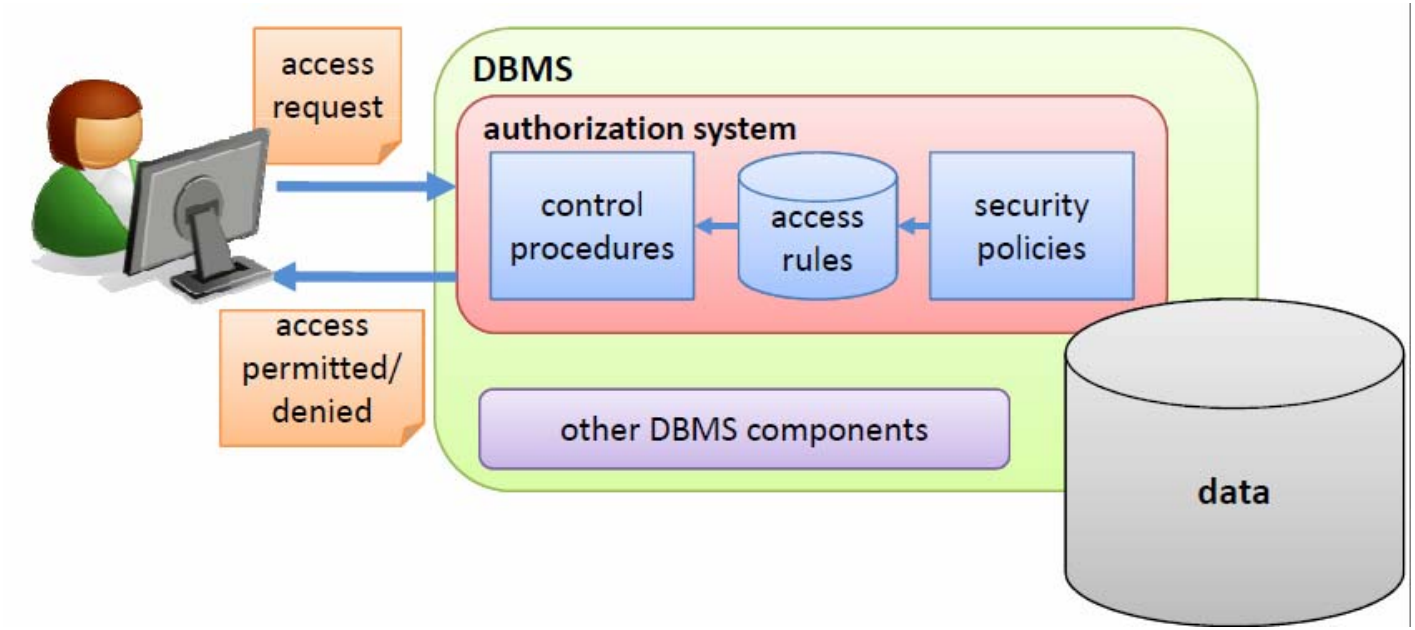
Database Security – Part 2

Access Control

Access Control

- **Access Control** refers to any means of controlling access to resources in a database.
- Can be seen as **the combination of authentication and authorization** plus **additional measures**, such as IP-based restrictions.



Authentication vs. Authorization

- **Authentication** is the process by which a system can identify users.
  - Who are the users?
  - Are the users really whom they represent themselves to be?

  – Identified by username/password, a smart card, a PIN number, a secret code sent in a letter, a fingerprint scan, and so on.

- **Authorization** is the process by which a system determines what level of access a user (who **is already authenticated**) has to secured resources.
  - Is a user authorized to access or modify a table?
  - ...

Main Approaches to Access Control

1. **Discretionary access control** (**DAC**)
   - Based on the concept of **access privileges** for giving users such privileges.
   - SQL support DAC; most commercial DBMSs also support DAC.
2. **Mandatory access control** (**MAC**)
   - Based on **system-wide policies** that cannot be changed by individual users.
   - SQL doesn't support MAC but some DBMSs support MAC.
3. **Role-based access control** (**RBAC**)
   - Based on **roles** (can be used with DAC and MAC).
   - SQL support privileges on roles; many DBMSs support RBAC.

Discretionary Access Control (DAC)

- Called **discretionary** because it allows a subject to grant other subjects privileges to access objects of the subject at its own discretion.
- DAC governs the access of subjects (e.g. accounts, etc.) to objects (relations, views, etc.) **on the basis of subjects' privileges**.
- SQL supports DAC through the `GRANT` and `REVOKE` commands.
  - `GRANT` gives privileges to users;
  - `REVOKE` takes away privileges from users.

Specifying Privileges - Grant

- The **syntax** of the GRANT command:

```
GRANT privileges ON object TO users [WITH GRANT OPTION]
```

Examples: Consider the relation schemas

SUPPLIER(id, sname, city, rating)
RATINGSTANDARD(no, description)

1. `GRANT SELECT ON Supplier TO Jerry;`
2. `GRANT INSERT, DELETE ON Supplier TO Tom;`
3. `GRANT UPDATE (rating) ON Supplier TO Tom;`
4. `GRANT REFERENCES (no) ON RatingStandard TO Bob;`

Specifying Privileges - Views

- Views provide **an important mechanism for discretionary authorization**.
- The **syntax** of creating a view:

```
CREATE VIEW view_name AS
      SELECT attribute_list
        FROM table_list
      [WHERE condition]
  [GROUP BY attribute_list [HAVING group_condition]]
  [ORDER BY attribute_list];
```

- Creating a view requires `SELECT` privilege on all relations involved in the view definition.

Specifying Privileges - Views

- **Example:** Consider the relation schema:

  SUPPLIER(id, sname, city, rating)

  How to give Bob read access to SUPPLIER *for suppliers in Paris (only), but not to supplier ratings?*

Specifying Privileges - Views

- **Example:** Consider the relation schema:

  SUPPLIER(id, sname, city, rating)

  How to give Bob read access to SUPPLIER *for suppliers in Paris (only), but not to supplier ratings?*

Step 1:
```
CREATE VIEW SUPPLIER-PARIS AS
    SELECT id, sname, city
    FROM SUPPLIER
    WHERE city='Paris';
```

Step 2:
```
GRANT SELECT ON SUPPLIER-PARIS TO Bob
```

Users of this view only see part of SUPPLIER (**horizontal subset** by applying `city='Paris'` and **vertical subset** by excluding `rating`).

Revoking Privileges - Revoke

- The **syntax** of the REVOKE command:

```
REVOKE [GRANT OPTION FOR] privileges ON object FROM users
```

Examples: Still consider the relation schema

SUPPLIER(id, sname, city, rating)

1. `REVOKE INSERT, DELETE ON SUPPLIER FROM Peter;`
2. `GRANT SELECT ON SUPPLIER TO Bob;`

   `Bob is working on the task ... and done!`

   `REVOKE SELECT ON SUPPLIER FROM Bob;`

Delegating Privileges

- **Can we pass on privileges to others?**
  - We are the object owner;
  - We have received the privilege with `GRANT OPTION`.

Example: Tom, the owner of SUPPLIER, wants to give Bob the right to grant his `SELECT` privilege on SUPPLIER to other users for one month.

```
GRANT SELECT ON Supplier TO Bob WITH GRANT OPTION;
```

```
One month later ...
```

```
REVOKE GRANT OPTION FOR SELECT ON Supplier FROM Bob;
```

Delegating Privileges - Recursive Revocation

- The privileges of an object can be given to a user *with* or *without* the `GRANT OPTION`

```
GRANT SELECT ON Supplier TO Bob;
```

```
GRANT SELECT ON Supplier TO Bob WITH GRANT OPTION;
```

- A user can only revoke privileges that he or she has granted earlier, with two optional keywords in `REVOKE` command:
  - `CASCADE`: revoking the privilege from a specified user also revokes the privileges from all users who received the privilege from that user.
  - `RESTRICT`: revoking the privilege only from a specified user.

Delegating Privileges - Recursive Revocation

- If a user receives a certain privilege from multiple sources, and the user would lose the privilege only after all sources revoke this privilege.
- **Example:**

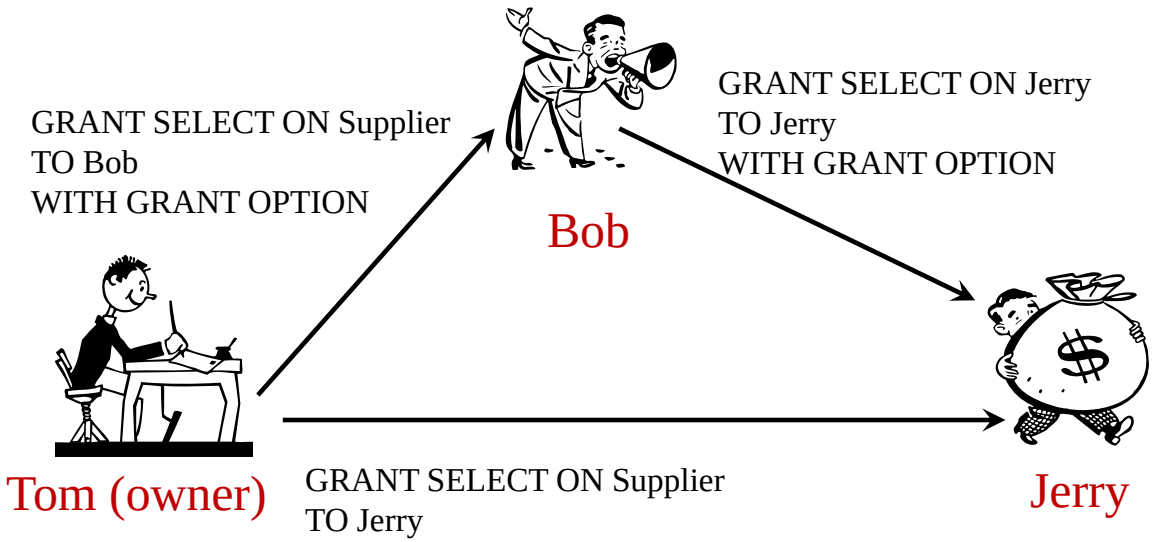
```
1.GRANT SELECT ON SUPPLIER TO Bob WITH GRANT OPTION;  (by Tom)

2.GRANT SELECT ON SUPPLIER TO Jerry;  (by Tom)

3.GRANT SELECT ON SUPPLIER TO Jerry WITH GRANT OPTION;  (by Bob)

4.REVOKE SELECT ON SUPPLIER FROM Bob CASCADE;  (by Tom)
```

- **Questions:**

1. Will Bob lose the SELECT privilege on SUPPLIER?
2. Will Jerry lose the SELECT privilege on SUPPLIER?

Delegating Privileges - Recursive Revocation

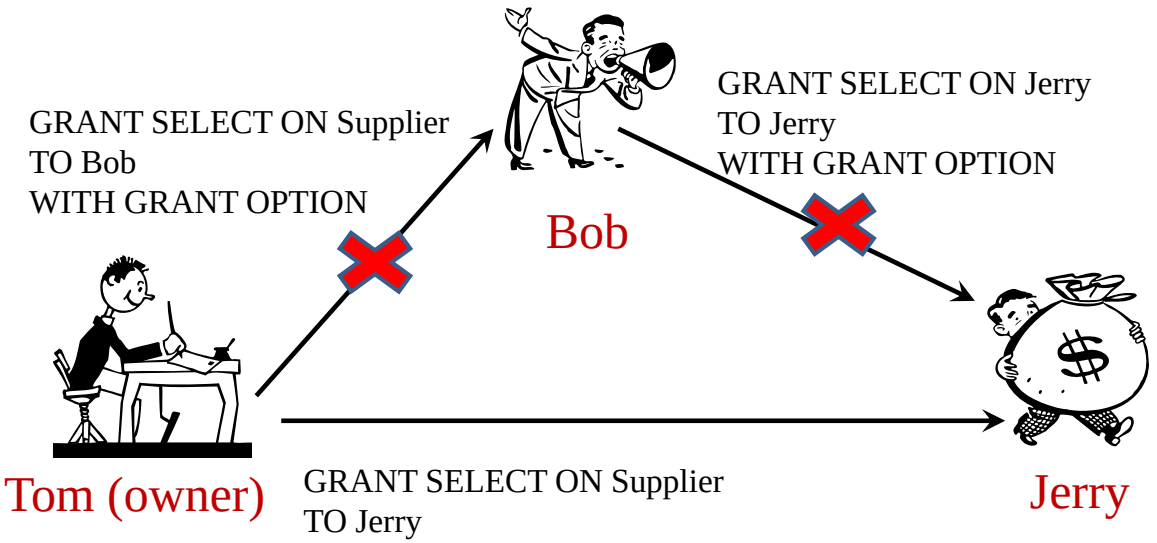
- **Example:**




```
1.GRANT SELECT ON Supplier TO Bob WITH GRANT OPTION;  (by Tom)

2.GRANT SELECT ON Supplier TO Jerry;  (by Tom)

3.GRANT SELECT ON Supplier TO Jerry WITH GRANT OPTION;  (by Bob)
```

Delegating Privileges - Recursive Revocation

- **Example:**




```
4.REVOKE SELECT ON SUPPLIER FROM Bob CASCADE;  (by Tom)
```

1. Bob will lose the privilege.
2. Jerry won't lose the privilege.

Delegating Privileges - Propagation

- There are techniques to limit the propagation of privileges. But not implemented in most DBMSs and not part of SQL.
  - Limiting **horizontal propagation**: limits that an account given the `GRANT OPTION` can grant the privilege to at most n other accounts;
  - Limiting **vertical propagation**: limits the depth of the granting privileges.

Mandatory Access Control (MAC)

- Restrict access to objects based on the **sensitivity of the information** contained in the objects and the formal **authorization** of subjects to access information of such sensitivity.
  - Sensitivity of the information (e.g., security classes)
    top secret (TS), secret (S), confidential (C), unclassified (U).

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

  - Authorization (e.g., clearances)

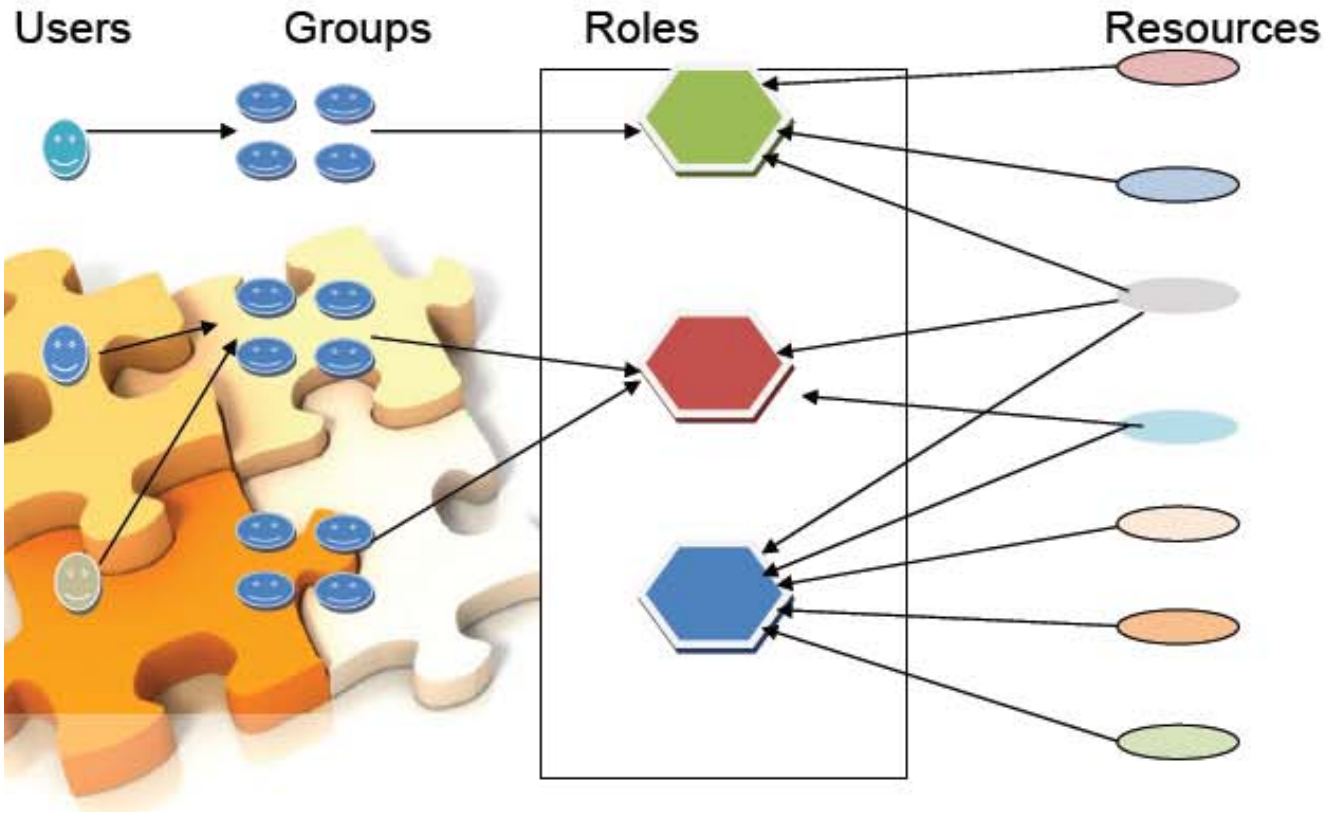
Example:

id	sname	city	rating	security class
1	S1	Paris	4	secret (S)
2	S2	Canberra	5	confidential (C)

- Bob with C clearance can only access the second tuple.
- Peter with S clearance can access both tuples.

Role-Based Access Control (RBAC)[1]

- Access rights are grouped by **roles**, and the use of resources is restricted to individuals assigned to specific roles.



[1] Comprehensive Approach to Database Security, Ajoy S. Kumar, 2008