SQL Authorization

- Privileges
- Grant and Revoke
- Grant Diagrams

Authorization (Ch 10.1)

- A file system:
  - Identifies certain privileges on the objects (files) it manages.
    - Typically read, write, execute.
  - Identifies certain participants to whom privileges may be granted.
    - Typically the owner, a group, all users.

Privileges – (1)

- SQL identifies a more detailed set of privileges on objects (relations) than the typical file system.
- Nine privileges in all are defined, some of which can be restricted to specific attributes of a relation.

Privileges – (2)

- Some important privileges on a relation (see the text for the complete list):
  1. SELECT = right to query the relation.
  2. INSERT = right to insert tuples.
  3. DELETE = right to delete tuples.
  4. UPDATE = right to update tuples.

- SELECT, INSERT, and UPDATE may apply to only a specified subset of the attributes.
- A relation may be either a base table or view

Example: Privileges

- Consider the statement below:

```
INSERT INTO Beers(name)
    SELECT beer FROM Sells
    WHERE NOT EXISTS
        (SELECT * FROM Beers
         WHERE name = beer);
```

beers that do not appear in Beers but are in Sells. We add them to Beers with a NULL manufacturer.

- We require privileges SELECT on Sells and Beers, and INSERT on Beers or Beers.name.

Database Objects

- The objects on which privileges exist include stored tables and views.
- Other privileges give the right to create objects of a type, e.g., triggers.
- Views are another important tool for access control.

Example: Views as Access Control

- We might not want to give the SELECT privilege for salary on

  Emps(name, addr, salary).

- It is safer to give SELECT on:

  ```
  CREATE VIEW SafeEmps AS
      SELECT name, addr FROM Emps;
  ```

- Queries on SafeEmps do not require SELECT on Emps, just on SafeEmps.

Triggers and Privileges

- Triggers can be tricky wrt privileges
- A user with a TRIGGER privilege for a relation can attempt to create any trigger for that relation
- However the condition and action part of a trigger may refer to other relations
  - The user needs the appropriate privileges for those relations also
- But when someone does something that awakens a trigger, they don't need the privileges for the condition and action parts of the trigger
- E.g.: Recall the INSTEAD OF example for inserting into a view.

Authorization ID's

- A user is referred to by *authorization ID*, typically their login name.
- There is an authorization ID called *PUBLIC*.
  - Granting a privilege to PUBLIC makes it available to any authorization ID.

Granting Privileges: General Points

- A user has all possible privileges on the objects (such as relations) that they create.
- The object owner may grant privileges to other users (authorization ID's), including PUBLIC.
- The object owner may also grant privileges WITH GRANT OPTION, which lets the grantee also grant this privilege.

The GRANT Statement

- To grant privileges:

  GRANT <list of privileges>

  ON <relation or other object>

  TO <list of authorization ID's>;

- If you want the recipient(s) to be able to pass the privilege(s) to others add:

  WITH GRANT OPTION

Example: GRANT

- Suppose you are the owner of Sells. You may say:

  GRANT SELECT, UPDATE(price)

  ON Sells

  TO sally;

- Now Sally has the right to issue any query on Sells
- She can update the price component only.

Example: Grant Option

- Suppose we also grant:

  GRANT UPDATE ON Sells TO sally

  WITH GRANT OPTION;

- Now, Sally not only can update any attribute of Sells, but she can grant to others the privilege

  UPDATE ON Sells.

  - Also, she can grant more specific privileges like

    UPDATE(price) ON Sells.

Revoking Privileges

- Form:

  REVOKE <list of privileges>
  ON <relation or other object>
  FROM <list of authorization ID's>;

- Your granting of these privileges can no longer be used by these users to justify their use of the privilege.
  - But they may still have the privilege because they obtained it from elsewhere.

REVOKE Options

- We must append to the REVOKE statement either:
  1. CASCADE. Now, any grants made by a revokee are also not in force, no matter how far the privilege was passed.
  2. RESTRICT. If the privilege has been passed to others, the REVOKE fails as a warning that something else must be done to "chase the privilege down."

Grant Diagrams

- It is useful to represent grants and privileges by means of a graph called a grant diagram.
- Nodes = user / privilege / grant option (y/n) / is owner (y/n)
  - Note that:

    UPDATE ON R,

    UPDATE(a) ON R, and

    UPDATE(b) ON R

    are represented by different nodes.
  - Also:

    SELECT ON R and

    SELECT ON R WITH GRANT OPTION

    similarly are represented by different nodes.
- Edge *X* ->*Y* means that node *X* was used to grant *Y*.

Notation for Nodes

- Use *AP* for the node representing authorization ID *A* with privilege *P*.
  - *P* * = privilege *P* with grant option.
  - *P* ** = the owner/source of the privilege *P*.
    - I.e., *A* is the owner of the object on which *P* is a privilege.
    - Note ** implies grant option.

Manipulating Edges – (1)

- When *A* grants *P* to *B*, we draw an edge from *AP** or *AP*** to *BP*.
  - Or to *BP** if the grant is with grant option.
- If *A* grants a subprivilege *Q* of *P* to *B* (say, UPDATE(a) on R when *P* is UPDATE ON R) then the edge goes to *BQ* or *BQ**, instead.

Manipulating Edges – (2)

- Fundamental rule:

  User *C* has privilege *Q* as long as

  1. there is a path from *XP*** to *CQ*, *CQ**, or *CQ***, and
  2. *P* is a superprivilege of *Q*.

- P is a superprivilege of Q if having P implies a user has Q
- Remember that *P* could be *Q*, and *X* could be *C*.

Manipulating Edges – (3)

- If *A* revokes *P* from *B* with the CASCADE option, delete the edge from *AP* to *BP*.
- But if *A* uses RESTRICT instead, and there is an edge from *BP* to anywhere, then reject the revocation and make no change to the graph.

Manipulating Edges – (4)

- Having revised the edges, we must check that each node has a path from some ** node, representing ownership.
  - This strategy also handles cycles in granting.
- Any node with no such path represents a revoked privilege and is deleted from the diagram.

Example: Grant Diagram

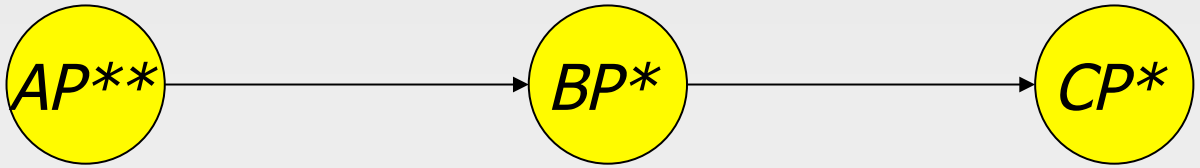
A owns the object on which P is a privilege

Example: Grant Diagram

A owns the object on which P is a privilege

A:
GRANT P
TO B WITH
GRANT OPTION

Example: Grant Diagram

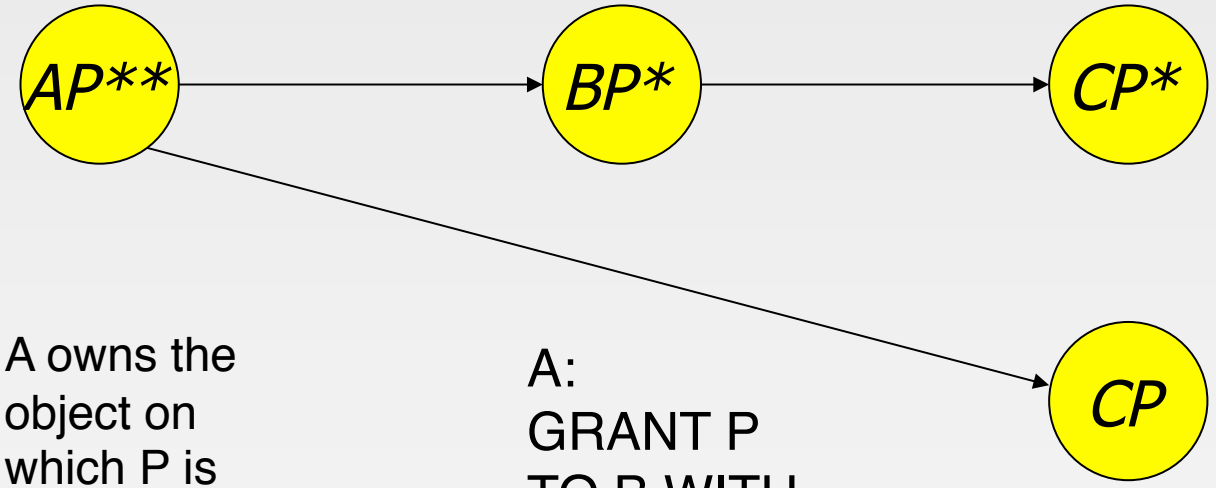


B:
GRANT P
TO C WITH
GRANT OPTION

A owns the object on which P is a privilege

A:
GRANT P
TO B WITH
GRANT OPTION

Example: Grant Diagram



B:
GRANT P
TO C WITH
GRANT OPTION

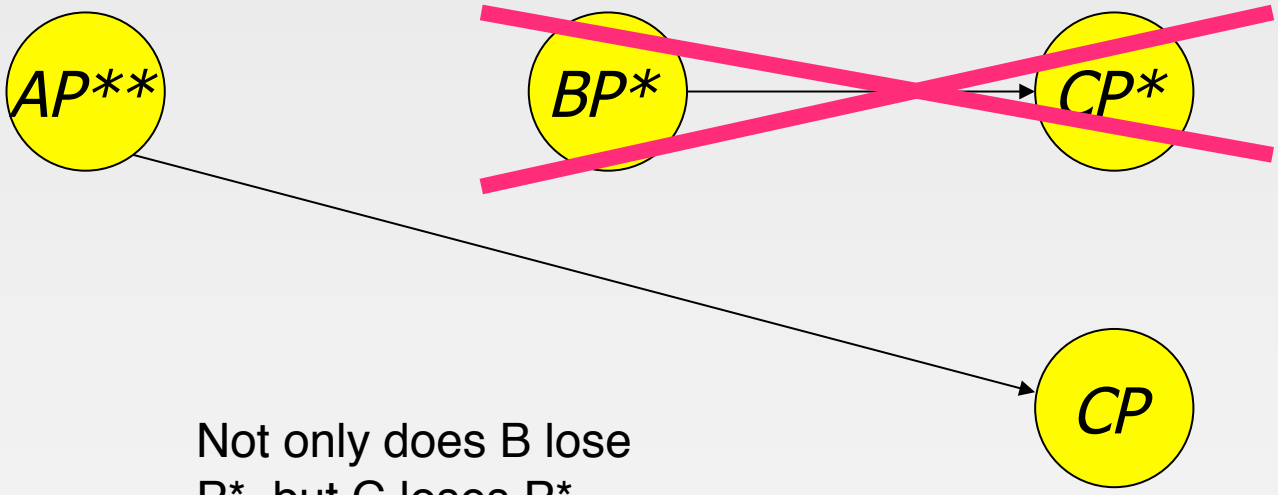
A owns the
object on
which P is
a privilege

A:
GRANT P
TO B WITH
GRANT OPTION

A:
GRANT P
TO C

Example: Grant Diagram

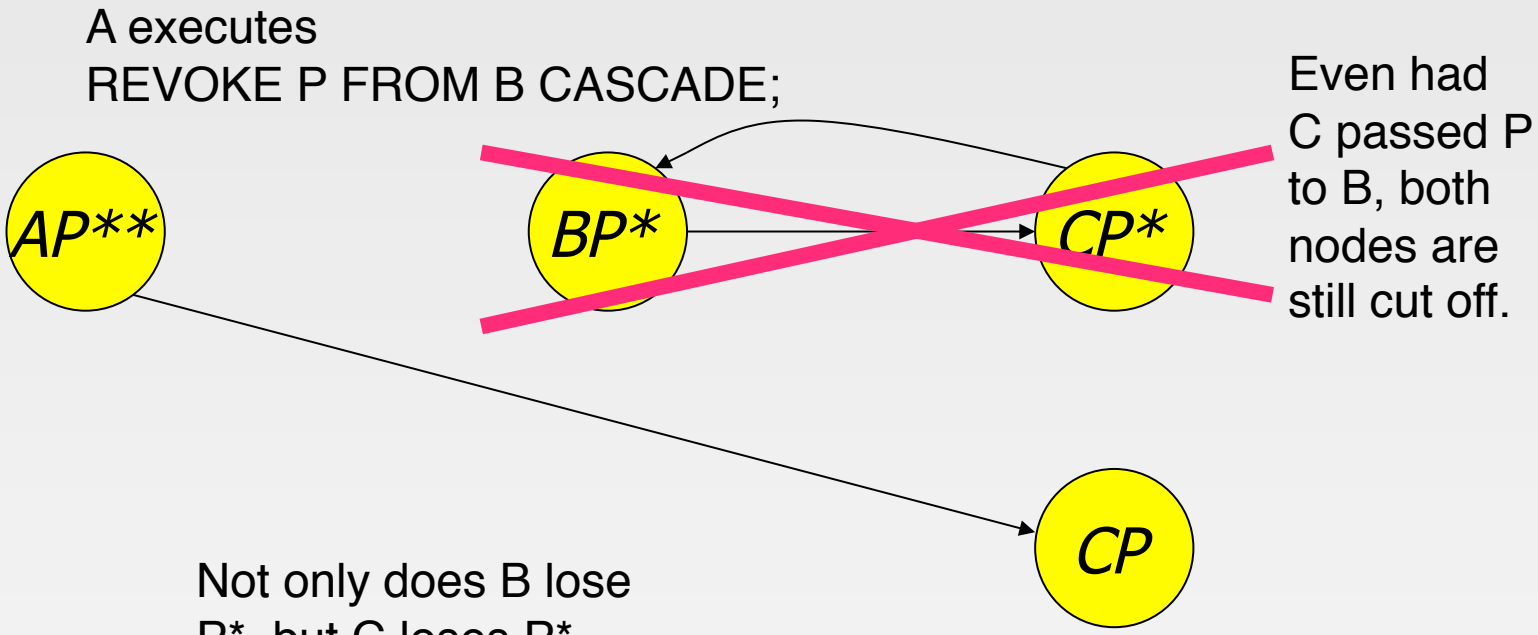
A executes
REVOKE P FROM B CASCADE;



Not only does B lose P*, but C loses P*. Delete BP* and CP*.

However, C still has P without grant option because of the direct grant.

Example: Grant Diagram



Subtle Cases

- Consider the sequence:

  U: GRANT INSERT ON R TO V

  U: GRANT INSERT(A) ON R TO V

  U: REVOKE INSERT ON R FROM V RESTRICT

- V retains INSERT(A) privilege from U

Subtle Cases (2)

- Consider the sequence:

  U: GRANT p TO V WITH GRANT OPTION

  V: GRANT p TO W

  U: REVOKE GRANT OPTION FOR p FROM V CASCADE

- V retains p privilege, but W does not.
- When U revokes the grant option, a new node is created for V having the p privilege but without the grant option.
  - Then the arc from U/p** to V/p* is deleted, disconnecting the W/p node

End of SQL Authorization