PROCEDURE show_user_tasks(IN user_surname VARCHAR(30))
BEGIN
    CREATE table if not exists user_tasks (u_task VARCHAR(20), t_deadline DATE) char set UTF8MB4;
    TRUNCATE pm_system.user_tasks;
    INSERT INTO pm_system.user_tasks SELECT pm_system.task.task_name as u_task,
    pm_system.task.deadline as t_deadline
    FROM ((pm_system.user INNER JOIN user_department) INNER JOIN task_user_department) INNER JOIN
task
    ON surname = user_surname
    AND user.id_user = user_department.id_user
    AND user_department.id_user_department = task_user_department.id_user_department
    AND task_user_department.id_task = task.id_task
    WHERE status = 1;
    SELECT * FROM user_tasks;
end //
DELIMITER ;
```

Виклик процедури:

```
CALL show_user_tasks('Zaremba');
```

Результат виклику процедури:

	u_task	t_deadline
1	закупити реманент	2020-06-20

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