МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота №7**

з дисципліни « Реляційні бази даних »

*назва дисципліни*

на тему: «ЗАХИСТ БД ВІД НЕСАНКЦІОНОВАНОГО ДОСТУПУ»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Я.І.

(прізвище й ініціали студента)

Прийняв: канд.тех.наук, каф. 603

Манжос Ю.С.

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2024

ПОСТАНОВКА ЗАДАЧІ

1. Слід створиту копію своєї БД.
2. Для кожного з варіантів Вашу БД можливо подати як сукупність трьох таблиць А –АВ—В. Слід створити кілька типів користувачів БД: «гість» може тільки читати дані;

* «операторА» може читати, писати, модифікувати та знищувати дані у таблиці А;
* «операторВ» може читати, писати, модифікувати та знищувати дані у таблиці В;
* «операторАВ» може читати, писати, модифікувати та знищувати дані у таблиці АВ, а також тільки читати дані з таблиць А та В;

1. Кожен з користувачів повинен мати своє ім’я та пароль, що Користувач має змінити відразу після входження у систему.

Це створить перший рівень захисту від несанкціонованого доступу до БД.

1. На другому рівні захисту даних слід додати «Шифрування- Дешифрування», а дані зберігати у таблицях зашифрованими. Кожна з таблиць повінна мати свій пароль. Для доступу до даних слід створити відповідні збережені процедури/функції, якій й мають забезпечувати інтерфейс користувача.
2. Визначити склад системних процедур та функцій для шифрування даних.
3. Створити збережені процедури, що будуть шифрувати дані перед занесенням їх у таблиці нової БД.
4. Створити збережені функції, що будуть відображати дешифровані дані.
5. Перевірити коректніість шифрування-дешифрування
6. Оформити звіт.

**Варіант завдань – 10**

Для кожного з варіантів використати створену БД та забезпечити дворівневий захист даних від несанкціонованого доступу у відповідності з планом.

Виконання роботи

Створення копії незашифрованої БД

Для цього було використано наступні запити до БД

-- створюємо точні копії таблиць

CREATE TABLE prices LIKE `вартість бензину`;

CREATE TABLE sales LIKE `продаж`;

CREATE TABLE client LIKE `клієнт`;

-- копіюємо дані в нові таблиці

INSERT INTO prices SELECT \* FROM `вартість бензину`;

INSERT INTO sales SELECT \* FROM `продаж`;

INSERT INTO client SELECT \* FROM `клієнт`;

-- створюємо нову БД, яка буде мати дані в зашифрованному вигляді

create database lw07var10bk;

-- переносимо копії таблиць у нову БД

RENAME TABLE lw07var10.prices TO lw07var10bk.prices;

RENAME TABLE lw07var10.sales TO lw07var10bk.sales;

RENAME TABLE lw07var10.client TO lw07var10bk.client;

Cтворення користувачів БД.

Код створення ролей та користувачів

-- створюємо нові ролі

--CREATE ROLE Guest, OperatorP, OperatorC, OperatorSL;

CREATE ROLE 'Guest'@'localhost';

CREATE ROLE 'OperatorP'@'localhost';

CREATE ROLE 'OperatorC'@'localhost';

CREATE ROLE 'OperatorSL'@'localhost';

-- перевіріямо створення ролей

SELECT user FROM mysql.user;

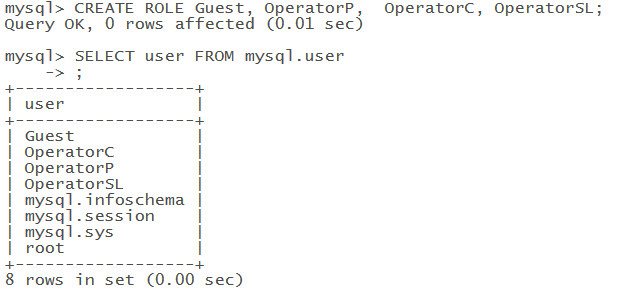


Рисунок 2 – перевірка створених ролей

Код створення тригера для таблиці **продаж**

-- створення нових користувачів

CREATE USER 'SimpleUser'@'localhost' PASSWORD EXPIRE DEFAULT;

CREATE USER 'UserP'@'localhost' PASSWORD EXPIRE DEFAULT;

CREATE USER 'UserC'@'localhost' PASSWORD EXPIRE DEFAULT;

CREATE USER 'UserSL'@'localhost' PASSWORD EXPIRE DEFAULT;

-- присвоєння паролей користувачам

ALTER USER 'SimpleUser'@'localhost' IDENTIFIED BY 'SimpleUser';

ALTER USER 'UserP'@'localhost' IDENTIFIED BY 'UserP';

ALTER USER 'UserC'@'localhost' IDENTIFIED BY 'UserC';

ALTER USER 'UserSL'@'localhost' IDENTIFIED BY 'UserSL';

-- надання привілегій ролям

GRANT SELECT ON lw07var10bk.\* TO 'Guest'@'localhost';

GRANT SELECT,INSERT, UPDATE, DELETE ON lw07var10bk.prices TO 'OperatorP'@'localhost';

GRANT SELECT,INSERT, UPDATE, DELETE ON lw07var10bk.client TO 'OperatorC'@'localhost';

GRANT INSERT, UPDATE, DELETE ON lw07var10bk.sales TO 'OperatorSL'@'localhost';

GRANT SELECT ON lw07var10bk.\* TO 'OperatorSL'@'localhost';

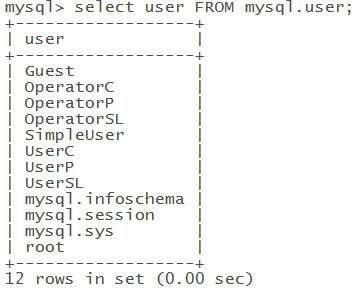


Рисунок 3 – перевірка створених користувачів

Надаємо ролі створеним користувачам в БД

FLUSH PRIVILEGES;

-- присвоєння ролей користувачам БД

Grant 'Guest'@'localhost' TO 'SimpleUser'@'localhost';

Set default role 'Guest'@'localhost' to 'SimpleUser'@'localhost';

Grant 'UserP'@'localhost' TO 'OperatorP'@'localhost';

Set default role 'UserP'@'localhost' to 'OperatorP'@'localhost';

Grant 'UserC'@'localhost' TO 'OperatorC'@'localhost';

Set default role 'UserC'@'localhost' to 'OperatorC'@'localhost';

ALTER USER 'SimpleUser'@'localhost' ACCOUNT UNLOCK;

ALTER USER 'UserP'@'localhost' ACCOUNT UNLOCK;

ALTER USER 'UserC'@'localhost' ACCOUNT UNLOCK;

ALTER USER 'UserSL'@'localhost' ACCOUNT UNLOCK;

Grant 'UserSL'@'localhost' TO 'OperatorSL'@'localhost';

Set default role 'UserSL'@'localhost' to 'OperatorSL'@'localhost';

SHOW GRANTS for Guest@localhost;

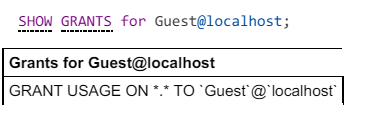


Рисунок 3 – результат перегляду привілегій Guest

Редагування привілегій ролі Guest

GRANT SELECT ON lw07var10bk.\* TO 'Guest' @'localhost';

SHOW GRANTS for 'Guest'@'localhost'

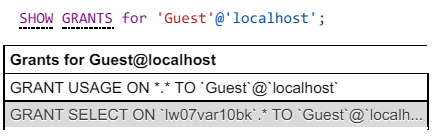


Рисунок 3 – результат редагування привілегій Guest

Шифрування записів в таблицях БД

Виконаємо запит для перевірки роботи тригера:

INSERT INTO `продаж` (`ДатаПродажу`, `НомерДисконту`, `МаркаБензину`, `КількістьЛітрів\_л`, `КодВартості`)

VALUES ('2024-04-30 12:00:00', 16, 'А-95', 50, 95);

INSERT INTO `продаж` (`ДатаПродажу`, `НомерДисконту`, `МаркаБензину`, `КількістьЛітрів\_л`, `КодВартості`)

VALUES ('2024-04-30 13:00:00', 17, 'А-92', 30, 92);

INSERT INTO `продаж` (`ДатаПродажу`, `НомерДисконту`, `МаркаБензину`, `КількістьЛітрів\_л`, `КодВартості`)

VALUES ('2024-04-30 14:00:00', 18, 'А-76', 20, 76);

SELECT \* FROM `вартість бензину`;

Рисунок 3 – результат виконання тригера з видалення старих записів продажів

Здійснити денормализацію «униз» однієї з таблиць, додавши у неї поле та забезпечити коректність інформації цього поля за допомогою тригерів.

створимо стовпець, який буде рахувати період, скільки людина є клієнтом в компанії. Для цього створимо стовпець командою:

ALTER TABLE `клієнт`

ADD `КількістьДнівКлієнт` INT;

Створимо тригер, який буде оновлювати значення в стовпчику при додаванні нового запису в таблицю клієнт

DELIMITER //

CREATE TRIGGER `update\_days\_clientship`

BEFORE INSERT ON `клієнт`

FOR EACH ROW

BEGIN  
 UPDATE `клієнт` SET `КількістьДнівКлієнтства` = DATEDIFF(NOW(), `DateGiven`);

SET NEW.`КількістьДнівКлієнт` = DATEDIFF(NOW(), NEW.`DateGiven`);

END//

DELIMITER ;

Для перевірки тригера виведемо по 5 клієнтів, які найдовше в компанії, та найновіші.

SELECT `idAccount`, `LastName`, `FirstName`, `КількістьДнівКлієнт`

FROM `клієнт`

ORDER BY `КількістьДнівКлієнт` DESC

LIMIT 5;

SELECT `idAccount`, `LastName`, `FirstName`, `КількістьДнівКлієнт`

FROM `клієнт`

ORDER BY `КількістьДнівКлієнт` ASC

LIMIT 5;

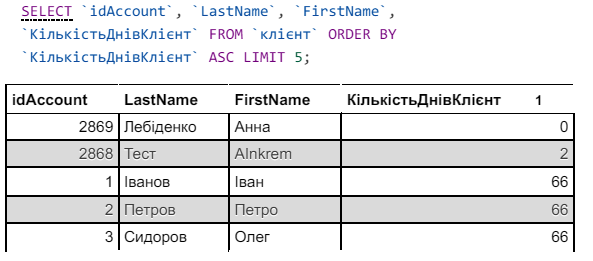


Рисунок 5 – результат виконання тригера з вирахування днів клієнтсва

За допомогою тригерів до однієї з таблиць додати журнал, в якому повинні бути зафиксировані відомості про дії користувачів над цією таблицею.

Створимо таблицю для журналу та тригер командою:

Create table журнал\_дій ( дія VARCHAR(20), час date);

DELIMITER //

CREATE OR REPLACE TRIGGER trigger\_after\_update\_client

AFTER UPDATE ON `клієнт`

FOR EACH ROW

BEGIN

INSERT INTO `журнал\_дій\_клієнт` (`дія`, `час`)

VALUES ('UPDATE', NOW());

END;

//

DELIMITER ;

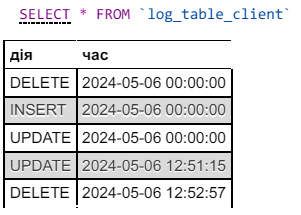


Рисунок 5 – запис дій в таблицю журналювання

ВИСНОВОК

У ході виконання лабораторної роботи було створено та використано тригери до створеної на першій лабораторній роботі бази даних.

Навчилися використовувати та створювати тригери в базах даних для автоматично реагування на події, що відбуваються в базі даних, і виконувати певні дії в залежності від цих подій, що дозволяє більш ефективно використовувати бази даних.