Security and Authorization

- **Confidentiality**: Information not disclosed to non-authorized users.
- **Integrity**: Only authorized users should be allowed to modify data in expected ways.
- **Availability**: Authorized users should not be denied access.

Security Policy { Operating System, Network, DBMS.

"Who can access/modify what"

DBMS

Access control. {
- Discretionary
  - Users do it
- Mandatory
  - It can't be overriden by users

Mandatory Access Control is DBMS dependent

- DBA determines overal restrictions of users.

We do not cover it

Discretionary: Owner of an "object" determines who can access/modify it

Object can be relations, views, stored-procedures, user defined functions, databases, etc.

Authorization Ids:

Every user is identified by one or more authorization IDs.

- userid
- role: a user can belong to several roles
- PUBLIC is special auth ID.

PRIVILEGES

1. SELECT - can query
2. INSERT - insert into a relation
3. UPDATE - update tuples in relation
   - They can be further restricted to a set of attributes.

SELECT(name, addr)

allows to query only these attributes.

INSERT(name, addr) allows to insert a tuple but only to specify given attributes (rest are set to default).

④ DELETE - delete from relation

⑤ REFERENCES - the right to create a foreign key constraint on a relation.

Say relation S has FK to relation R.
User has INSERT on S but no priv. on R.
- Every insertion into S requires a lookup into R.

By attempting to insert user can figure if a value is present in R!!.

Ex.

Ultra secret table Invited(sid) with the sid of students.

Johnny has no privilege on it. He can't read it

Johnny creates a new relation MyInvited(sid) with Foreign Key to Invited(sid)

Johnny can now try to insert to MyInvited <u>every</u> potential sid. If rejected, sid not in Invited.

For this reason, we need a special privilege to create Foreign Key constraints.

Why not simply require SELECT (sid) on Invited)?

It would be straightforward to query Invited.

REFERENCES only allows indirect look ups.

Important: REFERENCES is required only on the table beign referenced, and only by the user creating the Foreign Key.

- User Bob creates Invited
- For user Johny to be able to create Foreign Key from MyInvited to Invited, Johnny requires REFERENCES on Invited
- To be able to have INSERT or UPDATE on MyInvited a user does not need REFERENCES on Invited.

(6) USAGE. Applies to non-relation objects.

Right to use it

Non relevant for our course.

⑦ TRIGGER. Right to add a trigger on a relation

A trigger is likely to require one or more privileges to work.

- User creating trigger must have Privileges
- User executing trigger does not need them.

⇒ trigger is executed under privileges of creator of trigger.

⑧ EXECUTE. Right to execute a certain funtion or stored procedure.

⑨ UNDER. Right to define a subtype of a given type.

The creator of an DB element (table, view, UDF) is its owner.

- Owner has all privileges on object
- Owner (and dba) can give privileges to others.

Privilege Checking Process

Any DB operation involves:

- The database elements on which the operation is performed
- The agent that causes the operation
  - can be a user or a process
  - has a current authorization ID

The operation is executed only if the current authorization ID has all the privileges needed to perform the operation.

Example:

Assume table Students(sid, sname), Courses(cid, cname) and Enrolled(cid, sid)
Enrolled has FK references to both Students and Courses.

1) To create Enrolled we need REFERENCES in Students and Courses.

We only use REFERENCES when we create a relation.

2) To { query, insert, delete, update } we need { SELECT, INSERT, DELETE, UPDATE }

On either relation!!

REFERENCES only needed at creation.

3)
```
DELETE FROM Enrolled
  WHERE sid = (SELECT sid FROM
               STUDENTS S
               WHERE S.name
                 = 'Bob');
```

Minimum privileges required

DELETE on Enrolled

SELECT on Enrolled (sid)

- We need to query Enrolled to to find tuple to delete!!.

SELECT on Students (sid, sname)

4) UPDATE Students SET
sname = 'abc'

Min priv. required:
UPDATE Students (sname)

5) INSERT INTO Enrolled
VALUES ('123', 'xyz')

Requires only INSERT on Enrolled.
Look-ups do not require SELECT on Students nor Courses. They are allowed because the FK constraints

6) INSERT INTO Students (sid)
VALUES ('392')

Requires only INSERT on Students (sid)

Now assume that FK constraints were created with ON DELETE CASCADE

7) DELETE FROM Students
WHERE sid = '234';

If table Enrolled has one or more tuples for sid = '234'
We would require:
- DELETE ON Students
- SELECT ON Students
- DELETE ON Enrolled