Database Security

Course 3

Database Security Foundations (part 2)

Discretionary aspects of the security in databases

Plan

1. Security Policies
    - 1.1 Access control policies
        - Authorization policies
        - Role-based access control
    - 1.2 Administration policies
    - 1.3 Identification and authentication
    - 1.4 Audit of a database system
    - 1.5 Views in the security context
2. Enforcement of Security Policies
    - 2.1 SQL extensions for security
    - 2.2 Query rewrite

1. Security Policies

1. Security Policies (1)

- Security policies are **sets of rules that ensure the security of the system**. These policies include 2 categories:
  - Mandatory policies – previously presented;
  - Discretionary policies.
- Early versions of commercial databases provided the discretionary type of security.

1. Security Policies (2)

We recall:

- Mandatory policies are those that are mandatory by nature, **independent** of the application.
- Discretionary policies are those that are specified by the administrator or by a person responsible for the environment in which the system will operate.

1. Security Policies (3)

- The most popular **discretionary security policy** is the one regarding the **access control**.
  - These policies have been studied for **operating systems** since the ‘60s and for databases since the ‘70s.
  - The most representative database systems, *System R* and *INGRES* were among the first to investigate access control for database systems.
  - Since then, there have been some changes to these policies.

1. Security Policies (4)

- Another class of discretionary security policies includes **administration policies**.
- We will also discuss **identification and authentication** in the frame of discretionary policies.

- Same as in the presentation of mandatory policies, the introduction of discretionary policies focuses on the relational systems.
- Most principles apply to other systems also, such as object-oriented database systems and distributed databases.

1.1. Access control policies (1)

- These policies were first investigated for operating systems.
- The key issue in the context of OSs is whether a **process** can be granted access to a **file**.
  - Access can be for **read** or **write**, and the latter can include access to edit, add or delete.
- Subsequently, these principles were transferred to database systems such as *INGRES* and *System R*.
  - Since then, various forms of access control have been studied.
  - These include **role-based policies** currently implemented in a lot of commercial systems.

1.1. Access control policies (2)

- Access control policies also include mandatory policies, discussed earlier.
- The different types of access control are shown in Figure 1 and will be described in the following.

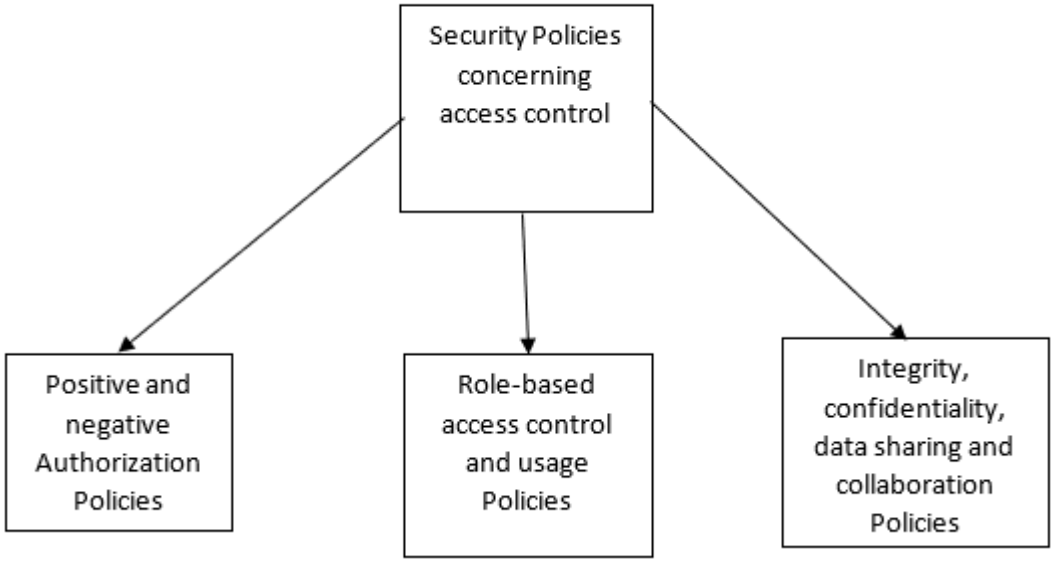

Figure 1 Access control types

1.1. Access control / Authorization policies (1)

- Most access control policies are based on authorization policies.
- These policies assume that users are granted **access to data** based on **authorization rules**.

1.1. Access control / Authorization policies (2)

Authorization rules:

- ***Positive authorizations***
  - Early systems focused on the rules that are now called positive authorization rules.
  - For example, user John is granted access to the relation *EMP,* or user Jane is granted access to the relation *DEPT*. These are the rules of access control over relations.
  - We can also grant access to attributes and tuples. For example, John has access to read the *salary* attribute and access to write the *name* attribute in the relation *EMP*.

1.1. Access control / Authorization policies (3)

Authorization rules (continued):

- ***Negative authorizations***
  - If John's access to an object is not specified, does this mean that John does not have access to that object?
  - In some systems, the non-specification of an authorization rule implicitly means negative authorization, while in other systems negative authorizations should be explicitly specified.
  - For example, we can enforce rules so that John does not have access to the relation *EMP,* or Jane does not have access to the relation *DEPT*.

1.1. Access control / Authorization policies (4)

- ***Conflict resolution***
  - When we have conflicting rules, how are they resolved?
  - For example, we can have a rule that grants John read access to the relation *EMP*.
  - We can also have a rule that does not give John read access to the attribute *salary* in *EMP*. This is a conflict.
  - Usually, the system applies the rule of the least privilege, meaning that John has access to *EMP* except the values of the attribute *salary*.

1.1. Access control / Authorization policies (5)

- ***Strong and weak authorization***
  - In the case of a **strong authorization**, the rule is valid regardless of conflicts, and in the case of a **weak** one, the rule does not apply if there is a conflict.
  - For example, if John is granted access to *EMP* and this is a strong authorization rule, and the rule by which John does not receive access to the *salary* attribute is a weak rule, we have a conflict situation.
  - This means that the strong authorization remains valid.

1.1. Access control / Authorization policies (6)

- ***Propagation of authorization rules***
  - For example, if John has read access to the relation *EMP*, does that mean that John has access to every item in *EMP*?
  - This usually happens unless there are rules that prevent the automatic propagation of an authorization rule.
  - If such a rule exists, we must explicitly establish authorization rules that specify the objects to which John has access.

1.1. Access control / Authorization policies (7)

- ***Special rules***
  - **Content** and **context** constraints are rules by which access is granted based on the content of the data or the context in which it is displayed.
  - These rules are extensions of mandatory policies but can also be applied in the frame of discretionary security.
  - For example, in the case of content-based constraints, John has read-only access only to tuples in department 100.
  - In the case of context or association constraints, John does not have read access to names and salaries taken together but may have individual access to them.
  - In the case of event-based constraints, after the elections (event), John has access to all elements of the *EMP* relationship.

1.1. Access control / Authorization policies (8)

- ***Consistency and completeness of the rules***
  - One of the problems is ensuring the consistency and completeness of the constraints.
  - Does this mean that if the rules or constraints are **inconsistent**, do we have conflict resolution rules to deal with the situation?
  - How can we ensure that all entities (attributes, relations, elements etc.) are included in users' access control rules? In other words, are the rules complete?
  - If not, what assumptions should we make about entities that have neither positive nor negative authorizations specified on them for a particular user or class of users?

1.1. Access control / Role-Based Access Control policies (1)

- ***Role-Based Access Control*** (RBAC) has become one of the most popular methods of access control.
  - The method has been implemented in commercial systems, including Oracle.
  - The idea is to grant users access based on their roles and functions.

1.1. Access control / Role-Based Access Control policies (2)

- Users need **access to data** according to their **functions**.
  - For example, a corporate president may have access to information about his vice presidents and board members, and the chief financial officer may have access to financial and subordinate employees' information.
  - A department manager may have access to data about those who work in that department and the human resources director may obtain information about the personal data of employees in the corporation.

1.1. Access control / Role-Based Access Control policies (3)

- Role-based access control is a type of **authorization policy** that depends on the user's **role** and **activities** associated to it.
- The literature contains the research directions on **hierarchies of roles**.
  - Some of the issues that arise in this area relate to how access is propagated or the possibility that one role contains another role.

1.1. Access control / Role-Based Access Control policies (4)

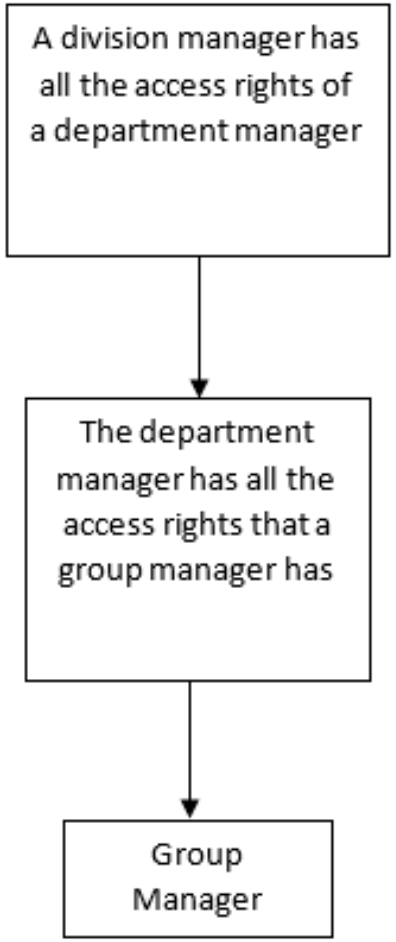

Figure 2 Roles hierarchy example

1.1. Access control / Role-Based Access Control policies (5)

- Consider the hierarchy of roles in Figure 2.
- If we grant access to a node in the hierarchy, does that **mean the access is propagated upwards**?
  - If a department manager has access to certain project information, is the access propagated to the parent node, which corresponds to a division manager?
  - If a team leader has access to the information of the employees in his team, does access propagate to the department manager (the parent in the role hierarchy)?
- What happens to the child nodes? Does access ever **propagates downwards**?
  - If a department manager has access to certain information, do his subordinates also have access to that information?
  - Are there cases in which subordinates have access to data to which the department manager does not have access?
  - What happens if an employee must report to 2 supervisors (the department manager and his project manager)?
  - What happens when the department manager works on a project and has to report to his project manager, who is led by him?
  - The multiple parents are illustrated in Figure 3, and a cycle is shown in Figure 4.

1.1. Access control / Role-Based Access Control policies (6)

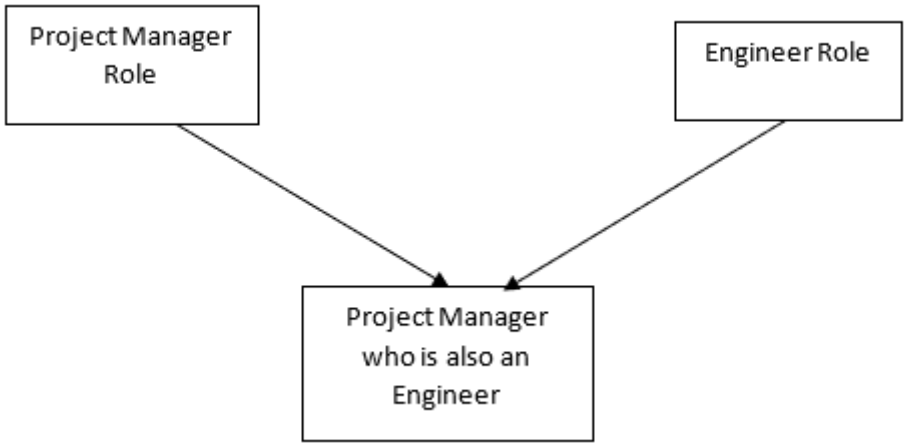

Figure 3. Multiple parents

1.1. Access control / Role-Based Access Control policies (7)

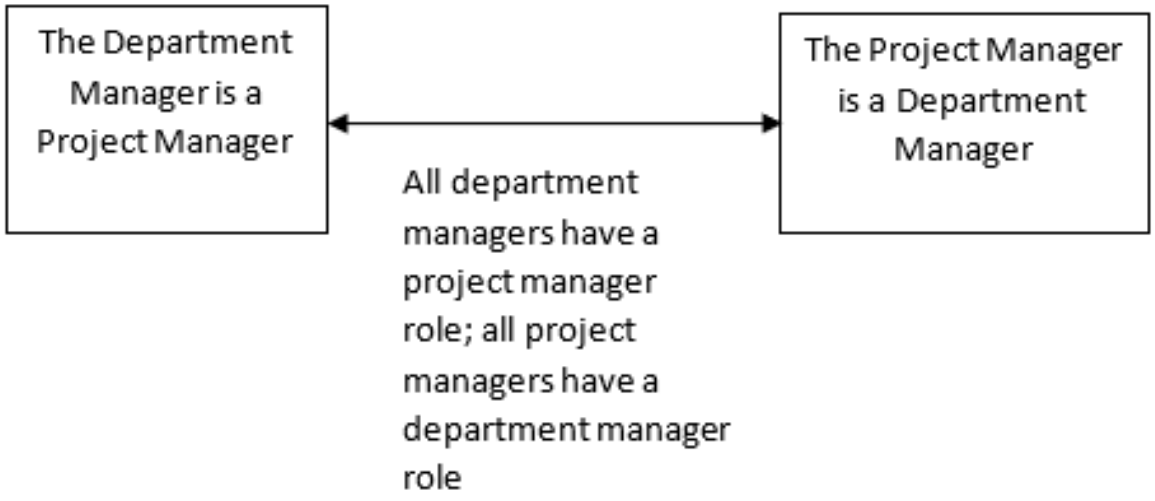

Figure 4. Cyclic graph

- Role-based access was analyzed for relational, object and distributed systems, but also for newer technologies (data warehouse, knowledge management systems, semantic web, e-commerce systems, digital libraries).

1.2. Administration Policies (1)

- Access control policies specify the access rights that users are granted to the data.
- Administration policies specify **who will administer the data**.
- Administration tasks include keeping the data in a **consistent state**, ensuring that **metadata** is updated as the data **changes**, and ensuring **recovery** from failures.

1.2. Administration Policies (2)

- The database administrator (**DBA**) is responsible for updating metadata, indexes and access methods and also for ensuring that access control rules are properly applied.
- An important role can also be played by the System Security Officer (**SSO**).
- Security issues may be the responsibility of the SSO and data issues may be the responsibility of the DBA.
- Other administration policies refer to the appointment of persons which should be responsible for the data.
  - Usually, the owners of the schemas have control over the data they create and can manage them during their existence.
  - There are situations in which the owners are not available for data management, in which case someone else is appointed to be responsible for them.

1.2. Administration Policies (3)

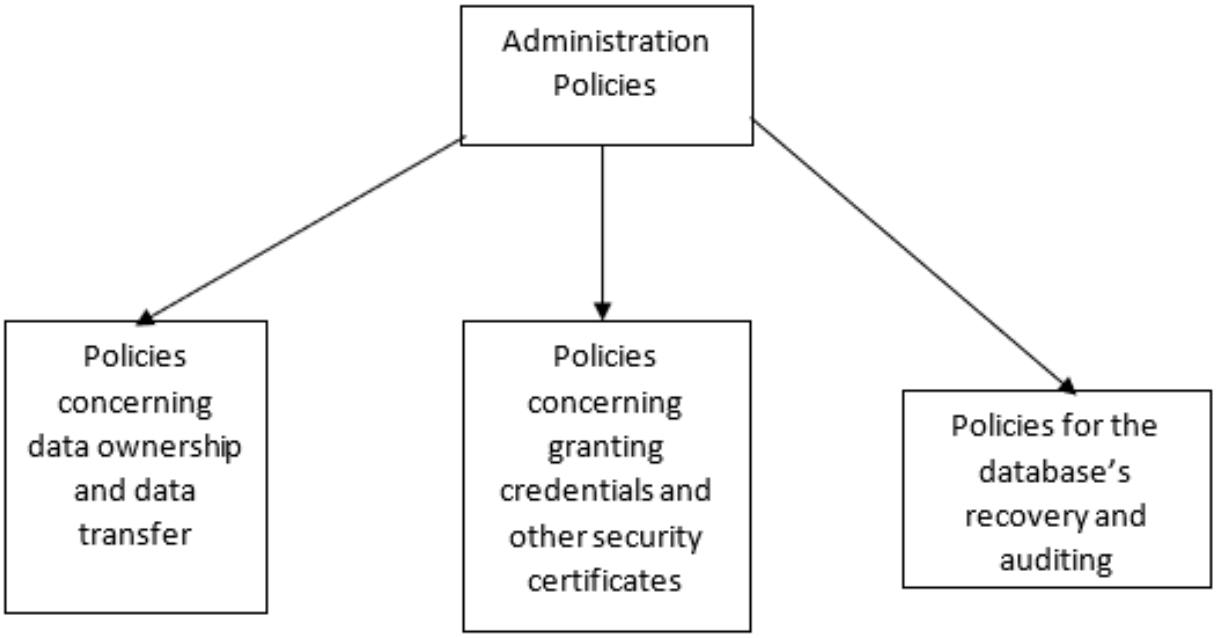

Figure 5

1.3. Identification and Authentication

- **Identification** requires users to provide a user ID and a password.
- **Authentication** is the stage in which the system must match the user ID with the password to ensure that the person is the one who pretends he is. A user can have multiple identities, depending on his roles.
- A lot of issues have been reported regarding the password-based schema. One of these is that hackers can break into the system. In a centralized system the problems are simpler than in a distributed one.
- More recently, **biometric techniques** have begun to be used. These include face and voice recognition for user authentication. The widespread use of biometric techniques is expected as these technologies evolve.

1.4. Audit of a database system (1)

- **Database audit** is performed for several purposes.
  - For example, databases can be audited in order to record:
    - Number of launched queries,
    - Number of updates,
    - Number of executed transactions,
    - Number of user connections.

- In addition to security, the aim of performing the audit is **to design the system more efficiently**.

1.4. Audit of a database system (2)

- How does the audit intervene in security?

- For example, by using the audit we can find the answers to the following questions:
  - has any access control rule been avoided, and information is provided to users?
  - did the inference problem appear?
  - has confidentiality been violated?
  - did unauthorized accesses occur?

1.4. Audit of a database system (3)

- Audits create an ***audit trail***, and audit data can be stored in a database.
- This database can be analyzed in order to detect any abnormal sequence or behavior.
- There have been many concerns about using **data mining to analyze audit** and detect unauthorized access.
- The analysis of audit information is particularly important in present, in the context of web transactions.
  - An organization should be able to initiate the analysis and determine issues such as credit card fraud and identity theft.

1.5. Views in the security context

- As a security mechanism, **views** have been studied for both mandatory and discretionary security.
  - For example, we may not want to grant access to an entire relationship, especially if it has many attributes. Therefore, the DBA could create views and grant access to them. As with mandatory security, security levels can be associated with views.

- Views have some specific problems, including the **update problem**.
  - This means that if the view is updated then we need to ensure that the base relations are updated.
  - Therefore, if a view is updated by John and **he does not have access to the base relation**, can it be updated?

2. Enforcement of Security Policies

2. Enforcement of Security Policies

- In the ‘70s, *System R* and *INGRES* developed techniques such enforcing security policies by **query modification** mechanisms.
- The **SQL** language has been extended to allow the specification of security policies and access control rules.
- More recently, languages such as XML and RDF have been extended to specify security policies.

2.1. *SQL* extensions for security (1)

- *SQL* security extensions allow the **policies' specification**.
- *SQL* was developed for **defining** and **processing** data in relational systems.
- Subsequently, various versions of *SQL* were developed, including *SQL* for objects, multimedia, and web.

2.1. *SQL* extensions for security (2)

- **Security policies** can be specified in the **data definition language**.
- The *SQL* language contains the ***GRANT*** and ***REVOKE*** commands for granting and, respectively, revoking users' access rights.
- For example, if user John receives read access to the *EMP* relationship, this can be specified by:

  GRANT read ON emp TO John;

2.1. *SQL* extensions for security (3)

- Revocation of access can be specified by:

  REVOKE read ON emp FROM John;

- *SQL* has been extended for more complex constraints, such as:
  - granting access to read a tuple from a relation to John
  - granting access for writing to an element of a relation to Jane.

2.2. Query rewrite (1)

- The query rewrite was first proposed in the *INGRES* project. The idea is to **modify the query based on constraints**.
- We consider a query from John which retrieves all the tuples in *EMP*. We assume that John has read access only to tuples for which the salary is less than 30,000 and the employee does not work in the security department.
- The query:

  *SELECT * FROM EMP;*

  will be rewritten as:

  *SELECT * FROM emp*
  Where salary < 30000
  And dept != 'Security';

2.2. Query rewrite (2)

- The *where* clause of the query contains all the constraints associated to the relation. There may be constraints that involve multiple relations.
- For example, we can have 2 relations, *EMP* and *DEPT,* linked by *Dept #*.
- The query is rewritten as follows:

*SELECT * FROM emp, dept*

WHERE emp.salary<30000

And emp.dept#=dept.dept#

And dept.name != 'Security';

2.2. Query rewrite (3)

- The high-level **algorithm** is presented below:

Input : The query *Q*, the set *S* of security constraints

Output : The rewritten query *Q'*

For *c* in *S*

If *c* relevant_for Q then

Modify the *where* clause of *Q* via a negation

End if;

End for;

Return *Q'*;

A perspective on the inference problem (1)

- The inference problem was studied in statistical databases several years before becoming a very important topic in MLS/DBMSs.

- Inference is the process of **launching queries and deducing (inferring) new information from legitimate received responses**.

- It becomes a problem if the user is not authorized to know the deduced information.

A perspective on the inference problem (2)

- Among the first to study this problem in statistical databases was the Census Bureau (USA).
  - Statistical databases are used by various organizations, from the census bureau to marketing organizations that want to study population's behavior patterns.
  - In essence, statistical databases provide sums, averages, deviations etc., very useful values in the study of the population in terms of numbers or behavior.

A perspective on the inference problem (3)

- Examples of situations in which the problem of statistical inference may appear:
  - providing average salaries, but protecting individual values;
  - health information from a particular county, protecting the individual records of persons in that county.
  - Can a user get the average/total salary of 10 people, then 9, then 8 and so on?
- A practice regarding statistical databases is not to use all the values in the database, but to work with **sample values.**
  - That is, the averages are calculated based on a representative sample.
  - From these, it is more difficult to obtain protected individual information.

A perspective on the inference problem (4)

- Statistical inference has become more important with the development of newer technologies, such as *data warehousing* and *data mining*.
  - For example, *data warehouse* technology has been developed to provide specific information for **decision making**, including sums and averages.
  - Data in the *data warehouse* may be unclassified, but there may be protected information that resides in the back-end database. Based on the information provided by the data warehouse, can the "sensitive" data in the back-end database be determined?

A perspective on the inference problem (5)

- *Data mining* provides previously unknown information, using various reasoning techniques such as **statistical inference**.
- The inference problem is relevant in the field of data mining.
- The challenge is to discover protected information by mining.
- A more recent research direction refers to ***privacy-preserving data mining***. The idea is to ensure confidentiality, but at the same time to provide information, probably slightly modified.