Database Security

IT5306 Principles of Information Security

Level III - Semester 5

List of sub topics

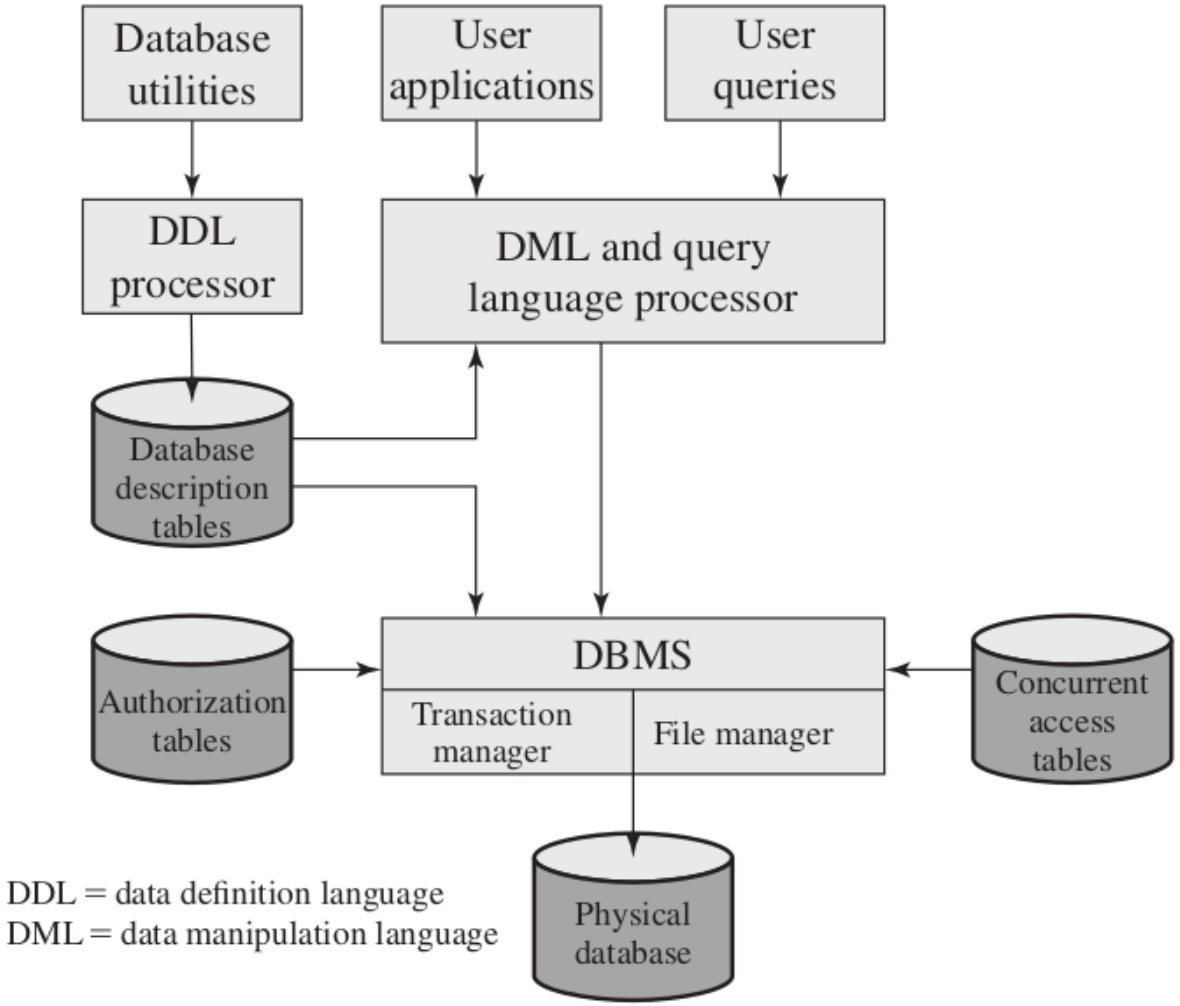
1.1 The Need for Database Security
1.2 SQL Injection Attacks
1.3 Database Access Control
1.4 Inference
1.5 Database Encryption

1.1 The Need for Database Security

- A database is a structured collection of data stored for use by one or more applications.
- In addition to data, a database contains the relationships between data items and groups of data items.
- A database management system (DBMS) is a suite of programs for constructing and maintaining the database and for offering ad hoc query facilities to multiple users and applications.
- A query language provides a uniform interface to the database for users and applications.

1.1 The Need for Database Security (cont.)

- Architecture of a DBMS:



1.1 The Need for Database Security (cont.)

- Organizational databases tend to concentrate sensitive information in a single logical system. E.g.,
  - Corporate financial data
  - Confidential phone records
  - Customer and employee information, such as name, Social Security number, bank account information, and credit card information
  - Proprietary product information
  - Health care information and medical records

- It is important to be able to store and serve such information while facing internal and external threats.
- Therefore, security specifically tailored to databases is an increasingly important component of an overall organizational security strategy.

1.1 The Need for Database Security (cont.)

- Database security has not kept pace with the increased reliance on databases due to multiple reasons:
  - There is a dramatic imbalance between the complexity of modern database management systems (DBMS) and the security techniques used to protect these critical systems.
  - Databases uses Structured Query Language (SQL), which is very complex and therefore, requires a full understanding of it to avoid its vulnerabilities.
  - The typical organization lacks full-time database security personnel. The result is a mismatch between requirements and capabilities.
  - Most enterprise environments consist of a heterogeneous mixture of database platforms. This creates an additional complexity hurdle for security personnel.

1.1 The Need for Database Security (cont.)

- Operating system security mechanisms typically control read and write access to entire files.
- So they could be used to allow a user to read or to write any information in, for example, a personnel file.
- But they could not be used to limit access to specific records or fields in that file.
- A DBMS typically does allow this type of more detailed access control to be specified.

1.2 SQL Injection Attacks

- The SQL injection (SQLi) attack is one of the most prevalent and dangerous network-based security threats.
- SQLi is an attack that exploits a security vulnerability occurring in the database layer of an application (such as queries).
- Using SQL injection, the attacker can extract or manipulate the web application's data.
- The attack is viable when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed, and thereby unexpectedly executed.

1.2 SQL Injection Attacks (cont.)

The following is steps taken by a typical SQLi attack on a website:

1. Hacker finds a vulnerability in a custom Web application and injects an SQL command.
2. The Web server receives the malicious code and sends it to the Web application server.
3. The Web application server receives the malicious code from the Web server and sends it to the database server.
4. The database server executes the malicious code on the database. The database returns data from credit cards table.
5. The Web application server dynamically generates a page with data including credit card details from the database.
6. The Web server sends the credit card details to the hacker.

1.2 SQL Injection Attacks (cont.)

- The SQLi attack typically works by prematurely terminating a text string and appending a new command.

- Consider the following script that build an SQL query by combining predefined strings with text entered by a user:

```
var Shipcity;
ShipCity = Request.form ("ShipCity");
var sql = "select * from OrdersTable where ShipCity = ' " +
ShipCity + " ' ";
```

- The intention of the script's designer is that a user will enter the name of a city, e.g., Redmond, then the following SQL query is generated:

```
SELECT * FROM OrdersTable WHERE ShipCity =
'Redmond'
```

1.2 SQL Injection Attacks (cont.)

- Suppose, however, the user enters the following:

  Boston'; DROP table OrdersTable--

- This results in the following SQL query:

  SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond'; DROP table OrdersTable--

- When the SQL server processes this statement, it will first select all records in OrdersTable where ShipCity is Redmond.

- Then, it executes the DROP request, which deletes the table.

1.2 SQL Injection Attacks (cont.)

SQLi Countermeasures

- Many SQLi attacks succeed because developers have used insecure coding practices.
- Thus, defensive coding is an effective way to dramatically reduce the threat from SQLi. This can be done by:
  - Manual defensive coding practices
  - Parameterized query insertion
  - SQL DOM

1.2 SQL Injection Attacks (cont.)

- **Manual defensive coding practices:** E.g., check that inputs that are supposed to be numeric contain no characters other than digits. Perform pattern matching to try to distinguish normal input from abnormal input.
- **Parameterized query insertion:** allowing the application developer to more accurately specify the structure of an SQL query, and pass the value parameters to it separately such that any unsanitary user input is not allowed to modify the query structure.
- **SQL DOM:** SQL DOM is a set of classes that enables automated data type validation and escaping. This approach uses encapsulation of database queries to provide a safe and reliable way to access databases.

1.2 SQL Injection Attacks (cont.)

SQLi Detection Methods

- **Signature based:** This technique attempts to match specific attack patterns. Such an approach must be constantly updated and may not work against self-modifying attacks.

- **Anomaly based:** This approach attempts to define normal behavior and then detect behavior patterns outside the normal range. A number of approaches have been used. In general terms, there is a training phase, in which the system learns the range of normal behavior, followed by the actual detection phase.

- **Code analysis:** Code analysis techniques involve the use of a test suite to detect SQLi vulnerabilities. The test suite is designed to generate a wide range of SQLi attacks and assess the response of the system.

1.3 Database Access Control

- Commercial and open-source DBMSs typically provide an access control capability for the database.

- Typically, a DBMS can support a range of administrative policies, including the following:
  - **Centralized administration:** A small number of privileged users may grant and revoke access rights.

  - **Ownership-based administration:** The owner (creator) of a table may grant and revoke access rights to the table.

  - **Decentralized administration:** In addition to granting and revoking access rights to a table, the owner of the table may grant and revoke authorization rights to other users, allowing them to grant and revoke access rights to the table.

1.3 Database Access Control (cont.)

- SQL provides two commands for managing access rights: GRANT and REVOKE.
- For different versions of SQL, their syntax can be slightly different.
- GRANT command can be used to grant one or more access rights or can be used to assign a user to a role. For access rights, the command can optionally specify that it applies only to a specified table.
- Consider the following statement:

GRANT SELECT ON ANY TABLE TO ricflair

- This statement enables user ricflair to query any table in the database.

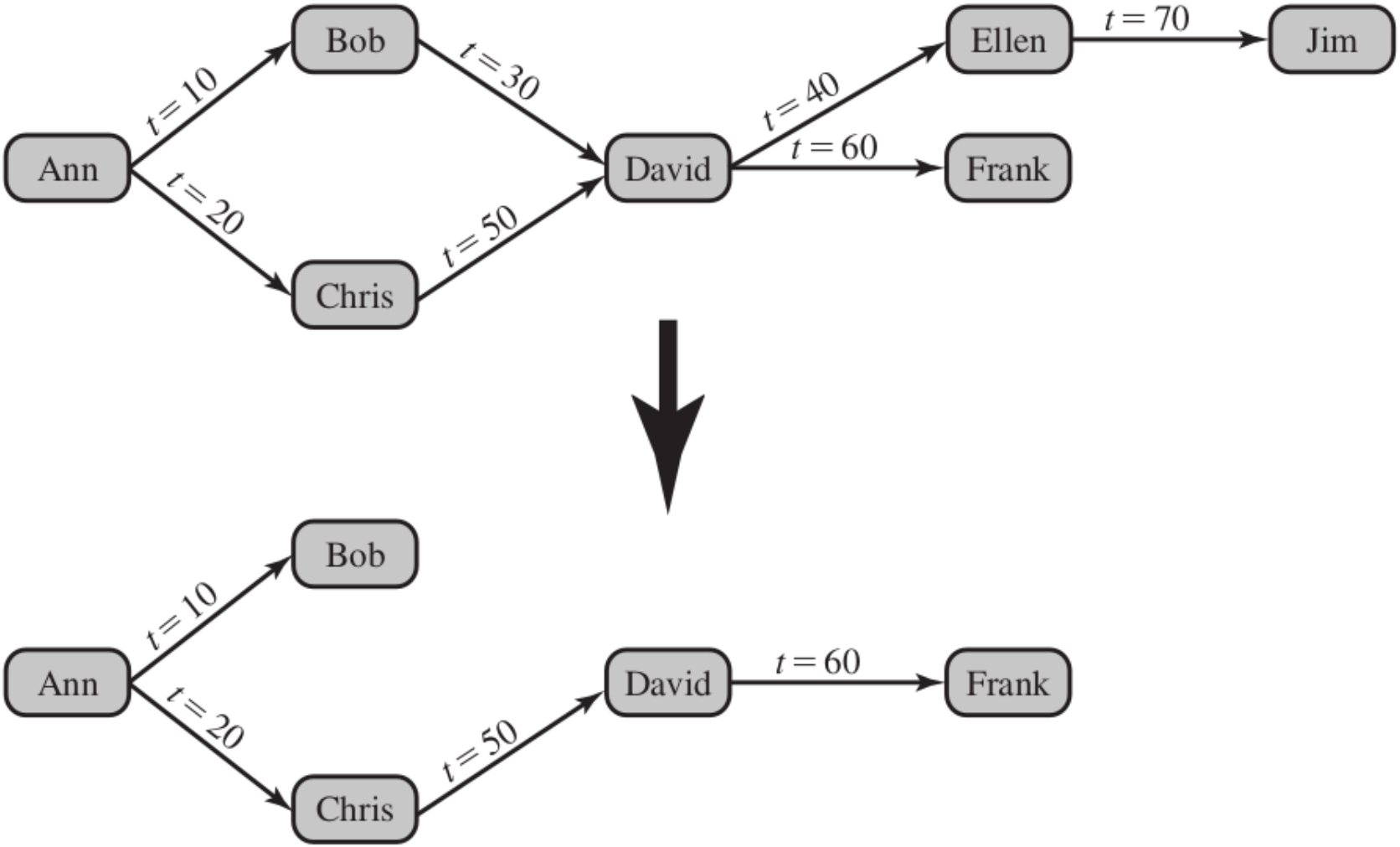
1.3 Database Access Control (cont.)

- The REVOKE command facilitates removing any already granted access rights from a user.
- Thus, the following statement revokes the access rights of the preceding example:

REVOKE SELECT ON ANY TABLE FROM ricflair

- Cascading Authorizations: The grant option enables an access right to cascade through a number of users.
- In the figure shown in next slide, Ann grants the access right to Bob, who intern grants access to more users down the line.
- Just as the granting of privileges cascades from one user to another using the grant option, the revocation of privileges also cascaded.

1.3 Database Access Control (cont.)



1.3 Database Access Control (cont.)

Role-based Access Control (RBAC)

- Unlike a file system associated with a single or a few applications, a database system often supports dozens of applications.

- In such an environment, an individual user may use a variety of applications to perform a variety of tasks, each of which requires its own set of privileges.

- It would be poor administrative practice to simply grant users all of the access rights they require for all the tasks they perform.

- RBAC provides a means of easing the administrative burden and improving security.

1.3 Database Access Control (cont.)

- In a discretionary access control environment, we can classify database users in three broad categories:
  - **Application owner:** An end user who owns database objects (tables, columns, rows) as part of an application.
  - **End user other than application owner:** An end user who operates on database objects via a particular application but does not own any of the database objects.
  - **Administrator:** User who has administrative responsibility for part or all of the database.

1.4 Inference

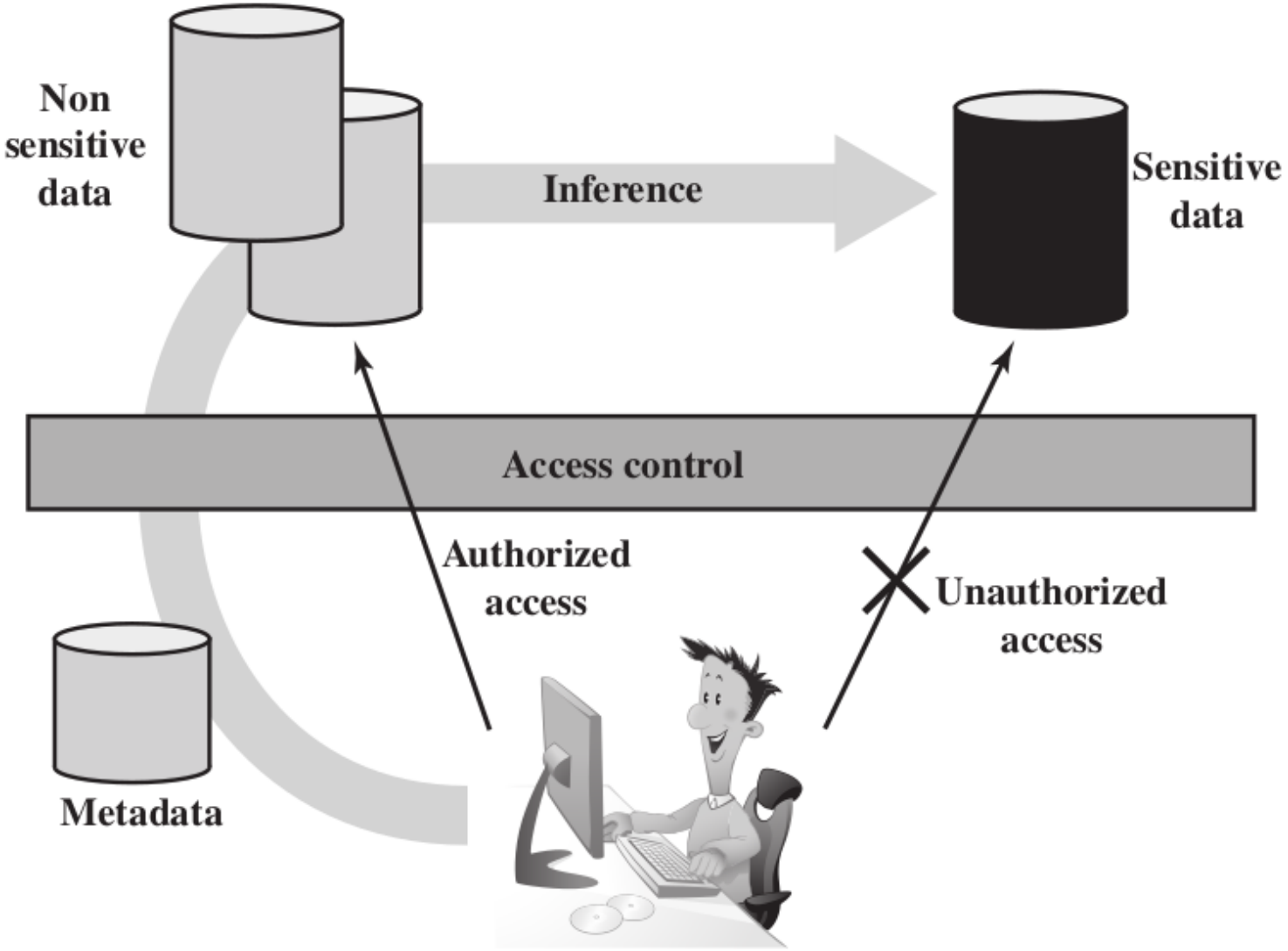
- Inference, as it relates to database security, is the process of performing authorized queries and deducing unauthorized information from the legitimate responses received.

- The inference problem arises when the combination of a number of data items is more sensitive than the individual items, or when a combination of data items can be used to infer data of a higher sensitivity.

- The attacker may make use of non-sensitive data as well as metadata. Metadata refers to knowledge about correlations or dependencies among data items that can be used to deduce information not otherwise available to a particular user.

- The information transfer path by which unauthorized data is obtained is referred to as an inference channel.

1.4 Inference (cont.)



1.4 Inference (cont.)

- There are two approaches to dealing with the threat of disclosure by inference:

- **Inference detection during database design:** This approach removes an inference channel by altering the database structure or by changing the access control regime to prevent inference. Examples include removing data dependencies by splitting a table into multiple tables or using more finegrained access control roles in an RBAC scheme. Techniques in this category often result in unnecessarily stricter access controls that reduce availability.

- **Inference detection at query time:** This approach seeks to eliminate an inference channel violation during a query or series of queries. If an inference channel is detected, the query is denied or altered.

1.5 Database Encryption

- In addition to various database security mechanisms, encryption can be used to further defend databases against attackers.

- There are two disadvantages to database encryption:

- **Key management:** Authorized users must have access to the decryption key for the data for which they have access. Because a database is typically accessible to a wide range of users and a number of applications, providing secure keys to selected parts of the database to authorized users and applications is a complex task.

- **Inflexibility:** When part or all of the database is encrypted, it becomes more difficult to perform record searching.

1.5 Database Encryption (cont.)

- Encryption can be applied to the entire database, at the record level, at the attribute level, or at the level of the individual field.
- In order to facilitate key management and ensuring flexibility, it would be better to encrypt individual records as blocks, instead of encrypting an entire database.