LET’S START WITH DBMS :)

RBAC(Role-based access control)

RBAC is a security system that controls who can access a database or system based on their role in the organization.

Instead of giving permissions directly to each user, the system creates roles with specific permissions, and users are assigned to these roles.

For example, a database administrator might create roles like DB_Read for reading data and DB_Write for modifying data. Only people with the right role can read or change the information.

LET’S START WITH DBMS :)

RBAC(Role-based access control)

- **Roles:** A set of permissions associated with a job function (e.g., admin, manager, developer)
- **Permissions:** The rights granted to perform certain actions (e.g., read, write, delete).
- **Users:** Individuals who are assigned one or more roles.

Advantages:

- Makes it easier to manage permissions, especially in big systems.
- Lowers the risk of misuse by making sure users only have access to what they need for their role.
- Improves security and helps keep rules consistent across the system.

LET’S START WITH DBMS :)

Encryption

Encryption in Databases refers to the process of converting data into a secure format that can only be read or accessed by someone with the correct decryption key. This helps protect sensitive information from unauthorized access
For example, an MNC might use encryption to protect records in their database. Even if a hacker gains access to the database, they wouldn't be able to read the data without the decryption key.

Types of Encryption
1. At-Rest Encryption
2. In-Transit Encryption
3. Column-Level Encryption
4. Full Disk Encryption:

LET’S START WITH DBMS :)

Encryption

- **At-Rest Encryption:** At-rest encryption protects data stored in the database by encrypting it when it is saved to disk. This means that if someone gains access to the physical storage or the database files, they won't be able to read the data without the encryption key.

- Example: A company encrypts employee salary records stored in their database. If someone accesses the database files directly, they see encrypted data like xYz1234!@# instead of the actual salary figures.

LET’S START WITH DBMS :)

Encryption

- **In-Transit Encryption:** In-transit encryption secures data as it moves between the database and applications or users. This is typically done using secure communication protocols like SSL/TLS to prevent eavesdropping or tampering during data transmission.

- Example: When you send an email, in-transit encryption ensures that the email content is scrambled while traveling to the recipient's server, so even if intercepted, it appears as random characters like qW9rTy!@ and is unreadable.

LET’S START WITH DBMS :)

Encryption

- **Column-Level Encryption:** Column-level encryption encrypts specific columns within a database. This is used when only certain pieces of data, like credit card numbers or social security numbers, need to be protected, leaving the rest of the data unencrypted for easier access.

- Example: A database holds user information with a column for Social Security Numbers (SSNs). This column is encrypted, so an SSN like 123-45-6789 might be stored as aBcXyZ!@3 in the database.

LET’S START WITH DBMS :)

Encryption

- **Full Disk Encryption**: Full disk encryption encrypts the entire disk where the database is stored. This ensures that all data on the disk is protected, not just the database but also any files, logs, or backups stored on that disk.

- Example: A laptop used by an employee is encrypted. If the laptop is stolen, the entire disk is protected, and the thief would see only encrypted data like r9T!@2d3Xy instead of readable files.

LET’S START WITH DBMS :)

Encryption

Advantages

- It ensures that sensitive information remains secure, even if the database is breached.

- Helps organizations meet regulatory requirements for data security, such as GDPR, HIPAA

- Limits who can view or modify encrypted data to those with the appropriate decryption keys.

Challenges

- Encryption can slow down database operations because of the additional processing required.

- If decryption keys are lost, the data may become permanently inaccessible.

LET’S START WITH DBMS :)

Data masking techniques

Data masking means hiding or changing sensitive information so it’s protected from unauthorized access. This allows the data to be used for things like testing, analytics, or training without exposing the real data. The goal is to create data that looks real but isn’t actually the original information.

For example : A bank needs to train its machine learning models to detect fraudulent transactions. To protect customers' personal information, the bank uses data masking to replace real account numbers and transaction details with fake but realistic-looking data. This way, the models can be trained effectively without risking exposure of actual customer data.

LET’S START WITH DBMS :)

Data masking techniques

Common Data Masking Techniques

Substitution:

- Replacing sensitive data with random but realistic-looking values. For example, swapping real names with random names from a list.

- Example: A company's employee database contains Social Security Numbers (SSNs). To mask these, the company replaces real SSNs with randomly generated fake SSNs. So, 123-45-6789 might be substituted with 987-65-4321.

LET’S START WITH DBMS :)

Data masking techniques

Common Data Masking Techniques

Shuffling:

- Rearranging the values within a column so that the data remains realistic but is not linked to the original records.

- Example: A hospital has a list of patients with their birth dates. To protect privacy, the hospital shuffles the birth dates among the patients. For instance, Patient A's birth date of 01/15/1980 might be swapped with Patient B's birth date of 07/22/1985.

LET’S START WITH DBMS :)

Data masking techniques

Common Data Masking Techniques

Encryption:

- Encrypting data so that it is unreadable without the decryption key. This can be used as a form of masking if the data doesn't need to be human-readable.

- Example: A retail company encrypts credit card numbers in its database. A number like 4111 1111 1111 1111 might be encrypted to something like Ae34Bf98Gh65Jk21, which can’t be read without the decryption key.

LET’S START WITH DBMS :)

Data masking techniques

Common Data Masking Techniques

Nulling Out:

- Replacing sensitive data with null values or placeholders like "XXXX". This removes the original data but may reduce the usefulness of the dataset.

- Example: An HR department wants to share employee records for analysis but needs to hide salary information. They replace the salary field with null values or placeholders like N/A or XXXX.

LET’S START WITH DBMS :)

Data masking techniques

Common Data Masking Techniques

Number Variance:

- Altering numeric data by adding or subtracting a random value within a certain range. For example, changing salary figures slightly to mask the exact amounts

- Example: A finance department needs to protect the exact sales figures but still wants to share data for trend analysis. They slightly alter the figures by adding or subtracting a small random amount. For example, a sales figure of $10,000 might be changed to $10,020 or $9,980.

LET’S START WITH DBMS :)

Data masking techniques

Common Data Masking Techniques

Tokenization:

- Tokenization is a technique used to replace sensitive data with non-sensitive equivalents, called "tokens." These tokens can be stored and processed without exposing the original sensitive data, while maintaining a mapping to the original values when needed.

- Example: Imagine a database that stores customer credit card information. To protect this sensitive data, tokenization can be applied. Each original credit card number is replaced with a unique token (e.g., TKN_XYZ123). These tokens have no meaningful relationship to the actual credit card numbers.