Security and Authorization

Introduction to DB Security

- Secrecy: Users should not be able to see things they are not supposed to.
  - E.g., A student can’t see other students’ grades.
- Integrity: Users should not be able to modify things they are not supposed to.
  - E.g., Only instructors can assign grades.
- Availability: Users should be able to see and modify things they are allowed to.

Types of Access Control

- Discretionary Access Control
- Mandatory Access Control

Discretionary Access Control

- Based on the concept of access rights or privileges for objects (tables and views), and mechanisms for giving users privileges (and revoking privileges).
- Creator of a table or a view automatically gets all privileges on it.
- DMBS keeps track of who subsequently gains and loses privileges.

Role-Based Authorization

- In SQL-92, privileges are actually assigned to authorization ids, which can denote a single user or a group of users.
- In SQL:1999 (and in many current systems), privileges are assigned to roles.
  - Roles can then be granted to users and to other roles.
  - Reflects how real organizations work.
  - Illustrates how standards often catch up with “de facto” standards embodied in popular systems.

Roles and Users

- Privileges can be assigned users as well as roles.
- Roles can then be granted to users and to other roles.
- Reflects how real organizations work.
- Example.
  - CREATE ROLE some_role;
  - GRANT SELECT ON Reserves TO some_role;
  - GRANT INSERT ON Sailors TO some_role;
  - GRANT UPDATE ON Boats TO some_role;
  - GRANT some_role TO Michael;
  - GRANT some_role TO Bill;

Roles and Users … (2)

- CREATE a <u>role</u> and GRANT provileges to that role.
- CREATE <u>users</u> and then GRANT <u>role</u> to the users.
  - A role called admin can have all privileges
  - A role called hr_user can have privileges to hr tables and views.
  - A role called fin_user can have privileges to finance tables and views.
  - All hr users are GRANTed hr_user role and finance users granted fin_user role.
- The privileges of the roles are cascaded to the users who are GRANTed that role.

GRANT Command

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

- The following privileges can be specified:
  - ❖ SELECT: Can read all columns (including those added later via ALTER TABLE command).
  - ❖ INSERT(col-name): Can insert tuples with non-null or non-default values in this column.
    - ❖ INSERT means same right with respect to all columns.
  - ❖ DELETE: Can delete tuples.
  - ❖ REFERENCES (col-name): Can define foreign keys (in other tables) that refer to this column.
- If a user has a privilege with the GRANT OPTION, can pass privilege on to other users (with or without passing on the GRANT OPTION).
- Only owner can execute CREATE, ALTER, and DROP.

GRANT and REVOKE of Privileges

- GRANT INSERT, SELECT ON Sailors TO Horatio
  - Horatio can query Sailors or insert tuples into it.
- GRANT DELETE ON Sailors TO Yuppy WITH GRANT OPTION
  - Yuppy can delete tuples, and also authorize others to do so.
- GRANT UPDATE (*rating*) ON Sailors TO Dustin
  - Dustin can update (only) the *rating* field of Sailors tuples.
- GRANT SELECT ON ActiveSailors TO Guppy, Yuppy
  - This does NOT allow the 'uppies to query Sailors directly!
- REVOKE: When a privilege is revoked from X, it is also revoked from all users who got it *solely* from X.

Schema and Users for Examples

- Schema
  - Sailors(sid, sname, rating, age)
  - Boats(bid, bname, color)
  - Reserves(sid, bid, day)
- Users
  - Joe
  - Yuppy
  - Michael
  - Guppy
  - Leah

Example

- Suppose Joe has created the the three tables
- Joe now executes the following:
  - GRANT INSERT, DELETE ON Reserves TO Yuppy WITH GRANT OPTION;
- Yuppy can now insert or delete Reserves rows and authorize someone else to do the same.
- Joe further executes:
  - GRANT SELECT ON Reserves TO Michael;
  - GRANT SELECT ON Sailors TO Michael WITH GRANT OPTION;
- Michael can now execute SELECT queries on Sailors and Reserves, and he can pass this privilege to others for Sailors but not for Reserves.

Example ... (2)

- With the SELECT privilege, Michael can create a view that accesses the Sailors and Reserves tables, for example, the ActiveSailors view:

```
CREATE VIEW ActiveSailors (name, age, day) AS
SELECT S.sname, S.age, R.day
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid AND S.rating > 6;
```

- However, Michael cannot grant SELECT on ActiveSailors to others. Why?
- A user who creates a view has precisely those privileges on the view that he has on every one of the views or base tables used to define the view.
- Since Michael doesn't have a privilege to grant SELECT on Reserves he can't grant SELECT on ActiveSailors.

Example … (3)

- On the other hand, suppose that Michael creates the following view:

```
CREATE VIEW YoungSailors (sid, age, rating) AS
SELECT S.sid, S.age, S.rating
FROM Sailors S
WHERE S.age < 18;
```

- The only underlying table is Sailors, for which Michael has SELECT with grant option. Therefore he can pass this on to Eric and Guppy:
  - GRANT SELECT ON YoungSailors TO Eric, Guppy;
- Eric and Guppy can now execute SELECT queries on view YoungSailors.
- Note, however, that Eric and Guppy don’t have the right to execute SELECT queries directly on the underlying Sailor table.

Example … (4)

- Suppose now Joe executes:
  - GRANT UPDATE (rating) ON Sailors TO Leah;
- Leah can update only the rating column of Sailors. E.g.
  - UPDATE Sailors S SET S.rating = 8;
- However, she cannot execute:
  - UPDATE Sailors S SET S.age = 25;
- She cannot execute either:
  - UPDATE Sailors S SET S.rating = S.rating-l;
- Why? – Select privilege?

REVOKE

- REVOKE [GRANT OPTION FOR] privileges ON object FROM users {RESTRICT | CASCADE}

- Suppose Joe is the creator of Sailors and executes:
  - GRANT SELECT ON Sailors TO Art WITH GRANT OPTION
- And then Art executes:
  - GRANT SELECT ON Sailors TO Bob WITH GRANT OPTION
- Finally Joe executes
  - REVOKE SELECT ON Sailors FROM Art CASCADE
- Art loses the SELECT privilege on Sailors.
- Then Bob, who received this privilege from Art, and only Art, also loses this privilege.

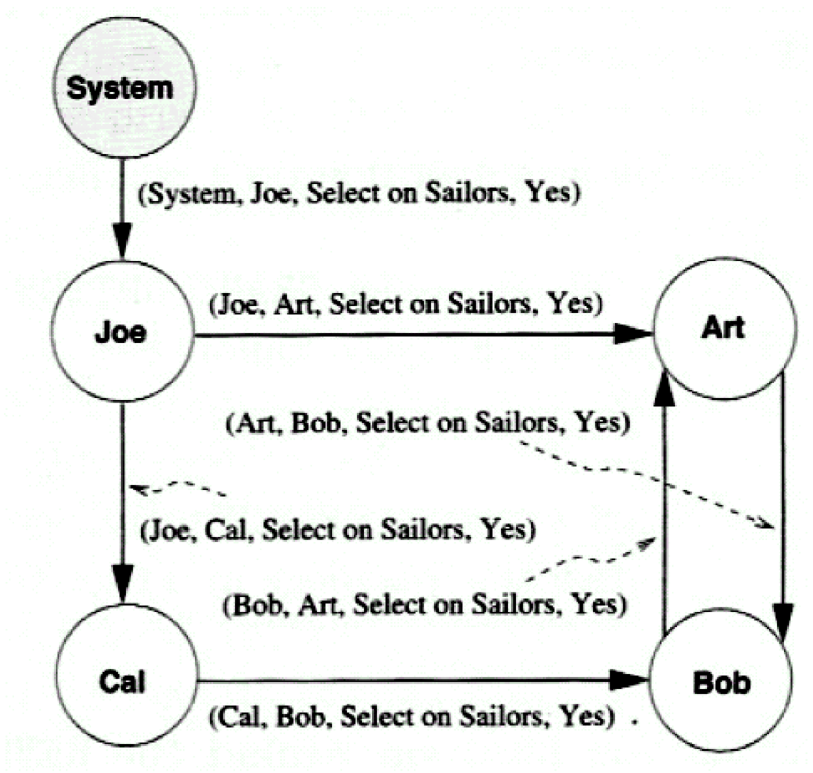
REVOKE …(2)

- Then Bob, who received this privilege from Art, and only Art, also loses this privilege.
  - Bob’s privilege is said to be abandoned

- When CASCADE is specified, all abandoned privileges are also revoked
- Possibly causing privileges held by other users to become abandoned and thereby revoked recursively.
- If the RESTRICT keyword is specified, the command is rejected if revoking privileges causes other privileges becoming abandoned.

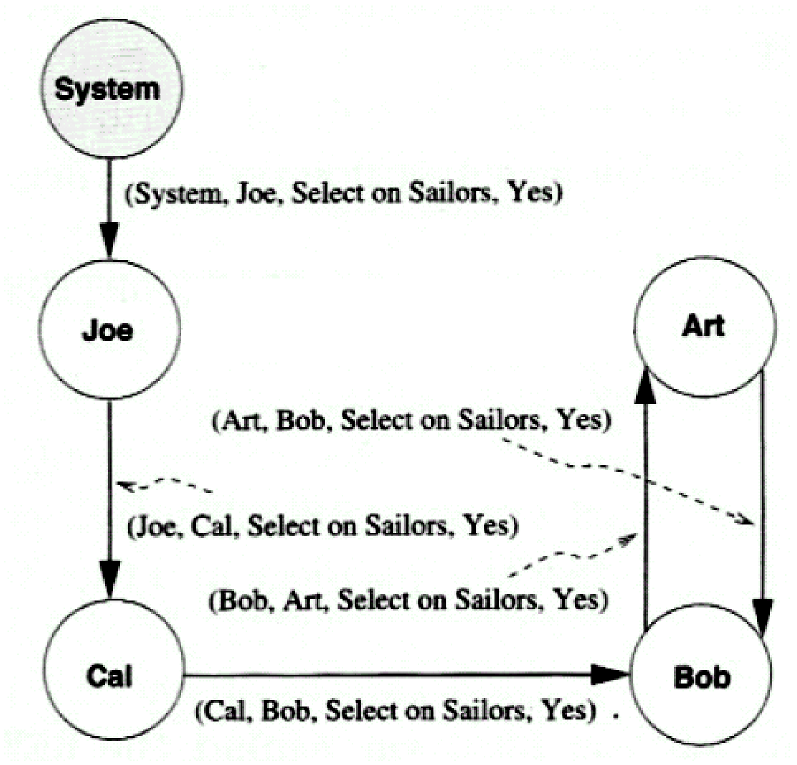
Authorization Graphs

- Nodes are users. Arcs indicate how privileges are passed.
- GRANT SELECT ON Sailors TO Art WITH GRANT OPTION (executed by Joe)
- GRANT SELECT ON Sailors TO Bob WITH GRANT OPTION (executed by Art)
- GRANT SELECT ON Sailors TO Art WITH GRANT OPTION (executed by Bob)
- GRANT SELECT ON Sailors TO Cal WITH GRANT OPTION (executed by Joe)
- GRANT SELECT ON Sailors TO Bob WITH GRANT OPTION (executed by Cal)


System
(System, Joe, Select on Sailors, Yes)
Joe
(Joe, Art, Select on Sailors, Yes)
Art
(Art, Bob, Select on Sailors, Yes)
(Joe, Cal, Select on Sailors, Yes)
(Bob, Art, Select on Sailors, Yes)
Cal
Bob
(Cal, Bob, Select on Sailors, Yes)

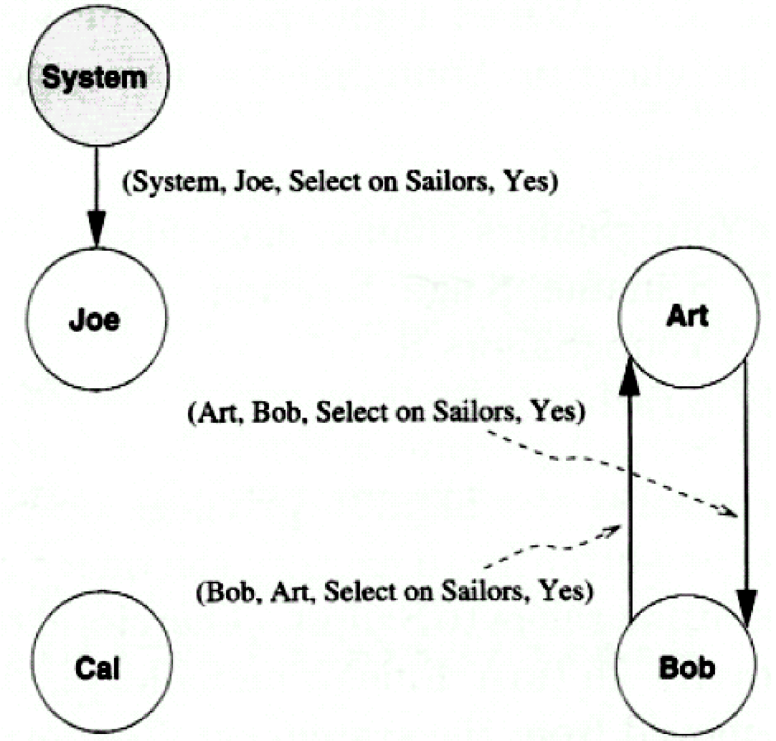
Effect of Revoke

- Suppose that Joe executes:
  - REVOKE SELECT ON Sailors FROM Art CASCADE
- The arc from Joe to Art is removed.
- Art still has the privilege
- He got it independently from Bob.


System
(System, Joe, Select on Sailors, Yes)
Joe
Art
(Art, Bob, Select on Sailors, Yes)
(Joe, Cal, Select on Sailors, Yes)
(Bob, Art, Select on Sailors, Yes)
Cal
Bob
(Cal, Bob, Select on Sailors, Yes)

Authorization Graphs

- Let’s suppose now that Joe decides to revoke Cal's SELECT privilege as well.
- The arc from Joe to Cal is removed.
- The arc from Cal to Bob is removed as well, since there is no longer a path from System to Cal.
- Art and Bob also have lost privileges as well because there isn’t a path from the System.


System
(System, Joe, Select on Sailors, Yes)
Joe
Art
(Art, Bob, Select on Sailors, Yes)
(Bob, Art, Select on Sailors, Yes)
Cal
Bob

GRANT/REVOKE on Views

- If the creator of a view loses the SELECT privilege on an underlying table, the view is dropped!
- If the creator of a view loses a privilege held with the grant option on an underlying table, (s)he loses the privilege on the view as well; so do users who were granted that privilege on the view!

Views and Security

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
    - Given ActiveSailors, but not Sailors or Reserves, we can find sailors who have a reservation, but not the *bid*’s of boats that have been reserved.
- Creator of view has a privilege on the view if (s)he has the privilege on all underlying tables.
- Together with GRANT/REVOKE commands, views are a very powerful access control tool.

Mandatory Access Control

- Based on system-wide policies that cannot be changed by individual users.
- Each DB object is assigned a security class.
- Each subject (user or user program) is assigned a clearance for a security class.
- Rules based on security classes and clearances govern who can read/write which objects.
- Most commercial systems do not support mandatory access control. Versions of some DBMSs do support it; used for specialized (e.g., military) applications.

Why Mandatory Control?

- Discretionary control has some flaws, e.g., the Trojan horse problem:
  - John creates table Heroes and gives INSERT privileges to Justin (who doesn’t know about this).
  - John modifies the code of an application program used by Justin to additionally write some secret data to table Heroes.
  - Now, John can see the secret info.
- The modification of the code is beyond the DBMS’s control, but it can try and prevent the use of the database as a channel for secret information.

Bell-LaPadula Model

- Objects (e.g., tables, views)
- Subjects (e.g., users, user programs)
- Security classes:
  - Top secret (TS), secret (S), confidential (C), unclassified (U):
  - TS > S> C > U
- Each object and subject is assigned a class.
- Subject S can read object O only if class(S) >=class(O)
  - (Simple Security Property)
- Subject S can write object O only if class(S) <= class(O)
  - (*-Property)

Intuition

- Idea is to ensure that information can never flow from a higher to a lower security level. E.g.,
  - If John has security class C, Justin has class S, and the secret table has class S:
  - John’s table, Heroes, has John’s clearance, C.
  - Justin’s application has his clearance, S.
  - So, the program cannot write into table Heroes.
- Never read a higher security level object but, can write into it.

Questions