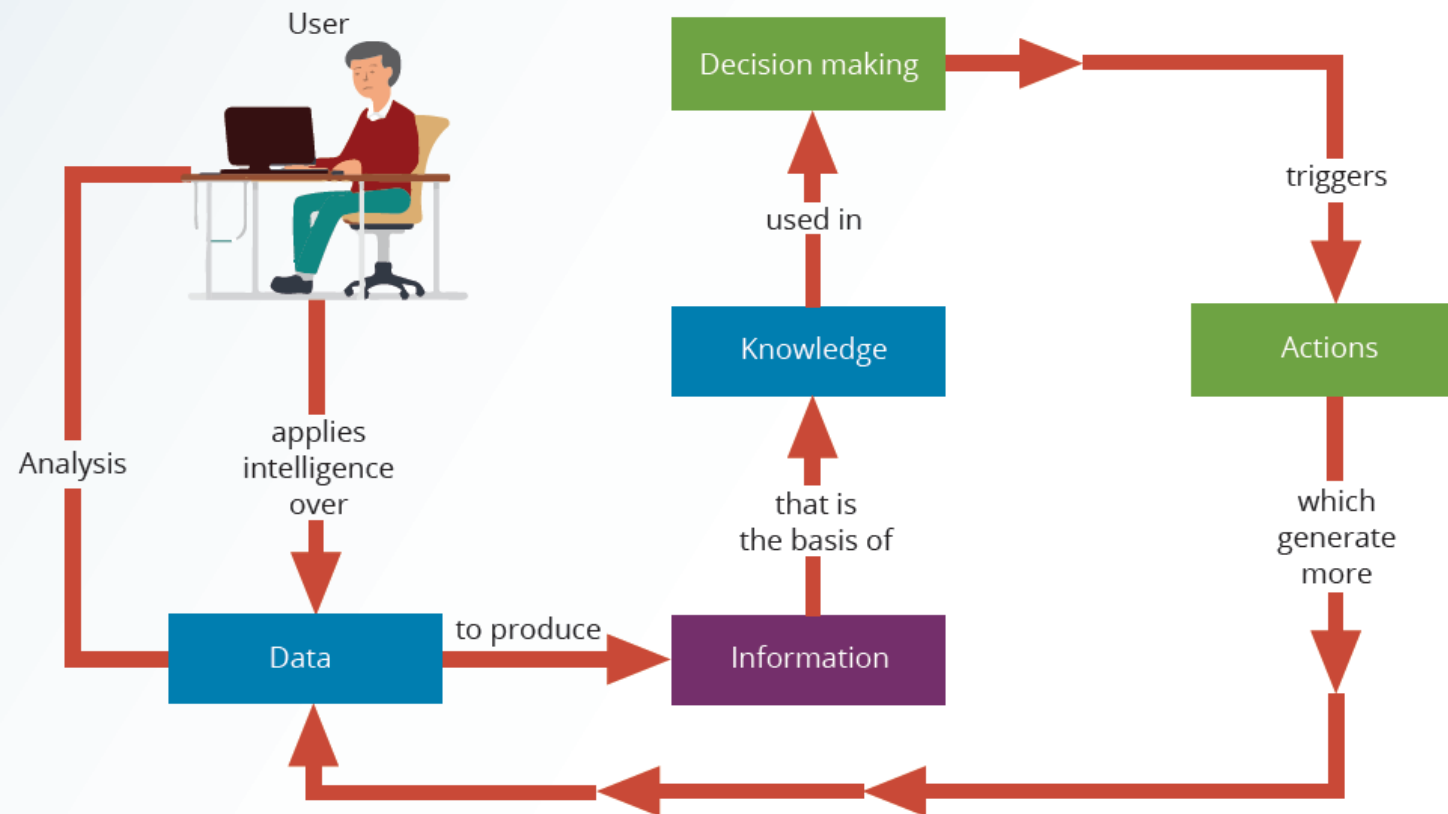


# Database Security and Administration

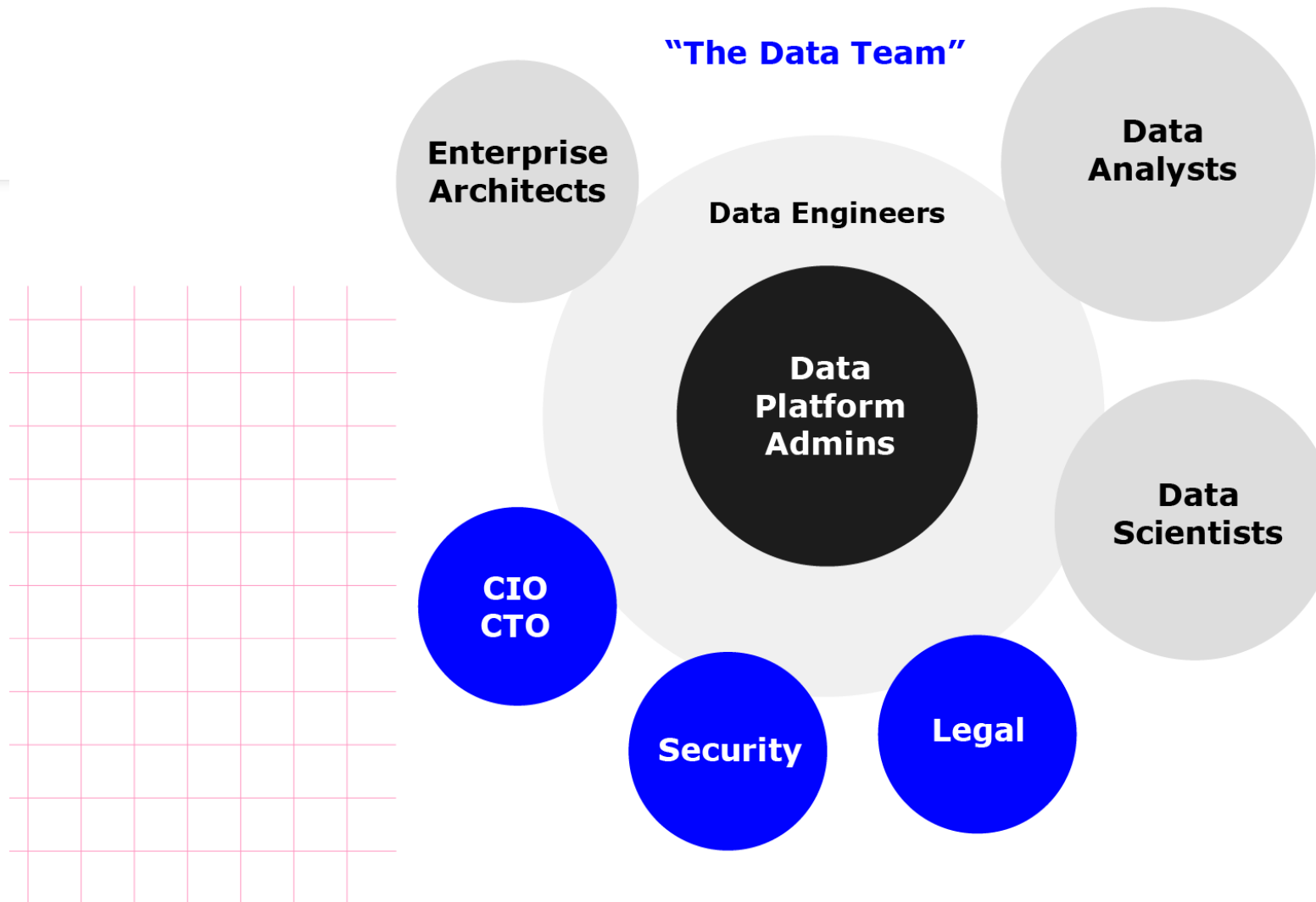
CS 341 Database Systems

# Data as a Corporate Asset

**Figure 16.1** The Data-Information-Decision-Making Cycle



## Core Personas



# Engineers Vs Analysts Vs Scientists



## Data Engineers

Build and maintain the infrastructure and systems that enable data collection, storage, and processing. They create data pipelines and ensure data is accessible, reliable, and available for use.



## Data Analysts

Analyze historical data to generate insights for business decisions. They clean, organize, and interpret data to identify trends and patterns.



## Data Scientists

Use advanced algorithms and machine learning models to predict future trends and solve complex problems. They go beyond data analysis to create predictive models and simulations.

# Database Security

- The mechanisms that protect the database against intentional or accidental threats.



# Security Vulnerabilities

- A security vulnerability is a weakness in a system component that could be exploited to allow unauthorized access or cause service disruptions.
- Such vulnerabilities could fall under one of the following categories:

## Technical

- An example would be a flaw in the operating system or web browser.

## Managerial

- For example, an organization might not educate users about critical security issues.

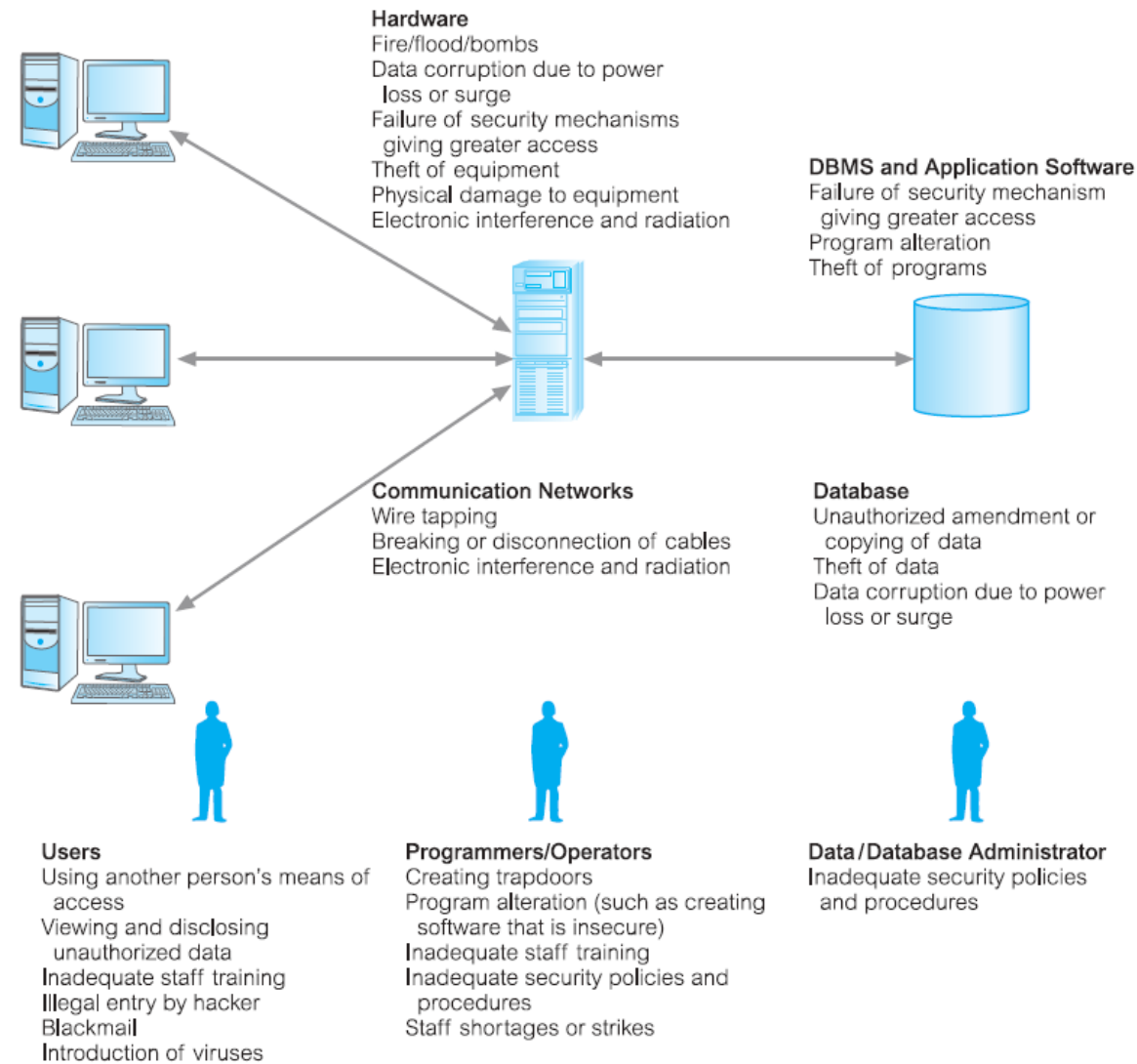
## Cultural

- Users might hide passwords under their keyboards or forget to shred confidential reports.

## Procedural

- Company procedures might not require complex passwords or the checking of user IDs.

*When a security vulnerability is left unchecked, it could become a **security threat**.*



**Figure 20.1** Summary of potential threats to computer systems.

# Authentication VS Authorization

## Authentication:

- A mechanism that determines whether a user is who he or she claims to be.

## Authorization:

- The granting of a right or privilege that enables a subject to have legitimate access to a system or a system's object.

- The responsibility to authorize use of the DBMS usually rests with the Database Administrator (DBA), who must also set up individual user accounts and passwords using the DBMS itself.

# Security in Oracle DBMS

**Privileges** - rights to execute a particular type of SQL statement or to access another user's objects.

- Some examples of Oracle privileges include the right to:
  - connect to the database (create a session);
  - create a table;
  - select rows from another user's table.
- In Oracle, there are two distinct categories of privileges:



# System Privileges

*A system privilege is the right to perform a particular action or to perform an action on any schema objects of a particular type*

- *Example, privileges to create tablespaces and to create users in a database are system privileges.*
- There are more than **eighty** distinct system privileges in Oracle.
- System privileges are granted to, or revoked from, users and roles.
- However, only users who are granted a specific system privilege with the ADMIN OPTION or users with the GRANT ANY PRIVILEGE system privilege can grant or revoke system privileges.

# Object Privileges

*An object privilege is a privilege or right to perform a particular action on a specific table, view, sequence, procedure, function, or package.*

- Different object privileges are available for different types of object.
- *For example, the privilege to delete rows from the Staff table is an object privilege.*
- A user automatically has all object privileges for schema objects contained in his or her schema.
- A user can grant any object privilege on any schema object he or she owns to any other user or role.

# Object Privileges

**TABLE 20.2** What each object privilege allows a grantee to do with tables and views.

OBJECT PRIVILEGE	TABLE	VIEW
ALTER	Change the table definition with the ALTER TABLE statement.	N/A
DELETE	Remove rows from the table with the DELETE statement. Note: SELECT privilege on the table must be granted along with the DELETE privilege.	Remove rows from the view with the DELETE statement.
INDEX	Create an index on the table with the CREATE INDEX statement.	N/A
INSERT	Add new rows to the table with the INSERT statement.	Add new rows to the view with the INSERT statement.
REFERENCES	Create a constraint that refers to the table. Cannot grant this privilege to a role.	N/A
SELECT	Query the table with the SELECT statement.	Query the view with the SELECT statement.
UPDATE	Change data in the table with the UPDATE statement. Note: SELECT privilege on the table must be granted along with the UPDATE privilege.	Change data in the view with the UPDATE statement.

# Database Roles based Security

A user can receive a privilege in two different ways:

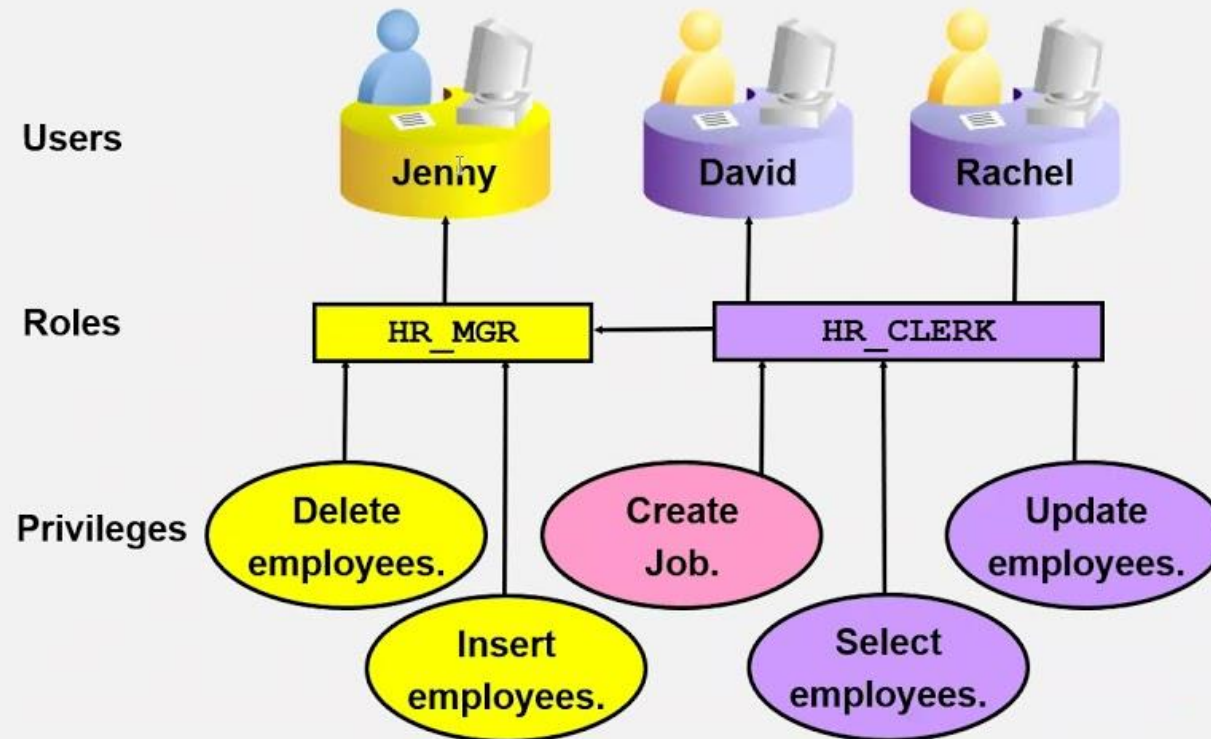
## Explicitly granted to **users**.

- For example, a user can explicitly grant the privilege to insert rows into the PropertyForRent table to the user MYUSER:
- **GRANT INSERT ON PropertyForRent TO MYUSER;**

Granted to a **role** (a named group of privileges), and then the role granted to one or more users.

- For example, a user can grant the privileges to select, insert, and update rows from the PropertyForRent table to the role named **Assistant**, which in turn can be granted to the user **MYUSER**.
- *A user can have access to several roles, and several users can be assigned the same roles.*

## Assigning Privileges to Roles and Assigning Roles to Users



# Syntax

## GRANT

PRIVILEGE, PRIVILEGE,  
PRIVILEGE

[ON OBJECT NAME]

**TO**

USER;

## REVOKE

PRIVILEGE, PRIVILEGE,  
PRIVILEGE

[ON OBJECT NAME]

**FROM**

USER;

# Examples - DCL

- GRANT **CONNECT, RESOURCE, DBA** TO **USER**;
- GRANT **UNLIMITED TABLESPACE** TO **USER**;
- GRANT **INSERT, UPDATE, DELETE** ON MYTABLE TO **USER**;
- REVOKE **DELETE** ON MYTABLE FROM **USER**;
- GRANT **ALL** ON MYTABLE TO **SUPERUSER**;
- REVOKE ALL [PRIVILEGES], GRANT OPTION FROM **USER** [, USER] ...

# System Privileges

## CONNECT

Includes the following system privileges: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW

## RESOURCE

Typically provides access to create, modify, and manage objects like tables, indexes, and views. Includes the following system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE

## DBA

All system privileges WITH ADMIN OPTION.  
Manage the entire database, including creating users, modifying storage, and more.

# Database Administrator (DBA)



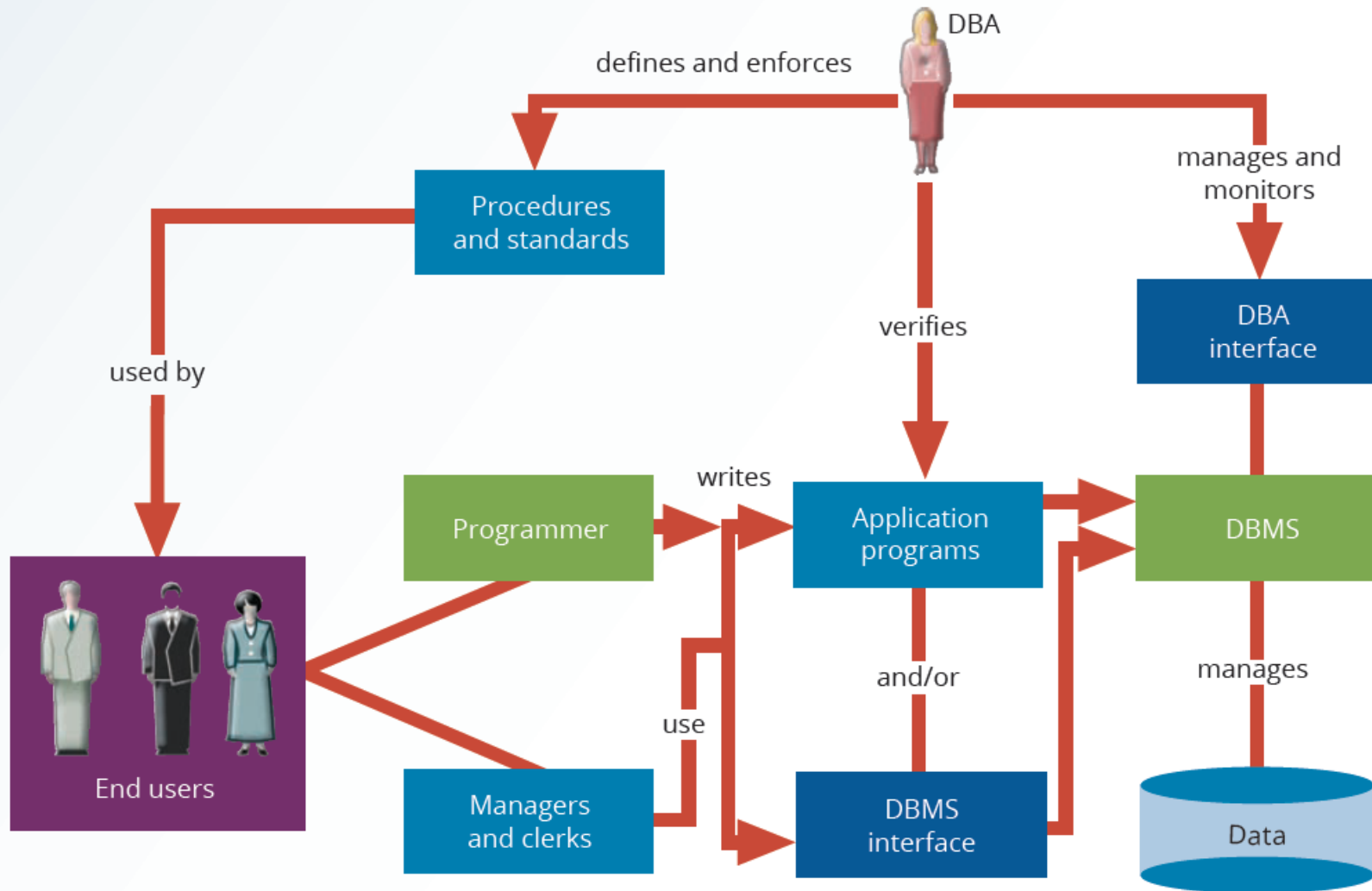
**Table 16.3 DBA Activities and Services**

DBA Activity		DBA Service
Planning		End-user support
Organizing		Policies, procedures, and standards
Testing		Data security, privacy, and integrity
Monitoring		Data backup and recovery
Delivering		Data distribution and use

**Table 16.2 Desired DBA Skills**

Managerial	Technical
Broad business understanding	Broad data-processing background and up-to-date knowledge of database technologies
Coordination skills	Understanding of Systems Development Life Cycle
Analytical skills	Structured methodologies Data flow diagrams Structure charts Programming languages
Conflict resolution skills	Knowledge of Database Life Cycle
Communication skills (oral and written)	Database modeling and design skills <ul style="list-style-type: none"> <li>• Conceptual</li> <li>• Logical</li> <li>• Physical</li> </ul>
Negotiation skills	Operational skills: Database implementation, data dictionary management, security, and so on

**Figure 16.6** A Summary of DBA Activities



# Policies Procedures and Standards

- A successful data administration strategy requires the continuous enforcement of policies, procedures, and standards for correct data creation, usage, and distribution within the database.
- The DBA must define, document, and communicate the following before they can be enforced:
- **Policies** are general statements of direction or action that communicate and support DBA goals.

# Policies Procedures and Standards

- **Standards** describe the minimum requirements of a given DBA activity; they are more detailed and specific than policies.
  - Standards are rules that evaluate the quality of the activity.
  - For example, standards define the structure of application programs and the naming conventions programmers must use.
- **Procedures** are written instructions that describe a series of steps to be followed during the performance of a given activity.
  - Procedures must be developed within existing working conditions, and they must support and enhance the work environment.

# DBA Managerial Role

- **End-user support** – providing data and information support to their departments.
- **Data security, Privacy and Integrity** – use security and integrity measures to enforce data administration and build mechanisms to safeguard against any possible threats.
- **Data Backup and Recovery** – DBA must ensure full recovery at times of failure. A new position is created named Data Security Officer (DSO) to look after disaster management.
  - Strategy for Periodic backup

**Full Backups:** Entire data set, regardless of any previous backups or circumstances.



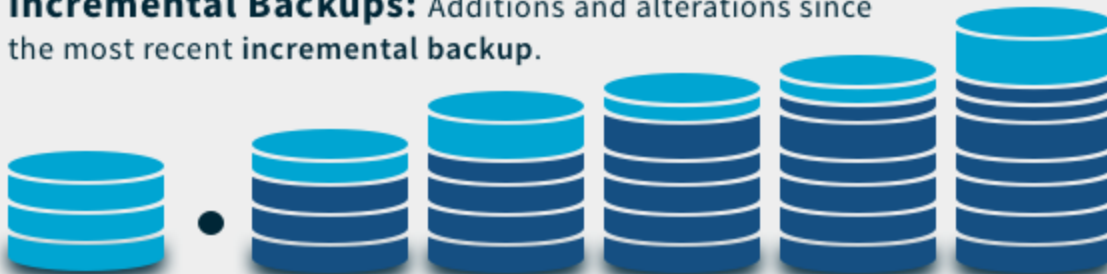
*Slowest backup time, fastest to restore*

**Differential Backups:** Additions and alterations since the most recent full backup.



*Medium backup time, Moderate restore time (Last full + latest differential)*

**Incremental Backups:** Additions and alterations since the most recent incremental backup.



*Shortest backup time, Longest restore time (Last full + all incrementals)*

Initial Full Backup • 1st Backup 2nd Backup 3rd Backup 4th Backup 5th Backup

 Data subject to backup

# DBA Technical Role

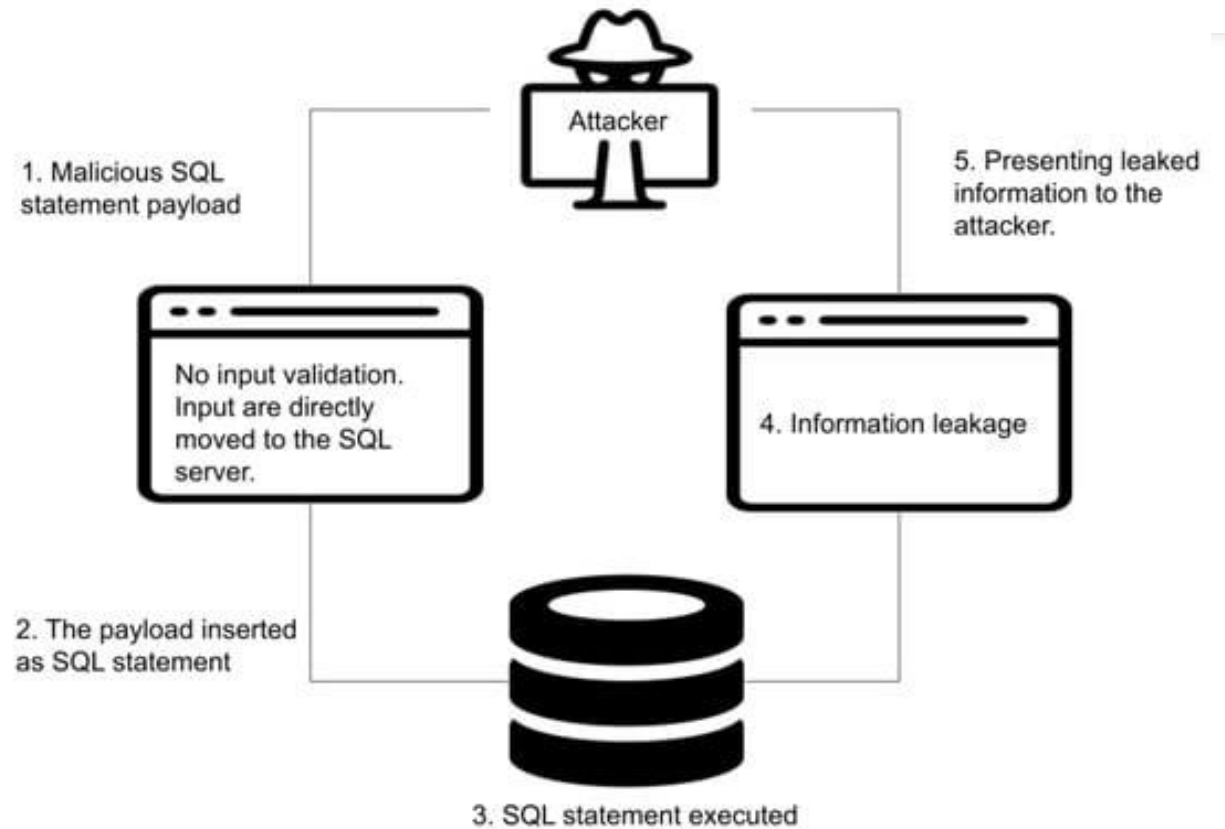
- Many of the DBA's technical activities are a logical extension of the DBA's managerial activities.
  - Evaluating, selecting, and installing the DBMS and related utilities
  - Designing and implementing databases and applications
  - Testing and evaluating databases and applications
  - Operating the DBMS, utilities, and applications
  - Training and supporting users
  - Maintaining the DBMS, utilities, and applications

# SQL Injection

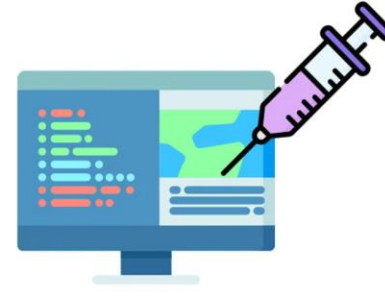


# SQL Injection Attack

*SQL Injection is a **code-based vulnerability** that allows an attacker to read and access sensitive data from the database.*

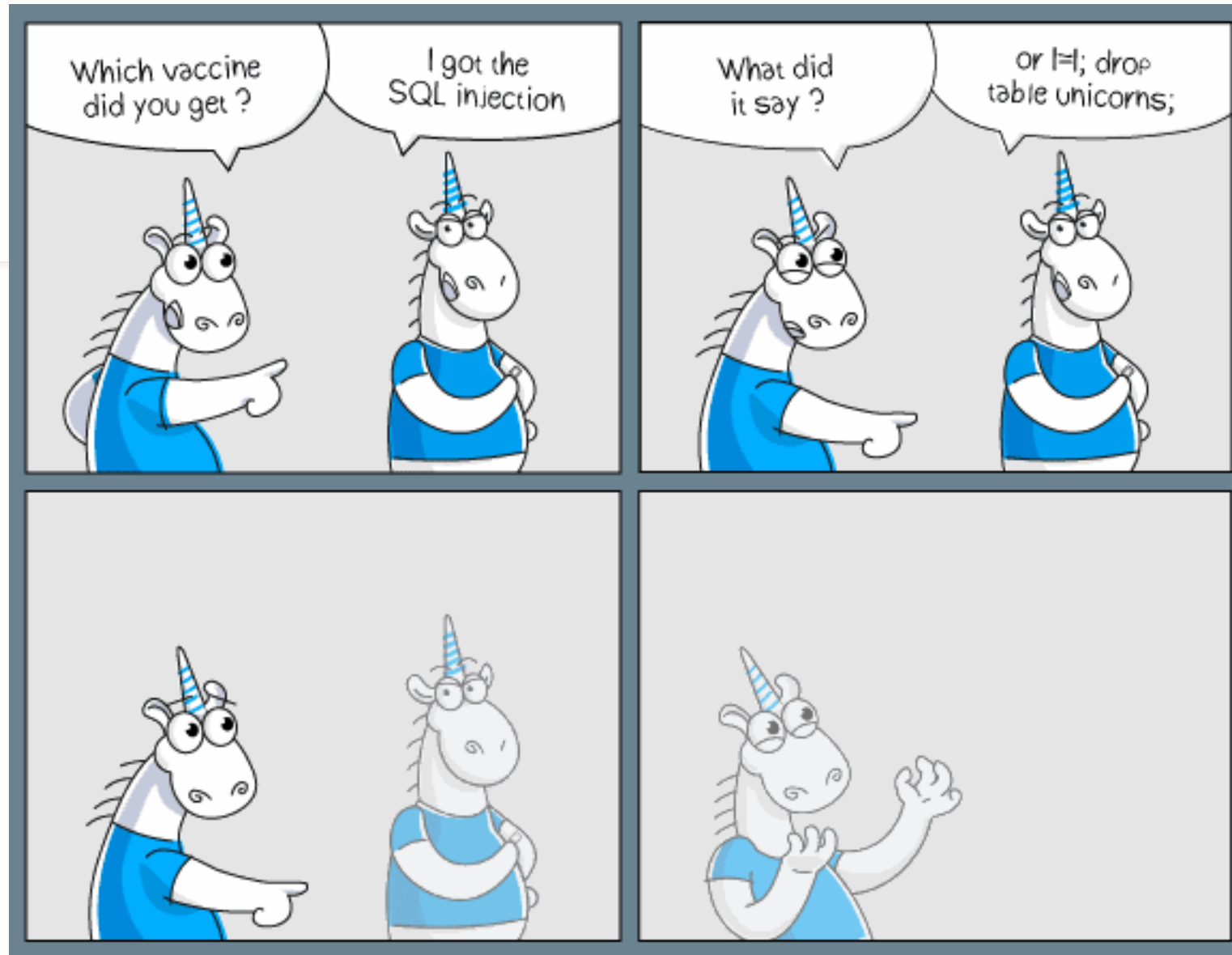


# SQL Injection Attack



- Attackers can bypass security measures of applications and use SQL queries to **modify, add, update, or delete** records in a database.
- A successful SQL injection attack can badly affect websites or web applications using relational databases

*SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a username/id, the user gives you an SQL statement that you will **unknowingly** run on your database.*



# Protection from SQL Injection

- *Input Sanitization* in the context of database security refers to the process of cleaning and validating user input to prevent harmful or malicious data, such as SQL injection attacks, from being processed by the database. Example using Prepared statements and parameterized queries
- Object relational mapping (objects in code instead of raw SQL)
- Third party authentication
- Web application Firewall
- Use of Better software
- Continuous monitoring and administration of SQL statements and database.

# Lab Module

Lab 11

# PL/SQL Packages

- Packages are schema objects that groups logically related PL/SQL types, variables, and subprograms.
- A package will have two mandatory parts –
  - *Package specification*
  - *Package body or definition*

# Why use Packages?

- Helps in making the code modular.
- Provides security by hiding the implementation details (abstraction)
- Helps in improving the functionality.
- Makes it easy to use the pre-compiled code.
- Allows the user to get quick authorization and access.

# Package Specification

```
CREATE PACKAGE package_name AS  
    PROCEDURE procedure_name;  
END package_name;  
/
```

- The package specification is the package interface which declares the types, variables, constants, exceptions, cursors and subprograms that can be referenced from outside the package.
- Note: All objects in the package specification are known as public objects.

```
CREATE PACKAGE emp_sal AS  
    PROCEDURE find_sal(e_id employees.id%type);  
END emp_sal;  
/
```

# Package body or definition:



- The package body or definition defines the queries for the cursors and the code for the subprograms.
- Note: All objects in the package body or definition are known as private objects.

```
CREATE OR REPLACE PACKAGE BODY package_name AS
    PROCEDURE procedure_name IS
        //procedure body
    END procedure_name;
END package_name;
/
```

```
CREATE OR REPLACE PACKAGE BODY emp_sal AS
    PROCEDURE find_sal(e_id employees.id%TYPE) IS
        e_sal employees.salary%TYPE;
    BEGIN
        SELECT salary INTO e_sal
        FROM employees WHERE id = e_id;
        dbms_output.put_line('Salary: ' || e_sal);
    END find_sal;
END emp_sal;
/
```

# Using Packaged Procedures

```
CREATE OR REPLACE PACKAGE BODY emp_sal AS
    PROCEDURE find_sal(e_id employees.id%TYPE) IS
        e_sal employees.salary%TYPE;
    BEGIN
        SELECT salary INTO e_sal
        FROM employees WHERE id = e_id;
        dbms_output.put_line('Salary: ' || e_sal);
    END find_sal;
END emp_sal;
/
```

```
BEGIN
    emp_sal.find_sal(1);
END;
```

# Benefits of Packages

- Allows us to overload the procedures.
- Improves the performance of the applications.
- Promotes code reusability.
- Gives us the freedom to build large applications quickly by reusing the already defined modules in the form of packages.
- Provides us with the power to declare variables, functions etc. globally thus making them accessible from anywhere in our queries.

# WITH clause

```
WITH cte_name (column1, column2, ...) AS (  
    -- Subquery or SQL statement here  
)  
SELECT *  
FROM cte_name;
```

```
WITH sid_103 AS (  
    SELECT sid FROM Reserves WHERE bid=103  
)  
SELECT * FROM sid_103;
```

```
WITH sid_103 AS (  
    SELECT sid FROM Reserves WHERE bid=103  
)  
SELECT DISTINCT sname FROM Sailors S WHERE  
sid IN (SELECT * FROM sid_103);
```

- Also known as CTE – Common Table Expression used to define a temporary table or result set that can be referenced in another query in SELECT, INSERT, UPDATE or DELETE.
- Works like Assignment operator in relational algebra allowing to break complex SQL queries into smaller and manageable parts.

*\* You can use the CTE as another table available to this specific query*

# Recall:

# Database Design

## Midterm Question



# Question

- Design the complete relational schema for Company ABC's retail database. Your schema should include:
  1. Entities with their attributes and the multiplicity of relationships between them. Also, identify all necessary foreign keys.
  2. A generalization hierarchy within the Staff entity and specify relevant constraints.
  3. Aggregation relationship, and a composition relationship between the relevant entities.
- Ensure the schema captures the full database structure clearly. If you make any assumptions, mention them in your answer.

Company ABC, a leading retail organization, is developing a comprehensive relational database to manage its operations more efficiently. The company operates stores in Karachi, Lahore, and Islamabad, each managed by a staff member in an executive role. Staff members, identified by a unique staff number, can hold various job roles such as Executive, Floor Manager, Janitorial, and Customer Service, with these roles forming a generalization hierarchy under the broader Staff entity. Staff attributes include name, contact details, address, age (over 18), salary, and valid Gmail-based email addresses (mandatory for registration) and are assigned to a specific store. Executives have a share percentage, floor managers are assigned shifts (day or night), janitorial staff have an hourly wage and staff in customer service categorize into 3 types i.e. information, complaint, refund.

ABC also maintains a detailed catalog of products, categorized into groups like Dairy, Meat, and Processed Foods, and recognized by a unique product number and price. Aggregation is applied between stores and orders, where stores record multiple orders placed by walk-in customers. Customers' names and contact details are saved for future transactions. Each order, composed of multiple products, represents a composition relationship i.e. deleting an order removes the related product list. This relational database structure, incorporating generalization, aggregation, and composition, aims to streamline data management, improve customer service, and enhance operational efficiency across ABC's stores.