

# 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** = right to query the relation.
  2. **INSERT** = right to insert tuples.
    - ◆ May apply to only one attribute.
  3. **DELETE** = right to delete tuples.
  4. **UPDATE** = right to update tuples.
    - ◆ May apply to only one attribute.

# Example: Privileges

- ◆ For the statement below:

```
INSERT INTO Lemonades(name)  
  SELECT lemonade FROM Sells
```

```
WHERE NOT EXISTS  
  (SELECT * FROM Lemonades  
   WHERE name = lemonade);
```

lemonades that do not appear in Lemonades. We add them to Lemonades with a NULL manufacturer.

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

# 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`.

# 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.



# Granting Privileges

- ◆ You have all possible privileges on the objects, such as relations, that you create.
- ◆ 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 Sells.  
You may say:

```
GRANT SELECT, UPDATE(price)  
ON Sells  
TO sally;
```

- ◆ Now Sally has the right to issue any query on Sells and 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 can grant to others the privilege UPDATE ON Sells.
  - ◆ Also, she can grant more specific privileges like UPDATE(price) ON Sells.

# 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  $X \rightarrow Y$  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  $P$ .
    - 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  $C$ .

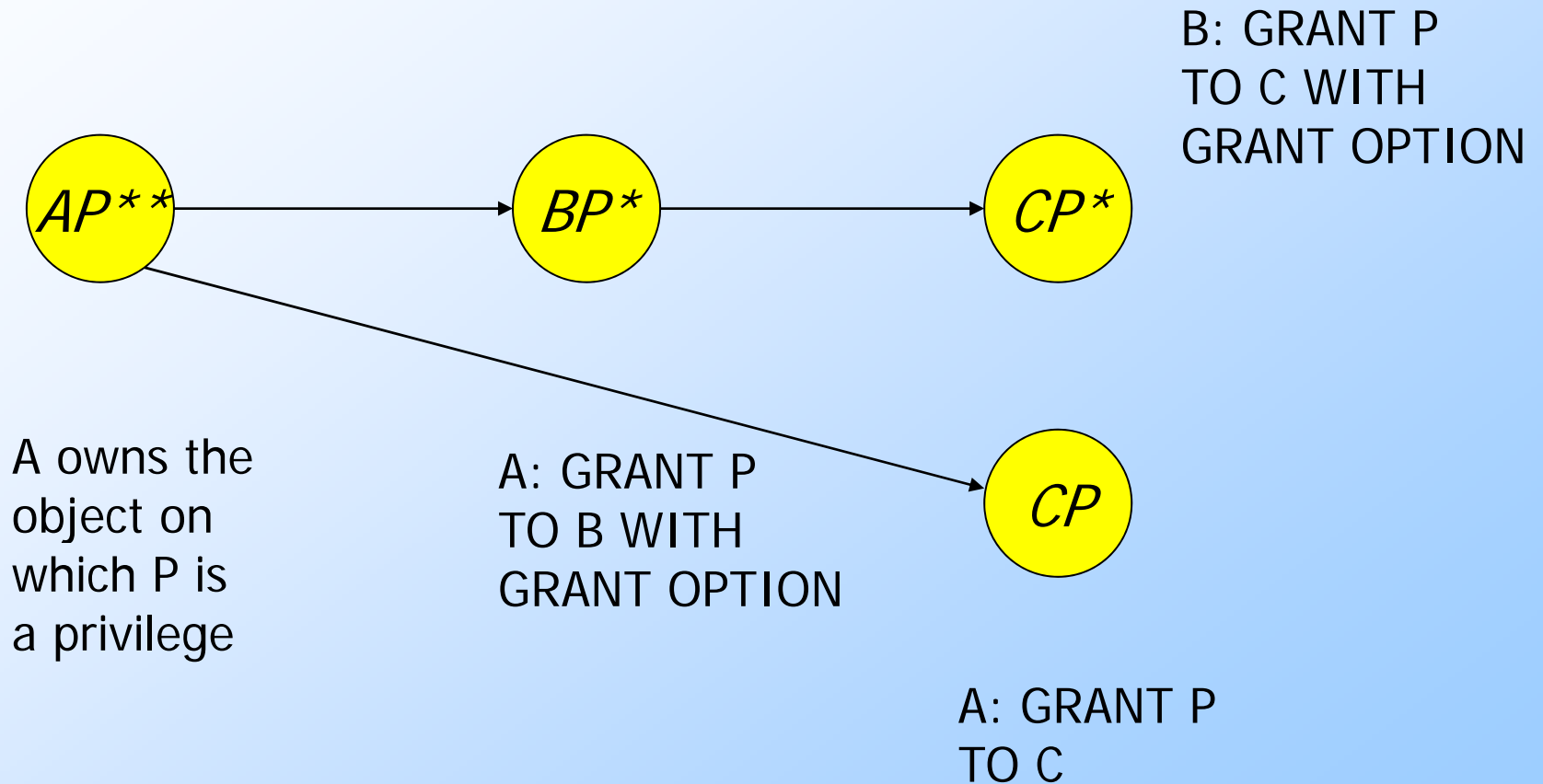
## Manipulating Edges – (3)

- ◆ 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)

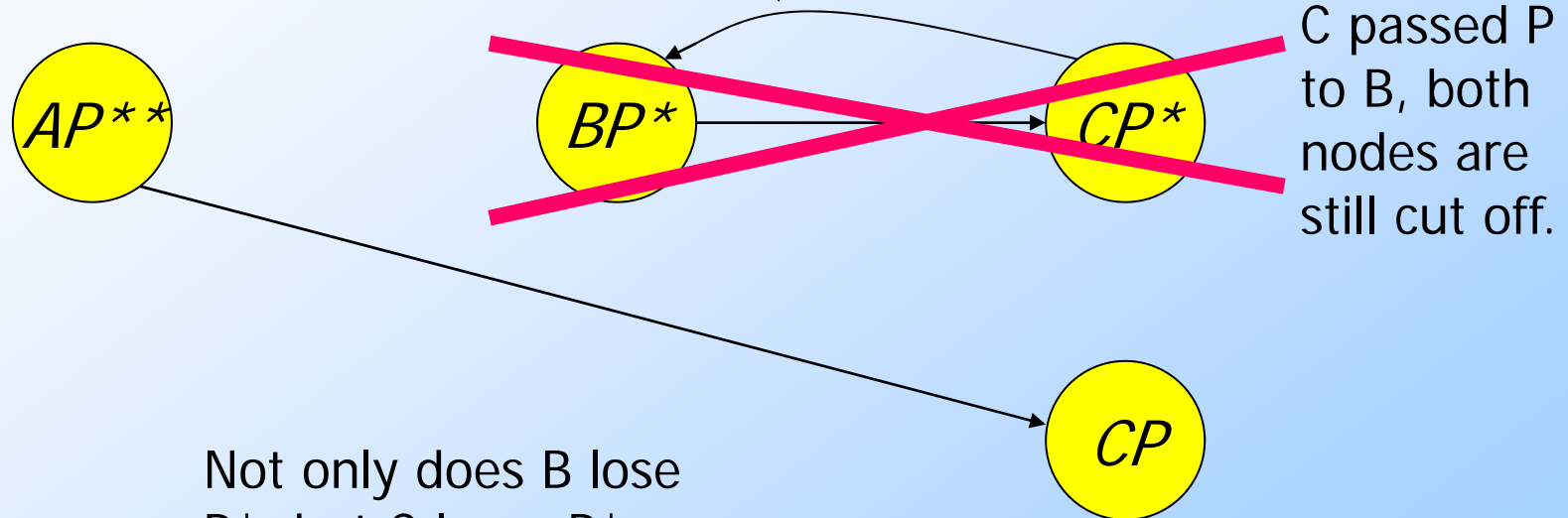
- ◆ Having revised the edges, we must check that each node has a path from some \*\* node, representing ownership.
- ◆ Any node with no such path represents a revoked privilege and is deleted from the diagram.

# Example: Grant Diagram



# 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.

# MySQL

```
CREATE USER user_specification  
[, user_specification] ...
```

user\_specification:

```
user [IDENTIFIED BY [PASSWORD]  
'password']
```

```
CREATE USER 'ali'@'localhost' IDENTIFIED BY 'xxx';
```

# MySQL

## GRANT

```
priv_type [(column_list)]
[, priv_type [(column_list)]] ...
ON [object_type] priv_level
TO user_specification [, user_specification] ...
[REQUIRE {NONE | ssl_option [[AND] ssl_option] ...}]
[WITH with_option ...]
```

## object\_type:

```
TABLE
| FUNCTION
| PROCEDURE
```

## priv\_level:

```
*
| *.*
| db_name.*
| db_name.tbl_name
| tbl_name
| db_name.routine_name
```

## user\_specification:

```
user [IDENTIFIED BY [PASSWORD] 'password']
```

```
GRANT SELECT ON deneme.* TO 'ali'@'localhost';
```



# MySQL

## REVOKE

```
priv_type [(column_list)]  
[, priv_type [(column_list)]] ...  
ON [object_type] priv_level  
FROM user [, user] ...
```

## REVOKE ALL PRIVILEGES, GRANT OPTION

```
FROM user [, user] ...
```

```
REVOKE SELECT ON deneme.* FROM 'ali'@'localhost';
```