Unit - 4

Comparison of DAC, MAC, and Data Encryption:

1. Discretionary Access Control (DAC)

In **Discretionary Access Control**, the database owner can grant or revoke permissions to other users. This example demonstrates granting and revoking access permissions to users on a table in a database.

Query: Granting and Revoking Permissions

```
-- Granting SELECT permission on 'employees' table to user 'john_doe'

GRANT SELECT ON employees TO john_doe;


-- Revoking SELECT permission on 'employees' table from user 'john_doe'

REVOKE SELECT ON employees FROM john_doe;
```

Explanation:

- **GRANT SELECT:** The GRANT statement is used to allow a user (john_doe) the permission to perform a specific action (in this case, SELECT) on the employees table.
- **REVOKE SELECT:** The REVOKE statement removes the permission previously granted to the user on the table.
- **DAC:** This type of control allows the owner of the database object (in this case, the employees table) to grant or revoke access to other users. The decision to grant access is left to the discretion of the object owner.

2. Mandatory Access Control (MAC)

In **Mandatory Access Control**, access to resources is determined by the system based on predefined policies and classifications. This example demonstrates how a user’s access might be determined by security labels (such as sensitivity levels). In most DBMSs, MAC is implemented using security models, but for the sake of simplicity, let’s assume we are dealing with a table and we have a column indicating the classification level of data.

Query: Using a Classification Column to Implement MAC

```
-- Granting access to sensitive data based on user clearance level

SELECT first_name, last_name, email

FROM employees

WHERE clearance_level >= (SELECT clearance_level FROM users WHERE username = 'admin')

 AND data_classification <= 'Confidential';
```

Explanation:

- **clearance_level and data_classification:** The employees table is assumed to have a clearance_level column and a data_classification column. These columns control access based on the user’s security clearance and the sensitivity of the data.
- **WHERE clause:** This condition ensures that users can only access employees' data if their clearance_level is greater than or equal to that of the admin user, and the data_classification level is classified as 'Confidential' or lower.
- **MAC:** In this case, the access control system is enforcing mandatory policies. The access rights are determined by predefined security clearance levels and the classification of data, not by the discretion of the users.

3. Data Encryption

Data encryption ensures that sensitive data is securely stored and transmitted. Here, we'll demonstrate how to encrypt and decrypt sensitive data using a **column-level encryption** example in SQL (Note: actual encryption requires database-specific tools, so the syntax may vary depending on your DBMS).

Query: Encrypting and Decrypting Data

-- Inserting an encrypted credit card number into the 'transactions' table

```
INSERT INTO transactions (transaction_id, user_id, encrypted_credit_card)

VALUES (1, 101, ENCRYPT('1234-5678-9876-5432', 'encryption_key'));
```

-- Selecting and decrypting the encrypted credit card number from the 'transactions' table

```
SELECT transaction_id, user_id, DECRYPT(encrypted_credit_card, 'encryption_key') AS decrypted_credit_card

FROM transactions

WHERE user_id = 101;
```

Explanation:

- **ENCRYPT() and DECRYPT():** These are hypothetical functions that represent the encryption and decryption of sensitive data. The actual functions used to encrypt or decrypt data vary depending on the DBMS (e.g., AES_ENCRYPT and AES_DECRYPT in MySQL).
- **INSERT:** The ENCRYPT function is used to encrypt the credit card number before storing it in the transactions table.
- **SELECT:** The DECRYPT function is used to decrypt the encrypted data when retrieved. The encrypted credit card number is stored in the database, and only authorized users with the correct decryption key can access the plain text data.
- **Data Encryption:** Encryption ensures that the sensitive data is stored in a secure, unreadable format, protecting it from unauthorized access, even if the database is compromised.

Summary of Concepts:

- **DAC:** Allows the resource owner to decide who has access to the data (e.g., granting or revoking permissions).
- **MAC:** Access to resources is enforced by the system based on predefined policies (e.g., based on user security clearance and data classification).
- **Data Encryption:** Protects sensitive data by converting it into an unreadable format that can only be decrypted by authorized parties with the appropriate key.