

МІНІСТЕРСТВО ОСВІТИ І НАУКИ НАЦІОНАЛЬНИЙ  
УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



**Лабораторна робота №10**  
**З дисципліни «Організація баз даних та знань»**

**Виконав:**  
студент групи КН-210  
Черещук Любомир

**Перевірів:**  
Кандидат тех. наук, ст. викладач  
Мельникова Н. І.

Львів – 2020

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

### 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.

### Хід роботи

1. Створив функції кодування та декодування прізвища, для цього виконав такі скрипти у MySQL.

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

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

Перевіряю правильність роботи функцій:

```
select id, name, mycms_encode(surname)  
from sportproducts.user;
```

	id	name	mycms_encode(surname)
▶	1	Ivan	BLOB
	2	Igor	BLOB
	3	Mark	BLOB
	4	Max	BLOB
	5	Ana	BLOB
	6	Bill	BLOB

```
select id, name, mycms_decode(tmptbl.code)  
from (select id, name, mycms_encode(surname) as code from sportproducts.user) as tmptbl;
```

	id	name	mycms_decode(tmptbl.code)
▶	1	Ivan	Mortu
	2	Igor	Smit
	3	Mark	Tomas
	4	Max	Grein
	5	Ana	Layl
	6	Bill	Drago

Спочатку я закодував поле (surname) , а потім декодував його.

2. Написав процедуру `get_by_date`, яка приймає як вхідні параметри 2 дати, як проміжок часу, і виводить усі замовлення в цьому проміжку в форматі(ім'я користувача, дата замовлення, назва товару).

Код процедури:

```
DELIMITER //

create procedure get_by_date (in date1 date, in date2 date)
begin
    declare error char(45);
    set error = 'Некоректно задані дати';
    if (date1 <= date2) then
        begin
            create table if not exists sportproducts.orders_by_date (username char(45),
            orderdate date, productname char(45));
            truncate sportproducts.orders_by_date;
            insert into sportproducts.orders_by_date
            select user.name as username, `order`.datetime as orderdate, product.name as productname from
            user
                inner join `order`
                    on user.id = `order`.User_id
                inner join order_product
                    on `order`.id = order_product.Order_id
                inner join product
                    on order_product.Product_id = product.id
            where `order`.datetime between date1 and date2;
        end;
    else select error;
    end if;
end

Далі викликаю цю процедуру:

call get_by_date('2020-03-20', '2020-03-22');

select * from orders_by_date;
```

Результат:

	username	orderdate	productname
►	Ivan	2020-03-20	tshort
	Max	2020-03-20	shorts
	Ivan	2020-03-21	sportshoes
	Igor	2020-03-22	protein
	Max	2020-03-20	shorts

Якщо перша дата буде більшою, ніж друга, тоді проміжок часу буде від'ємним, і це некоректні дані, для цього я передбачив виклик помилки з текстом: «Некоректно задані дати». :

```
call get_by_date('2020-03-25', '2020-03-22');
```

	error
►	Некоректно задані дати

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