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;

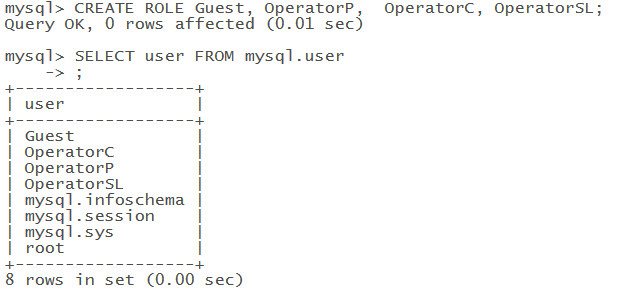


Рисунок 2 – перевірка створених ролей

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';

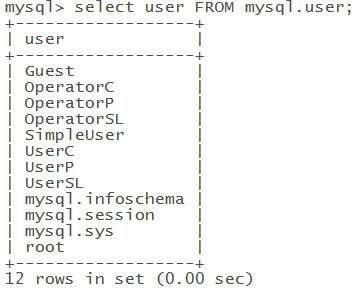


Рисунок 3 – перевірка створених користувачів

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;

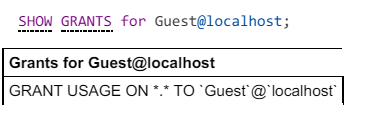


Рисунок 3 – результат перегляду привілегій Guest

Редагування привілегій ролі Guest

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

SHOW GRANTS for 'Guest'@'localhost'

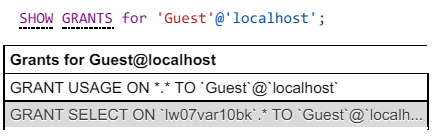


Рисунок 3 – результат редагування привілегій Guest

Шифрування записів в таблицях БД

виконаємо наступні запити до БД для роботи з зашифрованими данними

-- створення шифрованої таблиці з ключем

CREATE TABLE passTB (

pass VARBINARY(150)DEFAULT NULL

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- заносимо ключ та шифруємо запис

INSERT INTO passTB (pass)

VALUES (aes\_encrypt('P@$$w0rd', 'dji28173940'));

-- процедура перевірки паролів

DELIMITER //

CREATE PROCEDURE ComparePassword(

IN input\_password VARCHAR(255),

OUT is\_match BOOLEAN

)

BEGIN

DECLARE decrypted\_pass VARBINARY(150);

-- Спробуємо розшифрувати пароль з таблиці passtb використовуючи введений пароль

SET decrypted\_pass = AES\_DECRYPT(

(SELECT pass FROM passtb LIMIT 1),

input\_password

);

-- Якщо розшифрування успішне і пароль не порожній, тоді паролі збігаються

IF decrypted\_pass IS NOT NULL THEN

SET is\_match = TRUE;

ELSE

SET is\_match = FALSE;

END IF;

END //

DELIMITER ;

Створення шифрованих копій таблиць

Виконаємо наступний код для створення таблиці з ключем та процедуру перевірки паролю,

-- створення шифрованої таблиці з ключем

CREATE TABLE passTB (

pass VARBINARY(150)DEFAULT NULL

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- заносимо ключ та шифруємо запис

INSERT INTO passTB (pass)

VALUES (aes\_encrypt('P@$$w0rd', 'dji28173940'));

-- процедура перевірки паролів

DELIMITER //

CREATE PROCEDURE ComparePassword(

IN input\_password VARCHAR(255),

OUT is\_match BOOLEAN

)

BEGIN

DECLARE decrypted\_pass VARBINARY(150);

-- Спробуємо розшифрувати пароль з таблиці passtb використовуючи введений пароль

SET decrypted\_pass = AES\_DECRYPT(

(SELECT pass FROM passtb LIMIT 1),

input\_password

);

-- Якщо розшифрування успішне і пароль не порожній, тоді паролі збігаються

IF decrypted\_pass IS NOT NULL THEN

SET is\_match = TRUE;

ELSE

SET is\_match = FALSE;

END IF;

END //

DELIMITER ;

Створюємо структурні шифровані копії основних таблиць

-- створення шифрованих таблиць

-- Таблиця client

CREATE TABLE `client\_encrypted` (

`idAccount` int(11) NOT NULL,

`DateGiven` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`LastName` VARBINARY(255) NOT NULL,

`FirstName` VARBINARY(255) NOT NULL,

`Father` VARBINARY(255) NOT NULL,

`Percent` enum('0','1','2','3','4','5') NOT NULL,

PRIMARY KEY (`idAccount`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

-- Таблиця prices

CREATE TABLE `prices\_encrypted` (

`ДатаВстановленняВартості` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`МаркаБенизну` enum('А-76','А-92','А-95','А-98') CHARACTER SET cp1251 COLLATE cp1251\_ukrainian\_ci NOT NULL,

`ЦінаПродажу\_грн` VARBINARY(255) NOT NULL,

`КодВартості` VARBINARY(255) NOT NULL,

PRIMARY KEY (`КодВартості`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

-- Таблица sales

CREATE TABLE `sales\_encrypted` (

`НомерПродажу` int(11) NOT NULL,

`ДатаПродажу` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`НомерДисконту` VARBINARY(255) NOT NULL,

`МаркаБензину` enum('А-76','А-92','А-95','А-98') CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`КількістьЛітрів\_л` VARBINARY(255) NOT NULL,

`КодВартості` VARBINARY(255) NOT NULL,

PRIMARY KEY (`НомерПродажу`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Для автоматизації перенесення даних в зашифровані таблиці створено процедуру:

-- шифрування існуючих данних

DELIMITER //

CREATE PROCEDURE EncryptAllTablesData(

IN input\_password VARCHAR(255)

)

BEGIN

DECLARE is\_match BOOLEAN DEFAULT FALSE;

-- Вызов процедуры для проверки пароля

CALL ComparePassword(input\_password, is\_match);

-- Если пароль правильный, шифруем данные

IF is\_match THEN

-- Шифрование данных из таблицы `client`

INSERT INTO `client\_encrypted` (`idAccount`, `DateGiven`, `LastName`, `FirstName`, `Father`, `Percent`)

SELECT

`idAccount`,

`DateGiven`,

AES\_ENCRYPT(`LastName`, input\_password),

AES\_ENCRYPT(`FirstName`, input\_password),

AES\_ENCRYPT(`Father`, input\_password),

`Percent`

FROM `client`;

-- Шифрование данных из таблицы `prices`

INSERT INTO `prices\_encrypted` (`ДатаВстановленняВартості`, `МаркаБенизну`, `ЦінаПродажу\_грн`, `КодВартості`)

SELECT

`ДатаВстановленняВартості`,

`МаркаБенизну`,

AES\_ENCRYPT(`ЦінаПродажу\_грн`, input\_password),

AES\_ENCRYPT(`КодВартості`, input\_password)

FROM `prices`;

-- Шифрование данных из таблицы `sales`

INSERT INTO `sales\_encrypted` (`НомерПродажу`, `ДатаПродажу`, `НомерДисконту`, `МаркаБензину`, `КількістьЛітрів\_л`, `КодВартості`)

SELECT

`НомерПродажу`,

`ДатаПродажу`,

AES\_ENCRYPT(`НомерДисконту`, input\_password),

`МаркаБензину`,

AES\_ENCRYPT(`КількістьЛітрів\_л`, input\_password),

AES\_ENCRYPT(`КодВартості`, input\_password)

FROM `sales`;

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid password for encryption key';

END IF;

END //

DELIMITER ;

Перевіримо шифрування даних запитом до таблиць

SELECT \* FROM `sales\_encrypted` limit 2;

SELECT \* FROM `prices\_encrypted` limit 2;

SELECT \* FROM `client\_encrypted` limit 2;

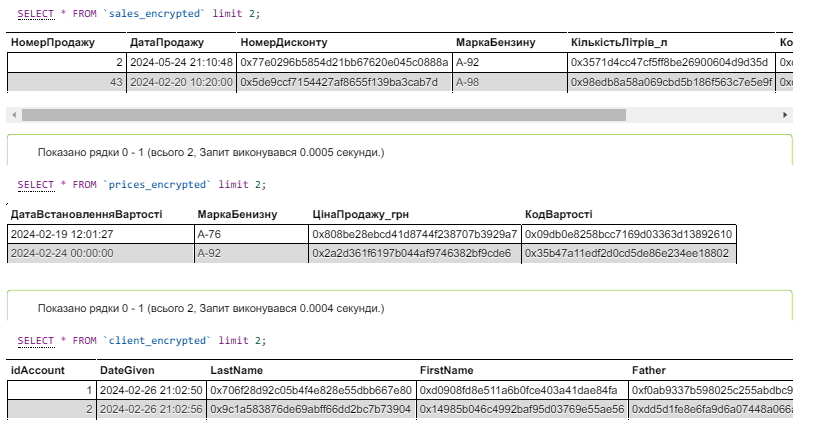


Рисунок 3 – перевірка шифрування данних

Створимо процедури для перегляду даних таблиць з розшифруванням

-- Процедура для перегляду таблиці client\_encrypted

DELIMITER //

CREATE PROCEDURE UnlockClientDataView(

IN input\_password VARCHAR(255)

)

BEGIN

DECLARE is\_match BOOLEAN DEFAULT FALSE;

-- Вызов процедуры для проверки пароля

CALL ComparePassword(input\_password, is\_match);

-- Если пароль правильный, показываем зашифрованные данные

IF is\_match THEN

SELECT

`idAccount`,

`DateGiven`,

AES\_DECRYPT(`LastName`, input\_password) AS `LastName`,

AES\_DECRYPT(`FirstName`, input\_password) AS `FirstName`,

AES\_DECRYPT(`Father`, input\_password) AS `Father`,

`Percent`

FROM

`client\_encrypted`;

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid password for encryption key';

END IF;

END //

DELIMITER ;

-- Процедура для перегляду таблиці prices\_encrypted

DELIMITER //

CREATE PROCEDURE UnlockPricesDataView(

IN input\_password VARCHAR(255)

)

BEGIN

DECLARE is\_match BOOLEAN DEFAULT FALSE;

-- Вызов процедуры для проверки пароля

CALL ComparePassword(input\_password, is\_match);

-- Если пароль правильный, показываем зашифрованные данные

IF is\_match THEN

SELECT

`ДатаВстановленняВартості`,

`МаркаБенизну`,

AES\_DECRYPT(`ЦінаПродажу\_грн`, input\_password) AS `ЦінаПродажу\_грн`,

AES\_DECRYPT(`КодВартості`, input\_password) AS `КодВартості`

FROM

`prices\_encrypted`;

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid password for encryption key';

END IF;

END //

DELIMITER ;

-- Процедура для перегляду таблиці sales\_encrypted

DELIMITER //

CREATE PROCEDURE UnlockSalesDataView(

IN input\_password VARCHAR(255)

)

BEGIN

DECLARE is\_match BOOLEAN DEFAULT FALSE;

-- Вызов процедуры для проверки пароля

CALL ComparePassword(input\_password, is\_match);

-- Если пароль правильный, показываем зашифрованные данные

IF is\_match THEN

SELECT

`НомерПродажу`,

`ДатаПродажу`,

AES\_DECRYPT(`НомерДисконту`, input\_password) AS `НомерДисконту`,

`МаркаБензину`,

AES\_DECRYPT(`КількістьЛітрів\_л`, input\_password) AS `КількістьЛітрів\_л`,

AES\_DECRYPT(`КодВартості`, input\_password) AS `КодВартості`

FROM

`sales\_encrypted`;

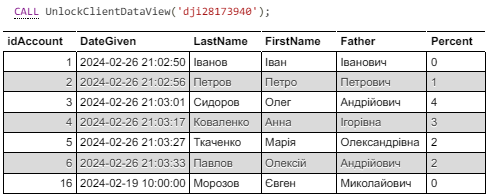
ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid password for encryption key';

END IF;

END //

DELIMITER





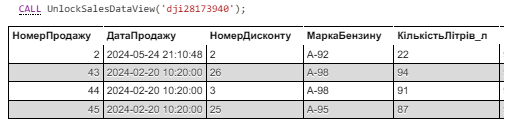


Рисунок 6 – результат перегляду частини розшифрованих даних за допомогою створених процедур.

Вставка нових значень в зашифрованому вигляді

Для цього було створено 3 процедури для відповідних таблиць

-- Процедура для вставки данних в таблицю client\_encrypted

DELIMITER //

CREATE PROCEDURE InsertIntoClientEncrypted(

IN input\_password VARCHAR(255),

IN idAccount INT,

IN LastName VARBINARY(255),

IN FirstName VARBINARY(255),

IN Father VARBINARY(255),

IN Percent ENUM('0','1','2','3','4','5')

)

BEGIN

DECLARE is\_match BOOLEAN DEFAULT FALSE;

-- Вызов процедуры для проверки пароля

CALL ComparePassword(input\_password, is\_match);

-- Если пароль правильный, вставляем данные

IF is\_match THEN

INSERT INTO `client\_encrypted` (`idAccount`, `DateGiven`, `LastName`, `FirstName`, `Father`, `Percent`)

VALUES (idAccount,

Now(),

AES\_ENCRYPT(LastName, input\_password),

AES\_ENCRYPT(FirstName, input\_password),

AES\_ENCRYPT(Father, input\_password), Percent);

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid password for encryption key';

END IF;

END //

DELIMITER ;

-- Процедура для вставки данних в таблицю prices\_encrypted

DELIMITER //

CREATE PROCEDURE InsertIntoPricesEncrypted(

IN input\_password VARCHAR(255),

IN МаркаБенизну ENUM('А-76','А-92','А-95','А-98'),

IN ЦінаПродажу\_грн DECIMAL(4,2),

IN КодВартості VARBINARY(255)

)

BEGIN

DECLARE is\_match BOOLEAN DEFAULT FALSE;

-- Вызов процедуры для проверки пароля

CALL ComparePassword(input\_password, is\_match);

-- Если пароль правильный, вставляем данные

IF is\_match THEN

INSERT INTO `prices\_encrypted` (`ДатаВстановленняВартості`, `МаркаБенизну`,

`ЦінаПродажу\_грн`, `КодВартості`)

VALUES (Now(), МаркаБенизну,

AES\_ENCRYPT(ЦінаПродажу\_грн, input\_password),

AES\_ENCRYPT(КодВартості, input\_password));

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid password for encryption key';

END IF;

END //

DELIMITER ;

-- Процедура для вставки данних в таблицю sales\_encrypted

DELIMITER //

CREATE PROCEDURE InsertIntoSalesEncrypted(

IN input\_password VARCHAR(255),

IN НомерПродажу INT,

IN НомерДисконту VARBINARY(255),

IN МаркаБензину ENUM('А-76','А-92','А-95','А-98'),

IN КількістьЛітрів\_л INT,

IN КодВартості VARBINARY(255)

)

BEGIN

DECLARE is\_match BOOLEAN DEFAULT FALSE;

-- Вызов процедуры для проверки пароля

CALL ComparePassword(input\_password, is\_match);

-- Если пароль правильный, вставляем данные

IF is\_match THEN

INSERT INTO `sales\_encrypted` (

`НомерПродажу`,

`ДатаПродажу`,

`НомерДисконту`,

`МаркаБензину`,

`КількістьЛітрів\_л`,

`КодВартості`

)

VALUES (

НомерПродажу,

NOW(),

AES\_ENCRYPT(НомерДисконту, input\_password),

МаркаБензину,

AES\_ENCRYPT(КількістьЛітрів\_л, input\_password),

AES\_ENCRYPT(КодВартості, input\_password)

);

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid password for encryption key';

END IF;

END //

DELIMITER ;

Для перевірки необхідно виконати наступні запити до БД

-- перевірка вставки зашифрованих даних

-- Вставка запису в таблицю client\_encrypted

CALL InsertIntoClientEncrypted(

'dji28173940', 10, 'Doe', 'John', 'Smith', '2'

);

-- Вставка запису в таблицю prices\_encrypted

CALL InsertIntoPricesEncrypted(

'dji28173940', 'А-95', 23.45, '12345'

);

-- Вставка запису в таблицю sales\_encrypted

CALL InsertIntoSalesEncrypted(

'dji28173940', 10, '67890', 'А-95', 50, '12345'

);

CALL GetClientData('dji28173940');

CALL GetPricesData('dji28173940');

CALL GetSalesData('dji28173940');







Рисунок 7 – результат занесення даних в шифровані таблиці

Як можемо побачити, дані правильно шифруються, вставляються і розшифровуються правильно та без помилок.

ВИСНОВОК

У ході виконання лабораторної роботи було створено копію бази даних, розробленої під час попередньої лабораторної роботи. Було визначено та створено кілька типів користувачів бази даних з різними рівнями доступу: 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).

Для кожного користувача було створено унікальне ім’я та пароль, які необхідно змінити при першому вході, забезпечуючи перший рівень захисту від несанкціонованого доступу. На другому рівні захисту даних було реалізовано шифрування та дешифрування даних, що зберігалися у таблицях в зашифрованому вигляді.

Для доступу до даних було розроблено відповідні збережені процедури та функції, що забезпечували інтерфейс користувача. Було визначено склад системних процедур та функцій для шифрування даних, створено збережені процедури для шифрування даних перед занесенням їх у таблиці нової бази даних, а також функції для відображення дешифрованих даних.

Нарешті, було перевірено коректність шифрування та дешифрування, що підтвердило правильність виконаних дій.