DATABASE SECURITY

1/21/2025

SANDUNI AALOKA

CONTENT

TYPES OF DATABASE SECURITY ISSUES

- **Legal and Ethical issues**
  - Some information may be deemed to be private and cannot be accessed legally by unauthorized organizations or persons.(passwords)
- **Policy issues**
  - Issues regarding what kinds of information should not be made publicly available. (personal medical records)
- **System-related issues**
  - How to handle security issues in various level of system. (Physical Level , Hardware Level , DBMS Level)
- **Issues in identifying multiple security levels**
  - Identifying security levels, categorizing data and user based on these classification.

DATABASE SECURITY GOALS

- **Confidentiality**
  - Confidentiality ensures that data is accessible only to authorized individuals or entities.
- **Integrity**
  - Database integrity refers to the requirement that information be protected from improper modification.
  - Integrity is lost if unauthorized changes are made to the data by either intentional or accidental acts. (Corrupted data result in inaccuracy, fraud and erroneous decisions)
- **Availability**
  - Database availability refers to making objects available to a human user or a program who/which has a legitimate right to access those data objects. (Database DoS Attack – Running Complex Queries)

DATABASE SECURITY CONTROL MEASURES

- **Access control**
  - The security mechanism of a DBMS must include provisions for restricting access to the database system as a whole.
  - Access control, is handled by creating user accounts and passwords to control the login process by the DBMS.
- **Inference control**
  - Statistical databases are used to provide statistical information or summaries of values based on various criteria.
  - Security for statistical databases must ensure that information about individuals cannot be accessed.
  - The corresponding control measures are called inference control measures.

DATABASE SECURITY CONTROL MEASURES

- **Flow control**
  - Flow control prevents information from flowing in such a way that it reaches unauthorized users.
  - Flow control mechanisms are designed to prevent information leakage or unauthorized communication channels, such as covert channels. Covert channels refer to implicit pathways through which information flows in violation of the established security policies of an organization.
- **Data encryption**
  - Data encryption, which is used to protect sensitive data that is transmitted via some type of communications network.
  - The data is encoded using some coding algorithm.
  - An unauthorized user who accesses encoded data will have difficulty deciphering it.
  - Authorized users are given decoding or decrypting algorithms (or keys) to decipher the data.

ACCESS CONTROL MECHANISMS

- **Discretionary Access Control Mechanisms (DAC)-** These are used to grant privileges to users, including the capability to access specific data.
- **Mandatory Access Control Mechanisms (MAC)** - These are used to enforce multilevel security by classifying the data and users into various security classes (or levels) and then implementing the appropriate security policy of the organization.
- **Role-Based Access Control Mechanisms (RBAC)**: Role-Based Access Control is an access control model that assigns access permissions to users based on their roles within an organization or system. Instead of granting permissions directly to individual users, permissions are assigned to roles, and users are assigned to appropriate roles based on their job responsibilities or functions.

DISCRETIONARY ACCESS CONTROL

- The typical method to enforce discretionary access control in a database system is based on the **granting** and **revoking** of privileges.

- **Privilege** - Privilege is an authority or permission given to access the database by executing SQL statements.

- **Grant** – It gives user access privilege to the database.
  - GRANT <privilege list> ON <relation name/view name> TO <user>
- **Revoke** – It is used to cancel the granted access privileges.
  - REVOKE <privilege list> ON <relation name/view name> FROM <user>

DISCRETIONARY ACCESS CONTROL

- There are two levels for assigning privileges to use the database system such as,

▪ **The account level.**
  - At this level, the DBA specifies the particular privileges that each account (users) holds independently of the relations in the database.

▪ **The relation (or table) level.**
  - At this level, the DBA can control the privilege to access each individual relation or view in the database.

DISCRETIONARY ACCESS CONTROL

- The privileges at **account level** apply to the capabilities provided to the account itself and can include,
  - CREATE SCHEME OR CREATE TABLE to create schema or base relation.
  - CREATE VIEW privilege.
  - ALTER privilege to apply schema changes such adding or removing attributes from relations.
  - DROP privilege to delete relation or view.

DISCRETIONARY ACCESS CONTROL

- In SQL, the following types of privileges can be granted on each individual **relation** R.

  - SELECT (retrieval or read) privilege on R
  - Modification (insert, update , delete) privileges on R
    - both the INSERT and UPDATE privileges can specify that only certain attributes can be updated by the account.
- References privilege on R
  - This gives the account the capability to reference relation R when specifying integrity constraints.
  - The privilege can also be restricted to specific attributes of R.

Note : To create a view, the account must have the SELECT privilege on all relations involved in the view

DISCRETIONARY ACCESS CONTROL

- Specifying Privileges through the Use of Views.
  - The mechanism of views is an important discretionary authorization mechanism in its own right.
  - if the owner A of a relation R wants another account B to be able to retrieve only some fields of R, then A can create a view V of R that includes only those attributes and then grant SELECT on V to B.
  - The same applies to limiting B to retrieving only certain tuples of R; a view V' can be created by defining the view by means of a query that selects only those tuples from R that A wants to allow B to access.
- Revoking of Privileges.
  - The owner of a relation may want to grant the SELECT privilege to a user for a specific task and then revoke that privilege once the task is completed.
  - In SQL, a REVOKE command is included for the purpose of canceling privileges.

DISCRETIONARY ACCESS CONTROL

- Propagation of Privileges Using the GRANT OPTION.
  - Whenever the owner A of a relation R grants a privilege on R to another account B, the privilege can be given to B **with or without** the GRANT OPTION.
  - If the GRANT OPTION is given, this means that B can also grant that privilege on R to other accounts.
  - If the owner account A now revokes the privilege granted to B, all the privileges that propagated based on that privilege should automatically be revoked by the system.

MANDATORY ACCESS CONTROL

• The discretionary access control techniques of granting and revoking privileges on relations has traditionally been the main security mechanism for relational database systems.

• This is an all-or-nothing method.

• A user either has or does not have a certain privilege.

• In many applications, and additional security policy is needed that classifies data and users based on security classes.

• This approach as mandatory access control, would typically be combined with the discretionary access control mechanisms.

MANDATORY ACCESS CONTROL

- Typical security classes are top secret (TS), secret (S), confidential (C), and unclassified (U), where TS is the highest level and U the lowest,

$$TS \geq S \geq C \geq U$$

- The commonly used model for multilevel security, known as Bell-LaPadula model,
  - Classifies each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of the security classifications, T, S, C, or U.
  - Clearance of a subject S as class(S) and Classification of an object O as class(O).

ROLE BASED ACCESS CONTROL

• Its basic notion is that privileges and other permissions are associated with organizational roles rather than with individual users. • Individual users are then assigned to appropriate roles.

• Commands related to Roles

- Create Role
- Destroy Role
- Grant
- Revoke

• RBAC can be used with traditional discretionary and mandatory access control.

ROLE BASED ACCESS CONTROL

- The role hierarchy in RBAC is a natural way to organize roles to reflect the organization’s lines of authority and responsibility.
- The higher role include all access rights of lower roles.

ROLE BASED ACCESS CONTROL

Create Role

- CREATE ROLE
  - CREATE ROLE sal_mgr;
  - CREATE ROLE hr_mgr; A layer of security can be added to roles by specifying a password.
- CREATE ROLE IDENTIFIED by
  - CREATE ROLE overall_mgr IDENTIFIED by manager_password;

ROLE BASED ACCESS CONTROL

Grant Role

•Roles are associated with the users using the GRANT command,

```
CREATE USER A1 IDENTIFIED BY password; GRANT overall-mgr TO A1;
```

•The user who has created the Role is able to grant it to any other user. Otherwise, one should be given privileges as follows to grant the role to another.

```
GRANT overall_manager TO A1 WITH ADMIN OPTION; or
GRANT ANY ROLE TO A1;
```

this give the privilege to grant any role that A1 have to another user.

ROLE BASED ACCESS CONTROL

Revoke

- Revoke Role
    - REVOKE overall_mgr FROM A1;
- Revoke privilege
    - REVOKE CREATE TABLE FROM overall_mgr;
- Revoke role from another role
    - REVOKE hr_mgr FROM overall_mgr;

Remove Role

- DROP ROLE overall_mgr;

ROLE BASED ACCESS CONTROL

Set Role

• If a user is granted one or more roles, he can invoke the SET ROLE command to enable or disable roles for the current user session.

- SET ROLE test_role IDENTIFIED BY test123;
- SET ROLE NONE

EXAMPLE 1

Branch(BranchName , Code, City, Assets)

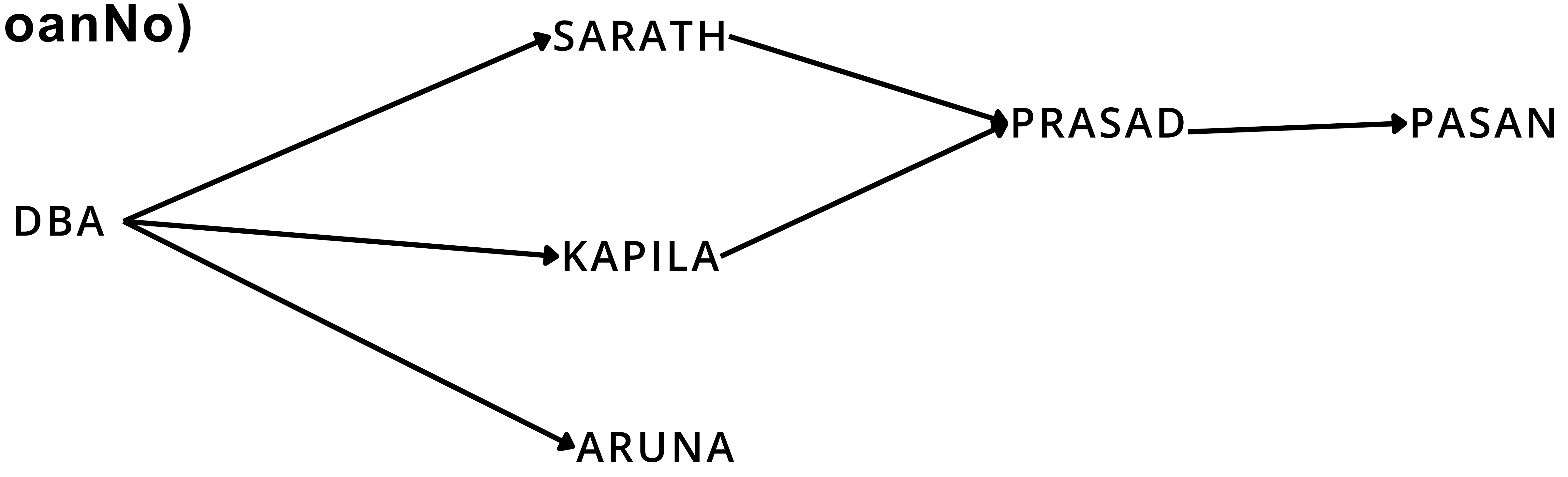
Account(AcctNo , Balance, Acc type, BranchName)

Loan(LoanNo , Amount, Loan type, BranchName)

Customer(CustomerNo , Name, Address,Phone)

C_A(CustomerNo , AcctNo)

C_L(CustomerNo , LoanNo)



EXAMPLE 1

1. User Sarath is able to retrieve the customer details of each customer at the Colombo branch along with his AcctNo and Balance and is able to update phone and address information of each customer Write SQL statements to allow this.

EXAMPLE 1

1. User Sarath is able to retrieve the customer details of each customer at the Colombo branch along with his AcctNo and Balance and is able to update phone and address information of each customer Write SQL statements to allow this.

```
CREATE VIEW Customer_account AS
SELECT C.*, A.AcctNo, Balance
FROM Customer C, Account A, C_A
WHERE C.CustomerNo = C_A.CustomerNo and
C_A.AcctNo = A.AcctNo
and A.BranchName = ‘Colombo’;

GRANT SELECT, UPDATE(Address, Phone) ON
Customer_account TO Sarath WITH GRANT OPTION;
```

EXAMPLE 1

2. Kapila is able to retrieve the name of each customer who has a loan but does not have an account at the bank.

EXAMPLE 1

2. Kapila is able to retrieve the name of each customer who has a loan but does not have an account at the bank.

```
 CREATE VIEW Loan_Only AS
 SELECT DISTINCT C.*, Amount, Type, BranchName
 FROM Customer C, Loan L, C_L
 WHERE C.CustomerNo = C_L.CustomerNo
 and C_L.LoanNo = L.LoanNo
 and NOT EXISTS (SELECT * FROM C_A
WHERE C.CutomerNo = C_A.CustomerNo);

GRANT SELECT ON Loan_Only TO Kapila WITH GRANT OPTION;
```

EXAMPLE 1

3. Prasad is able to retrieve the following,

- AcctNo, the name and the account balance of each customer who has a bank balance greater than Rs 10 000 at the Colombo branch in such account and,
- LoanNo, the name and the loan amount of each customer at the Colombo branch who only has loans but no accounts at the bank.

EXAMPLE 1

3. Prasad is able to retrieve the following,

- AcctNo, the name and the account balance of each customer who has a bank balance greater than Rs 10 000 at the Colombo branch in such account and,

```
CREATE VIEW Colombo_Account AS
SELECT AcctNo, Name, Balance
FROM Customer_account
WHERE Balance > 10000;

GRANT SELECT ON Colombo_Account TO Prasad
WITH GRANT OPTION;
```

EXAMPLE 1

3. Prasad is able to retrieve the following,

- LoanNo, the name and the loan amount of each customer at the Colombo branch who only has loans but no accounts at the bank.

```
CREATE VIEW Colombo_loan_only AS
SELECT LoanNo, Name, Amount
FROM Loan_only
WHERE BranchName = =--‘Colombo’

GRANT SELECT ON Colombo_loan_only TO Prasad WITH
GRANT OPTION;
```

EXAMPLE 1

4. To revoke the privileges of Sarath, the following command is issued.

    a. REVOKE SELECT, UPDATE (Address, ON customer_account FROM Sarath RESTRICT;

   Explain the impact of it on the users Prasad and Pasan.

EXAMPLE 1

4. To revoke the privileges of Sarath, the following command is issued.
   a. REVOKE SELECT, UPDATE (Address, ON customer_account FROM Sarath RESTRICT;

   Explain the impact of it on the users Prasad and Pasan.

There is NO impact on the privileges of Prasad and Pasan since the REVOKE command itself on Sarath will fail to prevent the abandoned privileges due to the key word RESTRICT.

EXAMPLE 2

1. DBA creates four accounts A1, A2, A3, A4. DBA wants only A1 to be able to create base relations.

EXAMPLE 2

1. DBA creates four accounts A1, A2, A3, A4. DBA wants only A1 to be able to create base relations.

GRANT CREATE TABLE TO A1; OR CREATE SCHEMA EXAMPLE AUTHORIZATION A1;

EXAMPLE 2

2. A1 creates two relation Employee and department under schema called example. A1 is the owner of two relations and has all relation privileges on each of them. A1 wants to grant to account A2 the privilege to insert and delete tuples in both of these relations. However, A1 does not want A2 to be able to propagate these privileges to additional accounts .

EXAMPLE 2

2. A1 creates two relation Employee and department under schema called example. A1 is the owner of two relations and has all relation privileges on each of them. A1 wants to grant to account A2 the privilege to insert and delete tuples in both of these relations. However, A1 does not want A2 to be able to propagate these privileges to additional accounts .

```
GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;
```

EXAMPLE 2

3. A1 wants to allow account A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts.

EXAMPLE 2

3. A1 wants to allow account A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts.

```
GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION;
```

EXAMPLE 2

4. A3 can grant the SELECT privilege on the EMPLOYEE relation to A4.

EXAMPLE 2

4. A3 can grant the SELECT privilege on the EMPLOYEE relation to A4.

```
GRANT SELECT ON EMPLOYEE TO A4;
```

EXAMPLE 2

5. A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3.

EXAMPLE 2

5. A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3

```
REVOKE SELECT ON EMPLOYEE FROM A3;
```

EXAMPLE 2

6. A1 wants to give back to A3 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A3 to be able to propagate the privilege. The limitation is to retrieve only the Name, Bdate, and Address attributes and only for the tuples with Dno = 5.

EXAMPLE 2

6. A1 wants to give back to A3 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A3 to be able to propagate the privilege. The limitation is to retrieve only the Name, Bdate, and Address attributes and only for the tuples with Dno = 5.

```
CREATE VIEW A3EMPLOYEE AS
SELECT Name, Bdate, Address
FROM EMPLOYEE
WHERE Dno = 5;

GRANT SELECT ON A3EMPLOYEE TO A3
WITH GRANT OPTION;
```

EXAMPLE 2

7. A1 wants to allow A4 to update only the Salary attribute of EMPLOYEE;

```
CREATE VIEW A3EMPLOYEE AS
SELECT Name, Bdate, Address
FROM EMPLOYEE
WHERE Dno = 5;

GRANT SELECT ON A3EMPLOYEE TO A3
WITH GRANT OPTION;
```

EXAMPLE 3(NOSQL)

In a social media platform whow would CAP theorem be applied?

EXAMPLE 3(NOSQL)

In a social media platform whow would CAP theorem be applied?

Consider a social media platform like Twitter. It needs to ensure that tweets are consistent across all users' timelines (Consistency) while also being highly available, allowing users to access their feeds at any time (Availability). However, during peak usage times, it might prioritize Availability over Consistency, leading to occasional delays or discrepancies in users' timelines.