Granting Privileges

- The owner has all priv. on the objects he/she creates.
- Owner can pass privileges to others

GRANT <list privileges> ON <object>
TO <userlist> [WITH GRANT OPTION];

Grant option

A user has grant option on a given priv. if he/she is the owner of the object or has received grant option on the given privilege.

Subprivileges

A user can pass more restricted priv. to other users (including with grant option)

Ex: User A has SELECT on R(a,b) with grant option.
User A can pass SELECT on R(a) to user B

A executes: GRANT SELECT(a) ON R TO B;

Example 2:

User A created table R(a,b)

A executes:

```
GRANT SELECT, UPDATE(a) ON R
  TO B WITH GRANT OPTION;
```

B executes

```
GRANT SELECT(a), UPDATE(a) ON R
 TO PUBLIC;
```

↑ Special authentication ID

⇒ it means every user

Revoking privileges

```
REVOKE <list privileges> ON <object>
  FROM <userlist> { CASCADE
                  | RESTRICT
```

When a user revokes privileges from a user the DBMS must verify if other users depend on that grant.

Ex: See example 2 above.

A granted to B SELECT

B granted to PUBLIC SELECT(a)

if A revokes SELECT from B, should PUBLIC lose it?

CASCADE: Removes privilege from user and any other users that depend on it

RESTRICT: If other users depend on the grant being removed, the revoke fails.

User A runs:

REVOKE SELECT ON R FROM B CASCADE;

Both B & PUBLIC lose privilege.

REVOKE SELECT ON R FROM B RESTRICT;

Revoke fails because PUBLIC priv. on R depends on B.

But what if PUBLIC also got the privilege from another user?

- it becomes complicated.

Grant Diagrams

Formalism to model grants and revokes and to determine, at runtime if a user has a given privilege.

Each node is a tuple:

⟨user, priv, grant option?, is owner?⟩

Edges (directed)

⟨X, Y⟩

⇒ implies that grant described in Y was granted by X

If successful a GRANT statement creates an edge (if it does not already exist).

If successful a REVOKE removes one or more edges.

Revoking only grant option

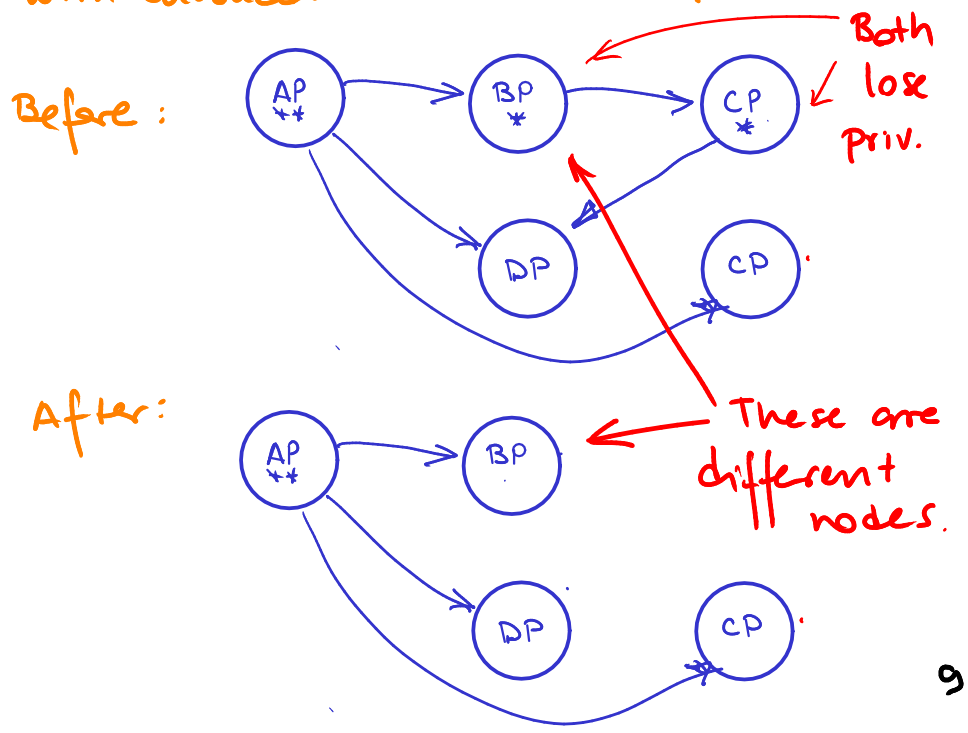
```
REVOKE GRANT OPTION FOR <priv>
  FROM <user> { CASCADE
              { RESTRICT
```

If user has privilege with grant option it is equivalent to:

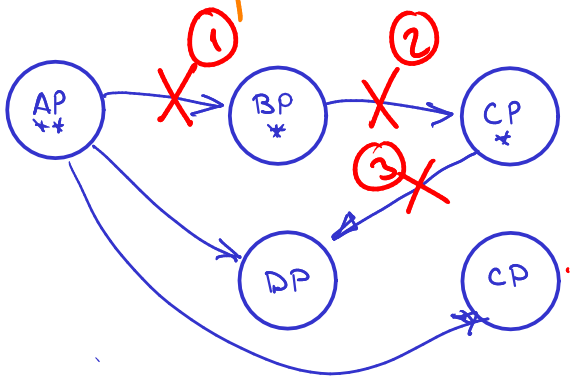
REVOKE privilege

GRANT privilege without grant.

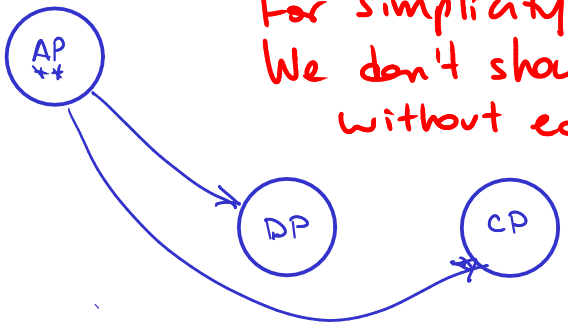
Now A revokes grant option from B with cascade:



Now A revokes P from B with CASCADE.



Resulting graph:



For simplicity
We don't show nodes without edges

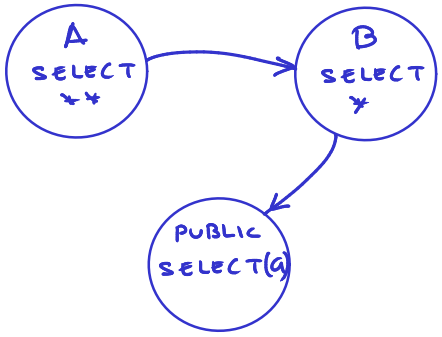
Users D and C keep P without grant.

For the sake of simplicity we only draw nodes involved in a sequence of grants.

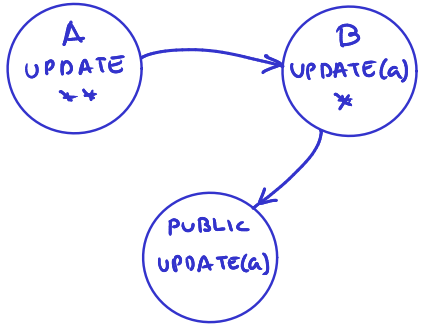
- A node with ** means is owner (implies GRANT option)
- A node with * means GRANT option.

SELECT

- One graph per super-privilege
- includes its subprivileges
e.g. SELECT(a)



UPDATE



Fundamental Rule:

User C has privilege Q as long as there exists a path from XP** to either CQ, CQ*, or CQ** and P is a super-privilege of Q.

Example:

Does PUBLIC have SELECT on R?

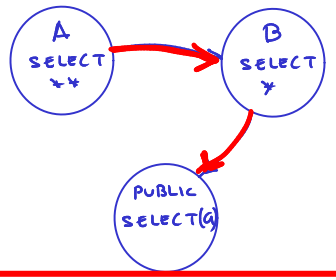
Node 

(not shown) above is unreachable from 

⇒ No, PUBLIC does not have SELECT.

Does PUBLIC have SELECT(a) R?

Yes, there is a path from A/select/** to PUBLIC/select.



Important: Cycles are not allowed in SQL grants.
eg. A grants to B, B grants to A

Revoking Privileges

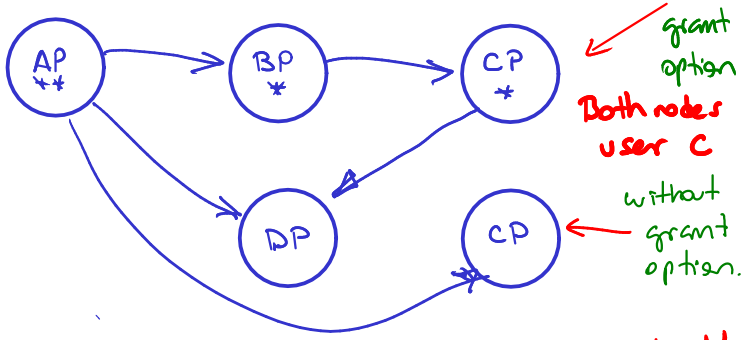
if A revokes P from B

RESTRICT: delete edge from AP to BP only if there are no edges from BP.
otherwise fail.

CASCADE:

- delete edge from AP to BP
- check, for every node X that has an incoming edge that there exists a path from OP** (the owner of the relation) to X. If no path exists,
- delete the incoming edges to X.
- delete any outgoing edges from X.

Ex:



Order of grants does not matter!!