Lecture 6

Server and Security

(part 2)

OUTLINES

Security and User Authorization in SQL

SQL Authorization

Privileges

Grant and Revoke

Grant Diagrams

Authorization

- A file system identifies certain privileges on the objects (files) it manages.
  - Typically read, write, execute.
- A file system 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, some of which can be restricted to one column of one relation.

Privileges – (2)

- Some important privileges on a relation:
  1. SELECT [on relation|attributes list] = right to query the relation. SELECT(attributes list), only those attributes may be seen(select) in a selection
  2. INSERT [on relation|attributes list] = right to insert tuples.

     May apply to only one attribute.

     How about other attributes?
  3. DELETE = right to delete tuples.
  4. UPDATE [on relation|attributes list] = right to update tuples.

     May apply to only one attribute, only those attributes may be updated

Leave other components to take their default value or NULL

Privileges – (3)

5. REFERENCE [on relation| attributes list]= right to be referenced in a constraint.

6. USAGE [schema elements other than relation and assertion]= right to use that element in one's own declaration.

7. TRIGGER [on relation]= right to define trigger on that relation.

8. EXECUTE = right to execute a piece of code, such as a PSM procedure or function.

9. UNDER= right to create subtype of a given type.

Example: Privileges

- For the statement below:

```
INSERT INTO Studio(name)
    SELECT DISTINCT studioName
    FROM Movies
    WHERE studioName NOT IN
        (SELECT name
         FROM Studio);
```

studios that do not appear in Studio. We add them to Studio with other attributes being NULLs.

- We require privileges SELECT on Movies and Studio, and INSERT on Studio or INSERT(name) on Studio

Database Objects

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

Example: Views as Access Control

- We might not want to give the SELECT privilege on Emps(name, addr, salary).
- But 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.

Creating Privileges

- The owner (of something) has all privileges associated with that thing.
- Three points at which ownership is established in SQL
  - When a schema is created
  - When a session is initiated by a CONNECT statement with AUTHORIZATION userName
  - When a module is created with AUTHORIZATION userName

Authorization ID's

- A user is referred to by *authorization ID*, typically their login name.
- There is an authorization ID PUBLIC.
  - Granting a privilege to PUBLIC makes it available to any authorization ID.
- Current authorization ID
  - The module authorization ID, if the module has an authorization ID, or
  - The session authorization ID if not

Privilege-checking process

- SQL operation can be executed only if the current authorization ID possesses all the privileges needed to carry out the operation on the database elements involved.
  - Current authorization ID = the owner of the data;
  - Current authorization ID has been granted by the owner of the data, or if the privileges have been granted to user PUBLIC;
  - EXECUTE privilege + In a module owned by the owner of the data or the granted user
  - The session authorization ID is the owner of the data or the granted user + a publicly available module

Granting Privileges

- You have all possible privileges on the objects, such as relations, that you create. (Owner)
- You may grant privileges to other users (authorization ID's), including PUBLIC.
- You may also grant privileges WITH GRANT OPTION, which lets the grantee also grant this privilege.

The GRANT Statement

- To grant privileges, say:

```
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 Studio. You may say:

```
GRANT SELECT, INSERT
ON Studio TO kirk;
```

- Now kirk has the right to issue any query on Studio and can insert a new tuple into Studio.

Example: Grant Option

- Suppose we also grant:

```
GRANT INSERT ON Studio TO kirk
      WITH GRANT OPTION;
```

- Now, kirk not only can insert a tuple into Studio, but can grant to others the privilege INSERT ON Studio.
  - Login in as kirk
  - `GRANT INSERT ON Studio TO sisko;`
  - Also, kirk can grant more specific privileges like `INSERT(name) ON Studio`.

Revoking Privileges

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

- Your grant 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 independently 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

- Nodes = user/privilege/grant option?/is owner?
  - UPDATE ON R, UPDATE(a) on R, and UPDATE(b) ON R live in different nodes.
  - SELECT ON R and SELECT ON R WITH GRANT OPTION live in 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* having privilege *P*.
  - *P* * = privilege *P* with grant option.
  - *P* ** = the source of the privilege *P(Owner)*.
    - 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* [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 there is a path from *XP* ** to *CQ*, *CQ* *, or *CQ* **, and *P* is a superprivilege of *Q*.
  - Remember that *P* could be *Q*, and *X* could be *C*.

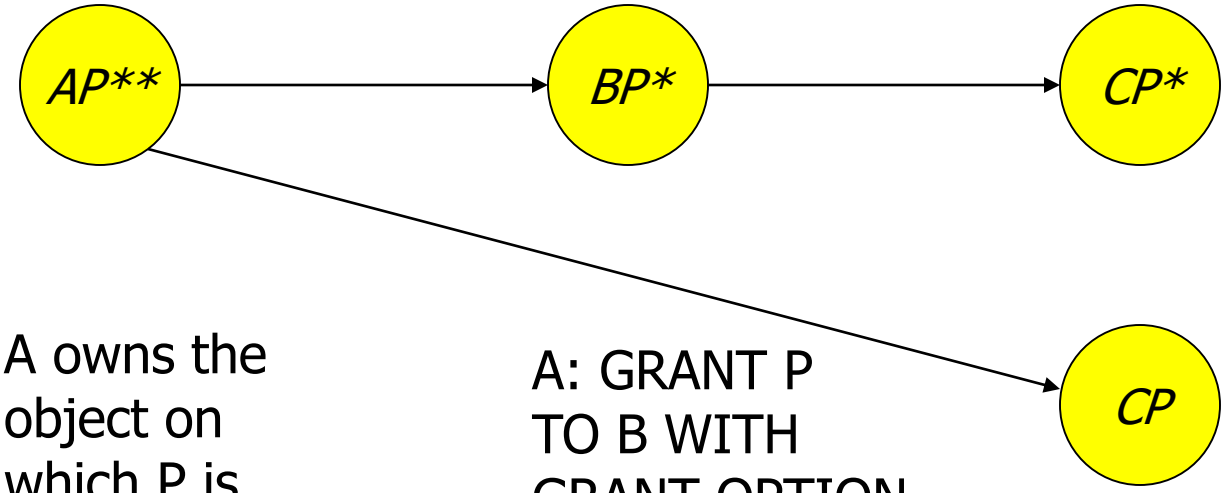
Manipulating Edges – (3)

- 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.
- Any node with no such path represents a revoked privilege and is deleted from the diagram.

Example: Grant Diagram



B: GRANT P
TO C WITH
GRANT OPTION

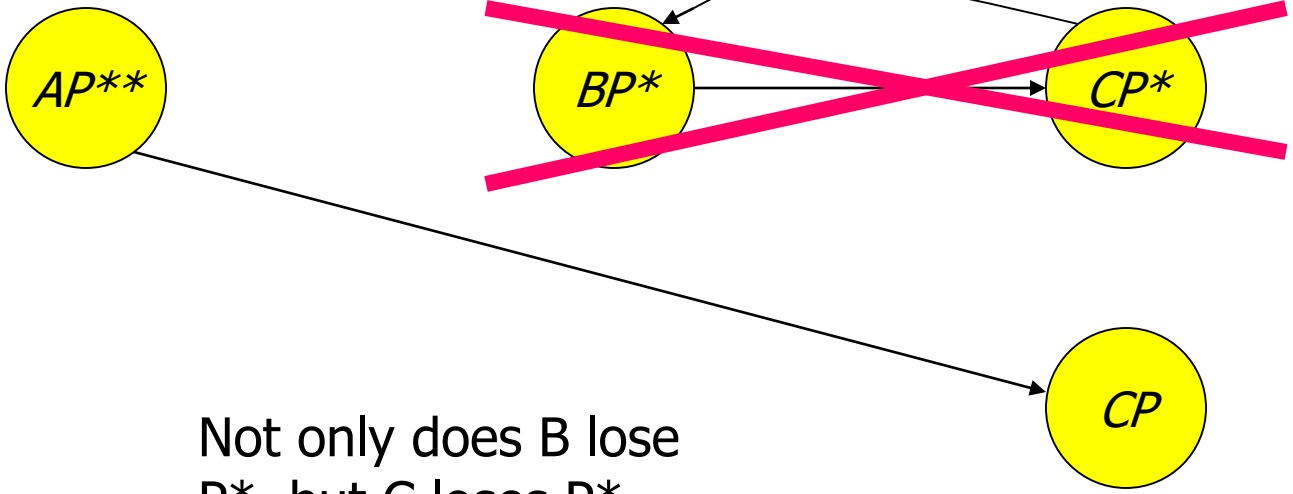
A owns the
object on
which P is
a privilege

A: GRANT P
TO B WITH
GRANT OPTION

A: GRANT P
TO C

Example: Grant Diagram

A executes
REVOKE P FROM B CASCADE;



Even had C passed P to B, both nodes are still cut off.

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.

Another example

Suppose: User janeway is the Owner of the MovieSchema which contains tables:

Movies(title, year, length, genre, studioName, producerC#)

Studio(name, address, presC#)

The statements exectued by janeway:

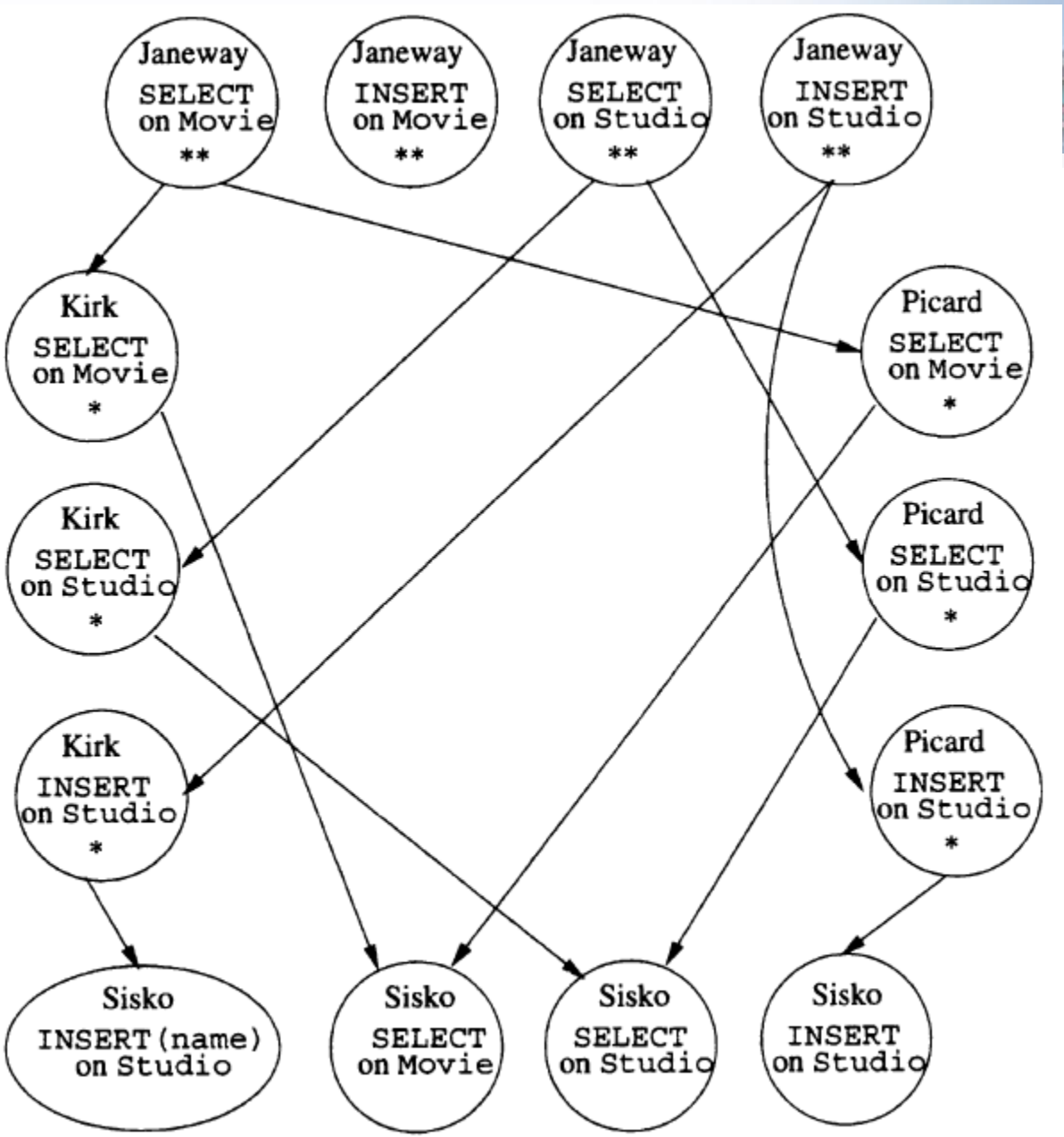
```
GRANT SELECT,INSERT on Studio TO kirk, picard
        WITH GRANT OPTION;
GRANT SELECT on Movies TO kirk, picard
        WITH GRANT OPTION;
```

Then picard grants to user sisko:

```
GRANT SELECT,INSERT on Studio TO sisko;
GRANT SELECT on Movies TO sisko;
```

Kirk also grants to user sisko:

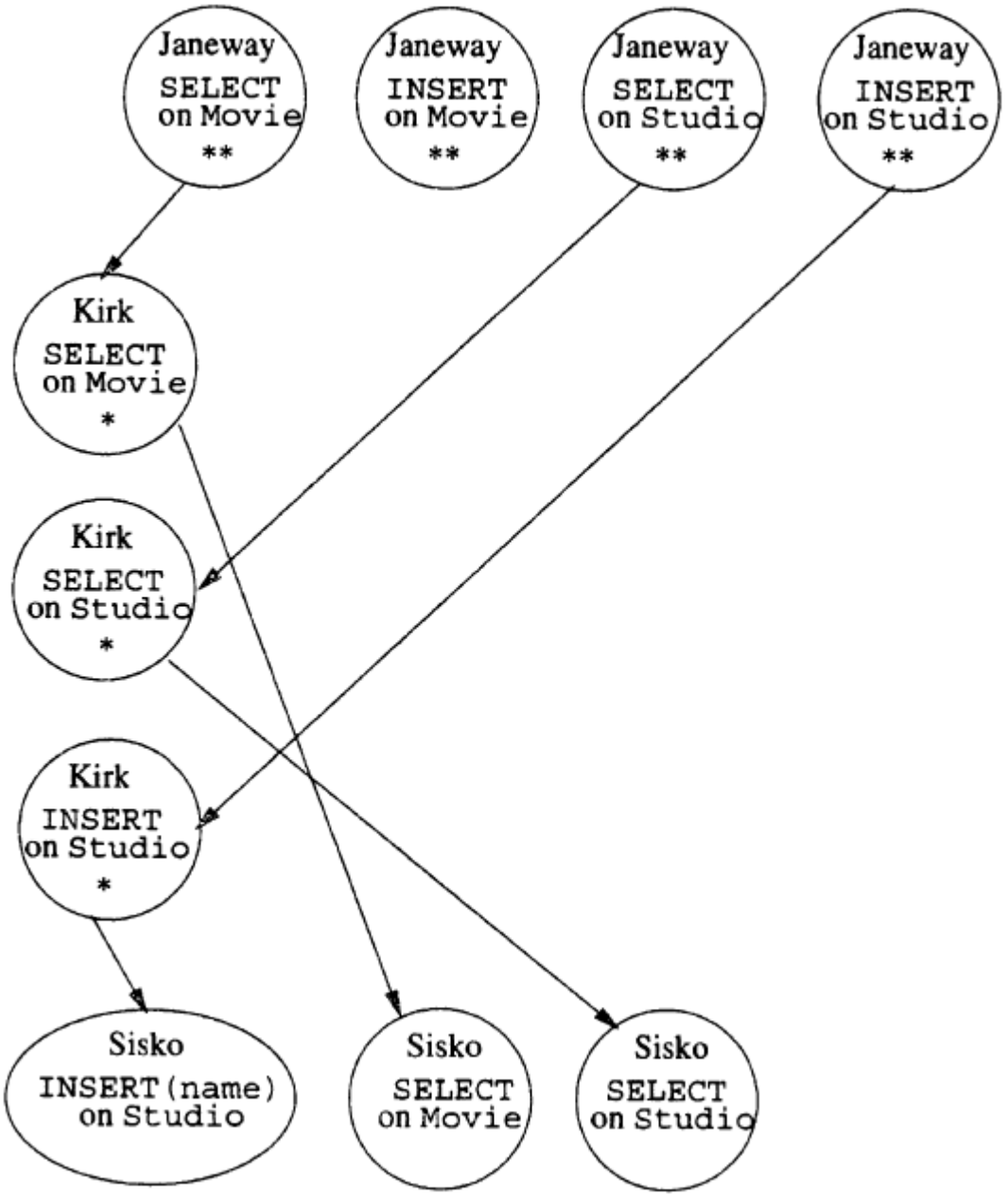
```
GRANT SELECT,INSERT(name) on Studio TO sisko;
GRANT SELECT on Movies TO sisko;
```


Janeway
SELECT
on Movie
**
Janeway
INSERT
on Movie
**
Janeway
SELECT
on Studio
**
Janeway
INSERT
on Studio
**
Kirk
SELECT
on Movie
*
Kirk
SELECT
on Studio
*
Kirk
INSERT
on Studio
*
Picard
SELECT
on Movie
*
Picard
SELECT
on Studio
*
Picard
INSERT
on Studio
*
Sisko
INSERT (name)
on Studio
Sisko
SELECT
on Movie
Sisko
SELECT
on Studio
Sisko
INSERT
on Studio

Suppose: User janeway do:

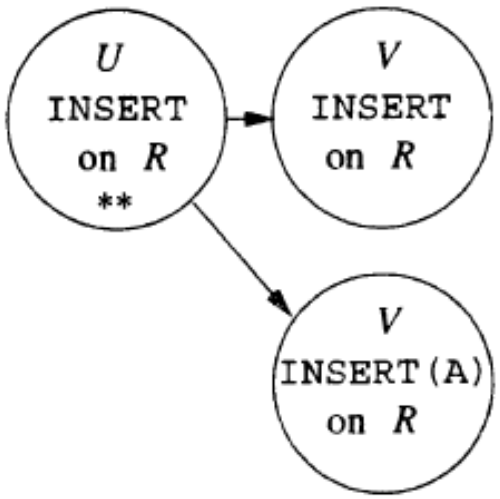
REVOKE SELECT,
INSERT on Studio
FROM picard CASCADE

REVOKE SELECT
on Movies
FROM picard CASCADE

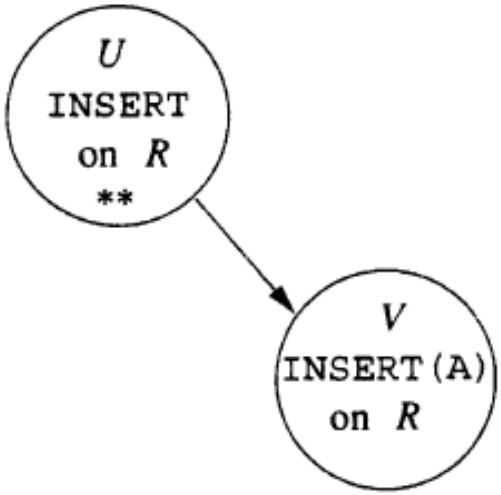


Example

Step	By	Action
1	U	GRANT INSERT ON R TO V
2	U	GRANT INSERT(A) ON R TO V
3	U	REVOKE INSERT ON R FROM V RESTRICT



(a) After step (2)

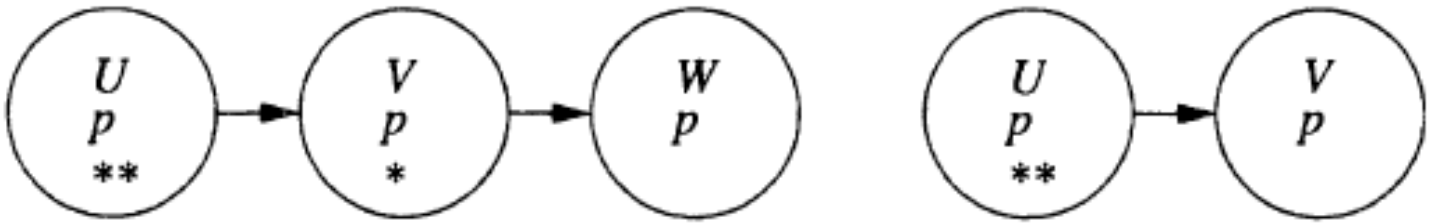


(b) After step (3)

Example

How about this?

Step	By	Action
1	U	GRANT p TO V WITH GRANT OPTION
2	V	GRANT p TO W
3	U	REVOKE GRANT OPTION FOR p FROM V CASCADE



(a) After step (2)

(b) After step (3)

…The End of This Lecture…

Q&A