Database Security and Authorization

Dr. Enosha Hettiarachchi

Outline

Outline (contd.)

1.1. Introduction to Database Security Issues (1)

- Types of Security
  - Legal and ethical issues
  - Policy issues
  - System-related issues
  - The need to identify multiple security levels

1.1. Introduction to Database Security Issues (2)

- Legal and ethical issues regarding the right to access certain information.
  - ***Example:*** *some information may be deemed to be private and cannot be accessed legally by unauthorized organizations or persons. In the US, there are numerous laws governing privacy of information.*

- Policy issues at the governmental, institutional, or corporate level as to what kinds of information should not be made publicly available.
  - ***Example:*** *credit ratings and personal medical records.*

1.1. Introduction to Database Security Issues (3)

- System-related issues such as the *system levels* at which various security functions should be enforced.
  - ***Example:*** *whether a security function should be handled at the physical hardware level, the operating system level, or the DBMS level.*

- The need in some organizations to identify multiple *security levels* and to categorize the data and users based on these classifications.
  - ***Example:*** *top secret, secret, confidential, and unclassified. The security policy of the organization with respect to permitting access to various classifications of data must be enforced.*

1.1. Introduction to Database Security Issues (4)

- Threats to databases
  - Loss of **integrity**
    - *Integrity is lost if unauthorized changes are made to the data by either intentional or accidental acts.*
  - Loss of **availability**
    - *Making objects available to a human user or a program to which they have a legitimate right.*
  - Loss of **confidentiality**
    - *Protection of data from unauthorized disclosure.*
    - *The impact of unauthorized disclosure of confidential information can range from violation of the Data Privacy Act to the jeopardization of national security.*

1.1. Introduction to Database Security Issues (5)

- To protect databases against these types of threats **four kinds of countermeasures** can be implemented:
  - **Access control**
  - **Inference control**
  - **Flow control**
  - **Encryption**

1.1. Introduction to Database Security Issues (6)

Access control

The security mechanism of a DBMS must include provisions for restricting access to the database as a whole

- This function is called **access control** and is handled by creating **user accounts** and **passwords** to control login process by the DBMS.

1.1. Introduction to Database Security Issues (7)

Inference control

The **security problem** associated with databases is that of controlling the access to a **statistical database**, which is used to provide statistical information or summaries of values based on various criteria.

- The countermeasures to **statistical database security** problem is called **inference control measures**.

1.1. Introduction to Database Security Issues (8)

Flow control

- Another security is that of **flow control**, which prevents information from flowing in such a way that it reaches unauthorized users.

- Channels that are pathways for information to flow implicitly in ways that violate the security policy of an organization are called **covert channels**.

1.1. Introduction to Database Security Issues (9)

Encryption

- A final security issue is **data encryption**, which is used to protect sensitive data (such as credit card numbers) that is being transmitted via some type of communication network.
- The data is **encoded** using some **encoding algorithm**.
  - An unauthorized user who access encoded data will have difficulty deciphering it, but authorized users are given decoding or decrypting algorithms (or keys) to decipher data.

1.1. Introduction to Database Security Issues (10)

- A DBMS typically includes a **database security** and **authorization subsystem** that is responsible for ensuring the security of a database against unauthorized access.

- **Two types of database security mechanisms:**
  - **Discretionary** security mechanisms
  - **Mandatory** security mechanisms

1.2 Database Security and the DBA

- The database administrator (**DBA**) is the central authority for managing a database system.
  - *The DBA's responsibilities include*
    - ◆ granting privileges to users who need to use the system
    - ◆ classifying users and data in accordance with the policy of the organization
- The DBA is responsible for the overall security of the database system.

1.2 Database Security and the DBA (2)

- The DBA has a DBA account in the DBMS
  - Sometimes these are called a **system or superuser account.**
  - These accounts provide **powerful capabilities** such as:
    1. **Account creation**
    2. **Privilege granting**
    3. **Privilege revocation**
    4. **Security level assignment**
  - Action 1 is access control, whereas 2 and 3 are discretionary mandatory and 4 is used to control mandatory authorization.

1.3 Access Protection, User Accounts, and Database Audits

- Whenever a person or group of persons need to access a database system, the individual or group must first apply for a user account.
  - The DBA will then create a new **account id** and a **password** for the user if he/she deems there is a legitimate need to access the database.

- The user must log in to the DBMS by entering account id and password whenever database access is needed.

1.3 Access Protection, User Accounts, and Database Audits(2)

- The database system must also keep **track of all operations** on the database that are applied by a certain user throughout **each login session**.
  - To keep a record of all updates applied to the database and of the particular user who applied each update, we can use the **system log**,
    - which includes an entry for each operation applied to the database that may be required for recovery from a transaction failure or system crash.

1.3 Access Protection, User Accounts, and Database Audits(3)

- If any tampering with the database is suspected, a **database audit** is performed
  - A database audit consists of *reviewing the log to examine all accesses and operations applied to the database during a certain time period.*

- A database log that is used mainly for security purposes is sometimes called an **audit trail**.

Discretionary Access Control Based on Granting and Revoking Privileges

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

2.1.Types of Discretionary Privileges

- The **account level**:
  - At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.
- The **relation level** (or **table level**):
  - At this level, the DBA can control the privilege to access each individual relation or view in the database.

2.1.Types of Discretionary Privileges(2)

- The privileges at the **account level** apply to the capabilities provided to the account itself and can include
  - the **CREATE SCHEMA** or **CREATE TABLE** privilege, to create a schema or base relation;
  - the **CREATE VIEW** privilege;
  - the **ALTER** privilege, to apply schema changes such adding or removing attributes from relations;
  - the **DROP** privilege, to delete relations or views;
  - the **MODIFY** privilege, to insert, delete, or update tuples;
  - and the **SELECT** privilege, to retrieve information from the database by using a **SELECT** query.

2.1.Types of Discretionary Privileges(3)

- The second level of privileges applies to the **relation level**
  - This includes **base relations** and virtual (**view**) relations.

- The granting and revoking of privileges generally follow an authorization model for discretionary privileges known as the **access matrix model** where
  - The **rows** of a matrix M represents **subjects** (users, accounts, programs)
  - The **columns** represent **objects** (relations, records, columns, views, operations).
  - Each position **M(i,j)** in the matrix represents the types of privileges (read, write, update) that **subject i** holds on **object j**.

2.1.Types of Discretionary Privileges(4)

- To control the granting and revoking of relation privileges, each **relation R** in a database is assigned an **owner account**, *which is typically the account that was used when the relation was created in the first place.*
  - The **owner of a relation** is given **all privileges** on that relation.
  - In SQL2, the DBA can assign an owner to a whole schema by creating the schema and associating the appropriate authorization identifier with that schema, using the **CREATE SCHEMA** command.
  - The owner account holder can **pass privileges** on any of the owned relation to other users by **granting** privileges to their accounts.

2.1.Types of Discretionary Privileges(5)

- In SQL the following **types of privileges can be granted on each individual relation R:**
  - **SELECT** (retrieval or read) privilege on R:
    - Gives the account retrieval privilege.
    - In SQL this gives the account the privilege to use the **SELECT** statement to retrieve tuples from R.
  - **MODIFY** privileges on R:
    - This gives the account the capability to modify tuples of R.
    - In SQL this privilege is further divided into **UPDATE**, **DELETE**, and **INSERT** privileges to apply the corresponding SQL command to R.
    - In addition, both the **INSERT** and **UPDATE** privileges can specify that only certain attributes can be updated by the account.

2.1.Types of Discretionary Privileges(6)

- In SQL the following types of privileges can be granted on each individual relation R (contd…):
  - **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.

- *Notice that to create a **view**, the account must have **SELECT** privilege on **all relations involved** in the view definition.*

2.2. Specifying Privileges Using Views

- The mechanism of **views** is an important discretionary authorization mechanism in its own right.
- For example,
  - 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.

2.3. Revoking Privileges

- In some cases it is desirable to grant a privilege to a user temporarily.
- For example,
  - 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.
  - Hence, a mechanism for **revoking** privileges is needed. In SQL, a **REVOKE** command is included for the purpose of **canceling privileges**.

2.4. Propagation of Privileges using the GRANT OPTION

- Whenever the owner A of a relation R grants a privilege on R to another account B, 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.
  - Suppose that B is given the **GRANT OPTION** by A and that B then grants the privilege on R to a third account C, also with **GRANT OPTION**. In this way, privileges on R can **propagate** to other accounts without the knowledge of the owner of R.

  - If the owner account A now revokes the privilege granted to B, all the privileges that B propagated based on that privilege should automatically be revoked by the system.

2.5. An Example to Illustrate Granting and Revoking of Privileges - Privileges

- Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

1. **System privileges** - This allows the user to CREATE, ALTER, or DROP database objects.

2. **Object privileges** - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

2.5. An Example to Illustrate Granting and Revoking of Privileges - Privileges

- **System privileges** - This allows the user to CREATE, ALTER, or DROP database objects.

System Privileges	Description
CREATE object	allows users to create the specified object in their own schema.
CREATE ANY object	allows users to create the specified object in any schema.

  - **The above rules also apply for ALTER and DROP system privileges.**

2.5. An Example to Illustrate Granting and Revoking of Privileges - Privileges

- **Object privileges** - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Object Privileges	Description
INSERT	allows users to insert rows into a table.
SELECT	allows users to select data from a database object.
UPDATE	allows user to update data in a table.
EXECUTE	allows user to execute a stored procedure or a function.

2.5. An Example to Illustrate Granting and Revoking of Privileges

- Suppose that the DBA creates four accounts
  - **A1, A2, A3, A4**
- and wants only A1 to be able to create new database tables (base relations). Then the DBA must issue the following GRANT command in SQL

```
GRANT CREATETAB TO A1;
```

- In SQL2 the same effect can be accomplished by having the DBA issue a **CREATE SCHEMA** command as follows:

```
CREATE SCHAMA EXAMPLE AUTHORIZATION A1;
```

  - User account A1 can create tables under the schema called **EXAMPLE**.

2.5. An Example to Illustrate Granting and Revoking of Privileges(2)

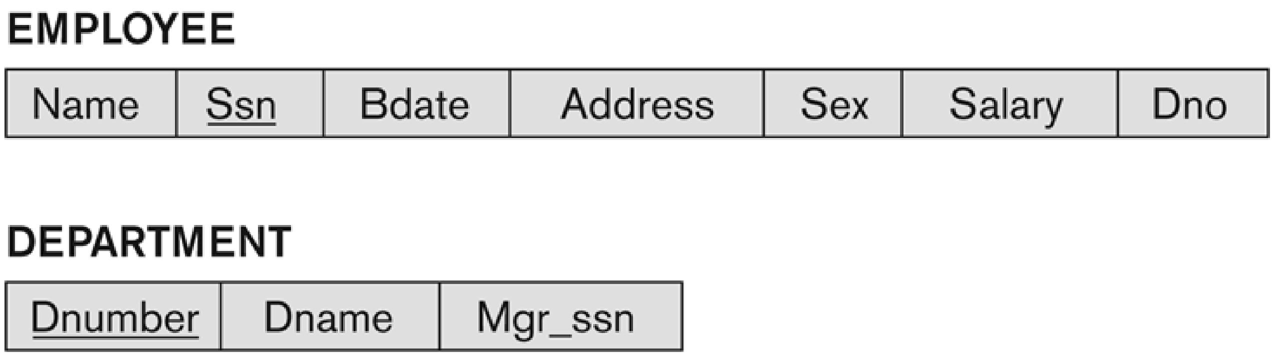
- Suppose that A1 **creates** the two base relations **EMPLOYEE** and **DEPARTMENT**
  - A1 is then **owner** of these two relations and hence all the relation privileges on each of them.

- Suppose that A1 wants to grant A2 the privilege to **insert and delete tuples** in both of these relations, but A1 does not want A2 to be able to propagate these privileges to additional accounts:

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

Note: owner account A1 has the GRANT OPTION, allowing it to grant privileges on the relation to other accounts. However, account A2 cannot grant INSERT and DELETE privileges on the EMPLOYEE and DEPARTMENT tables because A2 was not given the GRANT OPTION in the preceding command.

2.5. An Example to Illustrate Granting and Revoking of Privileges(3)



EMPLOYEE

Name	Ssn	Bdate	Address	Sex	Salary	Dno

DEPARTMENT

Dnumber	Dname	Mgr_ssn

2.5. An Example to Illustrate Granting and Revoking of Privileges(4)

- Suppose that **A1 wants to allow A3 to retrieve** information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts.
- A1 can issue the command:

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

- A3 can grant the **SELECT** privilege on the **EMPLOYEE** relation to A4 by issuing:

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

  - Notice that A4 can't propagate the SELECT privilege because GRANT OPTION was not given to A4

2.5. An Example to Illustrate Granting and Revoking of Privileges(5)

- Suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 can issue:

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

- The DBMS must now automatically revoke the SELECT privilege on EMPLOYEE from A4, too, because A3 granted that privilege to A4 and A3 does not have the privilege any more.

2.5. An Example to Illustrate Granting and Revoking of Privileges(6)

- Suppose that 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.

- **A1 then create the view:**

```
CREATE VIEW A3EMPLOYEE AS
   SELECT NAME, BDATE, ADDRESS
   FROM EMPLOYEE
   WHERE DNO = 5;
```

- After the view is created, **A1 can grant SELECT** on the view A3EMPLOYEE to A3 as follows:

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

2.5. An Example to Illustrate Granting and Revoking of Privileges(7)

- Finally, suppose that A1 wants to allow A4 to update only the SALARY attribute of EMPLOYEE;
- A1 can issue:

```
GRANT UPDATE ON EMPLOYEE (SALARY) TO
A4;
```

  - The **UPDATE** or **INSERT** privilege can specify particular attributes that may be updated or inserted in a relation.
  - Other privileges (**SELECT**, **DELETE**) are not attribute specific.

2.6. Specifying Limits on Propagation of Privileges

- Techniques to limit the propagation of privileges have been developed, although they have not yet been implemented in most DBMSs and are not a part of SQL.

  – Limiting **horizontal propagation** to an integer number i means that an account B given the GRANT OPTION can grant the privilege to at most i other accounts.

  – **Vertical propagation** is more complicated; it limits the depth of the granting of privileges.

  – *This limits the sequence of GRANT OPTIONS that can be given from one account to the next, based on a single original grant of the privilege.*

3. Mandatory Access Control and Role-Based Access Control for Multilevel Security

- 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, known as **Mandatory Access Control (MAC)**, would typically be **combined** with the **discretionary access control mechanisms.**
- Most commercial DBMSs currently provide mechanisms only for discretionary access control.

3.1 Comparing Discretionary Access Control and Mandatory Access Control

- **Discretionary Access Control (DAC)** policies are characterized by a high degree of flexibility, which makes them suitable for a large variety of application domains.

- The main drawback of **DAC** models is their vulnerability to malicious attacks, such as Trojan horses embedded in application programs.

3.1 Comparing Discretionary Access Control and Mandatory Access Control(2)

- By contrast, mandatory policies **ensure a high degree of protection** in a way, they prevent any illegal flow of information.
- **Mandatory policies** have the **drawback** of being **too rigid and they are only applicable in limited environments.**
- In many practical situations, **discretionary policies are preferred** because they **offer a better trade-off between security and applicability.**

3.2 Role-Based Access Control

- **Role-based access control (RBAC)** emerged rapidly in the 1990s as a proven technology for managing and enforcing security in large-scale enterprisewide systems.
- Its basic notion is that **permissions are associated with roles**, and users are assigned to appropriate roles.
- Roles can be created using the **CREATE ROLE** and **DESTROY ROLE** commands.
  - The **GRANT** and **REVOKE** commands discussed under DAC can then be used to assign and revoke privileges from roles.

3.2 Role-Based Access Control(2)

- **RBAC** appears to be a viable alternative to traditional discretionary and mandatory access controls; it ensures that only authorized users are given access to certain data or resources.
- Many DBMSs have allowed the concept of roles, where privileges can be assigned to roles.
- Role hierarchy in **RBAC** is a natural way of organizing roles to reflect the organization’s lines of authority and responsibility.

3.2 Role-Based Access Control(3)

- Another important consideration in **RBAC** systems is the **possible temporal constraints that may exist on roles**, such as time and duration of role activations, and timed triggering of a role by an activation of another role.

- Using an **RBAC** model is **highly desirable goal for addressing the key security requirements of Web-based applications.**

- In contrast, discretionary access control (**DAC**) and mandatory access control (**MAC**) models **lack capabilities needed to support the security requirements in emerging enterprises and Web-based applications.**

3.2 Role-Based Access Control(4)

- **Roles:** Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users, thereby automatically granting or revoking privileges.

3.2 Role-Based Access Control(5)

- **Creating Roles:**

The Syntax to create a role is:

```
CREATE ROLE role_name
[IDENTIFIED BY password];
```

3.2 Role-Based Access Control(6)

Creating Roles:

Create a role that can log in, but don't have a password

```
CREATE ROLE Nimal LOGIN;
```

Create a role with a password

```
CREATE ROLE Kamal WITH PASSWORD ‘12345';
```

Create a role with a password that is valid until the end of 2016

```
CREATE ROLE Kamal WITH LOGIN PASSWORD ‘12345'
VALID UNTIL '2017-01-01';
```

3.2 Role-Based Access Control(7)

Alter Role:

To alter the authorization method for a role, you must have the ALTER ANY ROLE system privilege.

```
ALTER ROLE Kamal WITH PASSWORD ‘45678';
```

3.2 Role-Based Access Control(8)

Drop Role:

To drop a role, you must have the DROP ANY ROLE system privilege.

The following statement drops the role HR_Executive:

```
DROP ROLE HR_Executive;
```

3.2 Role-Based Access Control(9)

Granting Privileges to Roles:

```
GRANT <privilege_name> TO <role_name>;
```

```
GRANT SELECT ON empInfo TO clerk;

GRANT INSERT, UPDATE ON empInfo TO
HR_Executive;
```

3.2 Role-Based Access Control(10)

Assigning Roles to Users:

```
GRANT <roles_name> TO <user_name>;
```

```
GRANT clerk TO user1;

GRANT HR_Executive TO user2;
```

3.2 Role-Based Access Control(11)

- It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user.
- If a role is identified by a password, then, when you GRANT or REVOKE privileges to the role, you definitely have to identify it with the password.

3.2 Role-Based Access Control(12)

- **Example:** To grant CREATE TABLE privilege to a user by creating a testing role:

1. First, create a testing Role

```
CREATE ROLE testing
```

2. Second, grant a CREATE TABLE privilege to the ROLE testing. You can add more privileges to the ROLE.

```
GRANT CREATE TABLE TO testing;
```

3.2 Role-Based Access Control(13)

3. Third, grant the role to a user.

```
GRANT testing TO user1;
```

- To revoke a CREATE TABLE privilege from testing ROLE, you can write:

```
REVOKE CREATE TABLE FROM testing;
```

4 Introduction to Statistical Database Security

- **Statistical databases** are used mainly to produce statistics on various populations.
- The database may contain **confidential data** on individuals, which should be protected from user access.
- Users are permitted to retrieve **statistical information** on the populations, such as **averages, sums, counts, maximums, minimums,** and **standard deviations**.

4 Introduction to Statistical Database Security(2)

- A **population** is a set of tuples of a relation (table) that satisfy some selection condition.
- Statistical queries involve applying **statistical functions** to a **population** of tuples.

4 Introduction to Statistical Database Security(3)

- For example, we may want to retrieve the *number* of individuals in a **population** or the *average income* in the population.
  - However, statistical users are not allowed to retrieve individual data, such as the income of a specific person.
- Statistical database security techniques must prohibit the retrieval of individual data.
- This can be achieved by prohibiting queries that retrieve attribute values and by allowing only queries that involve statistical aggregate functions such as COUNT, SUM, MIN, MAX, AVERAGE, and STANDARD DEVIATION.
  - Such queries are sometimes called **statistical queries**.

4 Introduction to Statistical Database Security(4)

- It is DBMS’s responsibility to ensure confidentiality of information about individuals, while still providing useful statistical summaries of data about those individuals to users. Provision of **privacy protection** of users in a statistical database is paramount.

- In some cases it is possible to **infer** the values of individual tuples from a sequence statistical queries.
  - This is particularly true when the conditions result in a population consisting of a small number of tuples.

5 Introduction to Flow Control

- **Flow control** regulates the distribution or flow of information among accessible objects.
- A **flow** between object X and object Y occurs when a program reads values from X and writes values into Y.
  - Flow controls check that information contained in some objects does not flow explicitly or implicitly into less protected objects.

- A **flow policy** specifies the channels along which information is allowed to move.
  - The simplest flow policy specifies just two classes of information:
    - confidential (C) and nonconfidential (N)
  - and allows all flows except those from class C to class N.

5.1 Covert Channels

- A **covert channel** allows a transfer of information that violates the security or the policy.
- A **covert channel allows** information to pass from a higher classification level to a lower classification level through **improper means**.

5.1 Covert Channels(2)

- **Covert channels** can be classified into two broad categories:
  - **Storage channels** do not require any temporal synchronization, in that information is conveyed by accessing system information or what is otherwise inaccessible to the user.
  - **Timing channel** allow the information to be conveyed by the timing of events or processes.
- Some security experts believe that one way to avoid covert channels is for programmers to not actually gain access to sensitive data that a program is supposed to process after the program has been put into operation.

6 Encryption and Public Key Infrastructures

- **Encryption** is a means of maintaining secure data in an insecure environment.
- **Encryption** consists of applying an **encryption algorithm** to data using some prespecified **encryption key**.
- The resulting data has to be **decrypted** using a **decryption key** to recover the original data.

Summary

- 1 Database Security and Authorization
- 2 Discretionary Access Control
- 3 Mandatory Access Control and Role-Based Access Control for Multilevel Security
- 4 Statistical Database Security
- 5 Flow Control
- 6 Encryption and Public Key Infrastructures

-END-