# 11 Access Rights in SQL

11.1 The SQL security model

11.2 Granting and revoking privileges

Kemper / Eickler: chap. 12; Elmasri: chap. 22,
Melton: chap. 14

## 11.1.1 Introduction to terminology

- Privacy:
  - Users should not be able to see and use data they are not supposed to.

  e.g., A student can't see other students' grades.
- Security:
  - No one should be able to enter the system and / or impact its behavior without being authorized to do so.

  e.g. delete or change data without being authorized
- Integrity:
  - Authorized users should not be able to modify things they are not supposed to, (e.g. in a way which affects contraints)

  e.g., Only instructors can assign grades.
- Availability:
  - Users should be able to see and modify things they are allowed to – e.g. the DB should always be operational

## ... Terminology

- Authorization: give rights to individuals
- Authentication: is user the one she pretends to be?
  - Passwords
  - encryption
  - organizational support
- Auditing
  - logging all kinds of events



## Discretionary Access Control

- Creator responsible for ACC
  $\Rightarrow$ explicit grant of rights on objects to individuals

Security model

{ (o, s, a, p, b) | access control rules }

where:

o $\in$ O Objects like tables, stored procedures, ...

s $\in$ S Subjects: individual users, application progs, ..

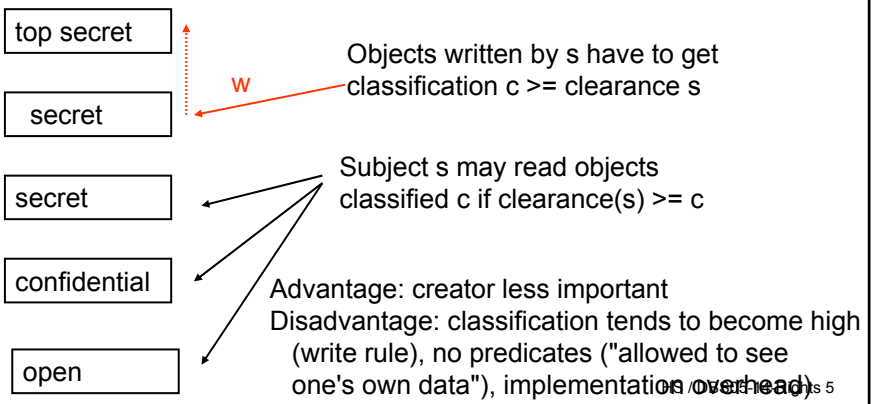
a $\in$ A Actions: read (SELECT), write, execute

p : predicate which qualifies the objects of this rule

b: allowed to pass on rights to other individuals

## Mandatory Access Control

- Objects are classified according to classification hierarchy
- subjects have trustability t (clearance)






## SQL

- SQL Model DAC disadvantage??
- SQL implementation

  – Access matrix for rules

    e.g. Table_privileges (grantor, grantee, table, privilege, grantable)

  - Views (⇐ predicate)
  - Query modification

- Auditing

## SQL security related terminology

- User
  - Not the schema object, just a name for a session of an individual user
  - Identification by Authorization ID (user name)
- Role
  - Name for a role, to which rights may be assigned
  - may be granted to users / applications
    kind of convenience functionality (not in Postgres)
- Privileges (Rights)
  - System privileges
  - Object (data) privileges: creator has all privileges
- Operations
  - GRANT <privilege>
  - REVOKE <privilege>
- Policy

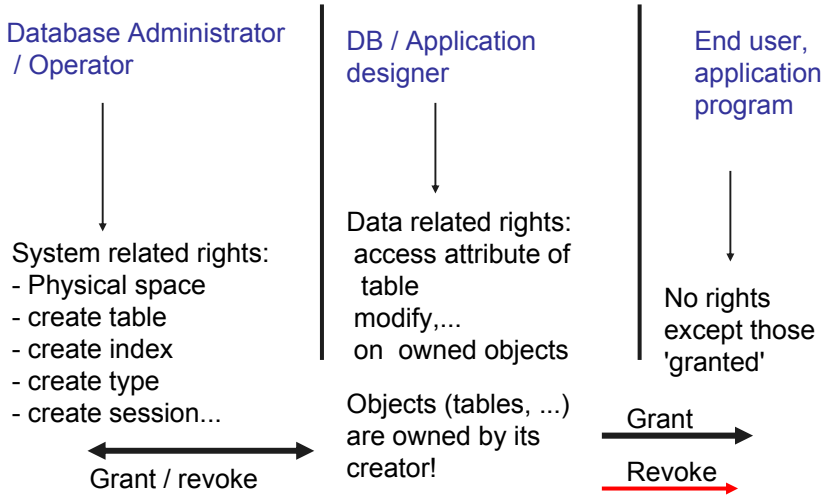


## Example: Requirements for access protection

- Company
  - Personal, accounting, inventory, orders, clients, …
- Requirement (1) :
  system staff should not be able to read the data
- Requirement (2) :
  rights should be assignable in a task specific way
  and rights should be as restrictive as possible.
  Order entry clerks (or programs!) should
  - not be able to read any personal data
  - be able to insert new orders, but not modify or delete
  - read client data
- Requirement (3):
  rights should be assigned to functions (roles) rather than to people or programs
- Requirement (4)
  rights should be revocable

## Access Control policy

- Policy specifies rights users should (not) have






## User Access Control: Privileges

- Privileges
  - Right to perform SQL statement type on objects
  - Assigned to users or roles (authorization IDs)
  - Creator of object: all privileges for that object
  - Administrator: management of system privileges
- Object Privilege types:
  - SELECT [<column list>] on table or view -- most systems: no column list
  - INSERT [<column list>] on table or view
  - DELETE on table or view
  - UPDATE [<column list>] on table or view
  - REFERENCES [<column list>] : right to refer to relation in constraint
  - EXECUTE
  - ALL PRIVILEGES: short form for all privileges

## User Access Control: Privileges, example

SQL operations:

frequently more than one privilege needed

- Example

```
INSERT INTO Format (format)
       SELECT t.format
       FROM Tape t
       WHERE t.format NOT IN (SELECT format
                              FROM Format);
```

- Privileges needed:
  - SELECT on Tape
  - SELECT on Format
  - INSERT on Format

Compare Views V:
when using V, no privileges needed for the defining views or tables



## Roles and users

ROLES

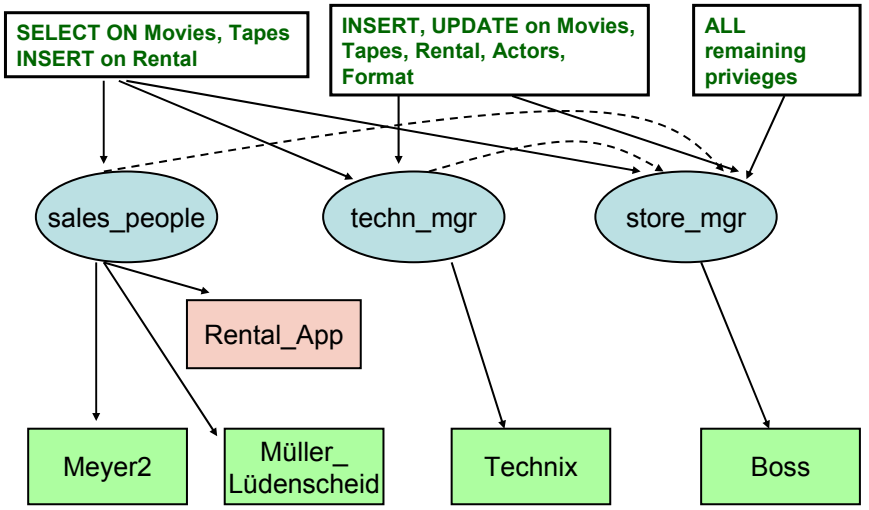
- define a set of privileges for a (potentially) large set of users

```
CREATE ROLE sales_people;
-- grant some privileges to sales_people
-- grant sales_people role to users
```

- much more economic than direct privileges
  (no security by obscurity, … hopefully)
- roles may be assigned to roles
- often assigned to applications instead of individual users

```
set role <rolename>
```

## Privileges, Roles, users / applications (expl)



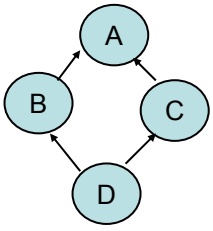



## Implicit Access Control

Partial order on Roles, operations and types

Roles:

- sets of privileges, no relationship between different roles
- difficult to manage
- role hierarchy

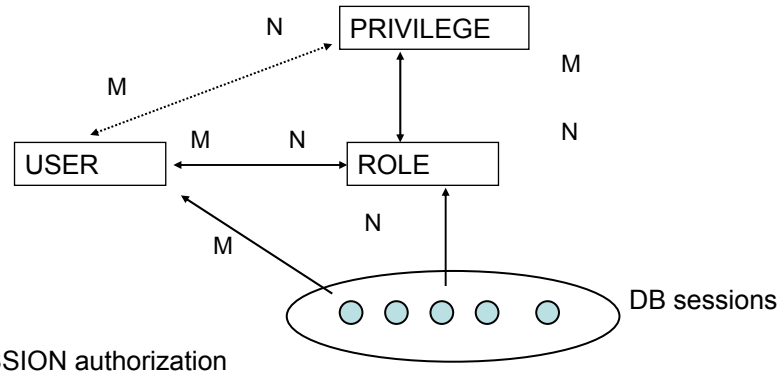



B and C have the privileges of D (implicitly) and different explicit ones

A implicitly has B and C privileges

Example above:
Store_mgr has at least rights of sales_people and tech_mgr

# Session, users, functions, roles

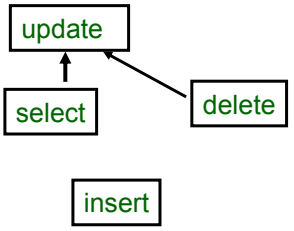



SESSION authorization
may be temporarily echanged by
SET SESSION AUTHORIZATION user | role statement,
e.g. in a PL/SQL function

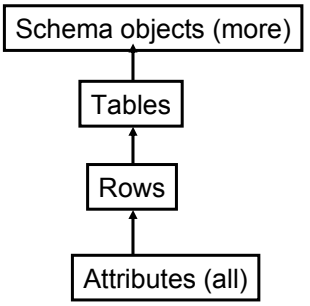


# Implicit access control

Operations




Types




Assigment of
- operations and types to roles
- roles to individual users (applications)

makes privileges more transparent…

…hopefully

## 11.2 Granting and revoking privileges

- GRANT

```
GRANT <privileges> ON <object>
                   TO [<users>|<role>]
                   [WITH GRANT OPTION]
```

  - GRANT OPTION: Right to pass privilege on to other users
  - Only owner can execute CREATE, ALTER, and DROP

Privilege to INSERTparticular columns in a table

```
GRANT INSERT
        ON <tablename(<attributenames>)>
        TO <users> [WITH GRANT OPTION]
```

Access matrix : <user> has <right> on <object>



## User level Access Control: Privileges

- Examples:

```
GRANT INSERT, SELECT ON  Movie  TO Klaus
```

Klaus can query Movie or insert tuples into it.

```
GRANT DELETE ON Movie TO shop_owner WITH GRANT OPTION
```

Anna can delete tuples, and also authorize others to do so

```
GRANT UPDATE (price_Day) ON Movie TO movie_staff
```

Staff can update (only) the price field of Movie tuples

```
GRANT SELECT ON MovieView  TO Customers
```

This does NOT allow the customers to query Movie directly!

## User Access Control: Privileges on views

- Creator has privilege on view if privilege on all underlying tables
- Creator loses SELECT privilege on underlying table ⇒ view is dropped
- Creator loses a privilege on underlying table ⇒ creator loses privilege on view
- Creator loses a privilege held with grant option on underlying table ⇒ users who were granted that privilege on the view lose privilege on view



## User Access Control: Revoke privileges

- Revoke privilege

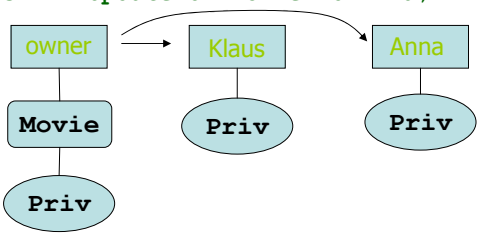
```
REVOKE <privileges>
       ON <object>
       FROM <users> {RESTRICT | CASCADE}
```

  - RESTRICT: only revoke if none of the privileges have been granted by *these* users
  - CASCADE: revoke from all users that have been granted the privilege by these users
  - Privilege given from different users – must be revoked from all users to loose privilege

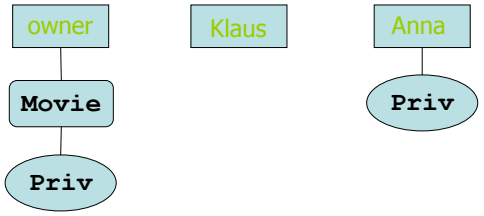
# User Access Control: Examples

Owner: `GRANT Update ON Movie TO Klaus;`

Owner: `GRANT Update ON Movie TO Anna;`




Owner: `REVOKE Update ON Movie FROM Klaus RESTRICT;`

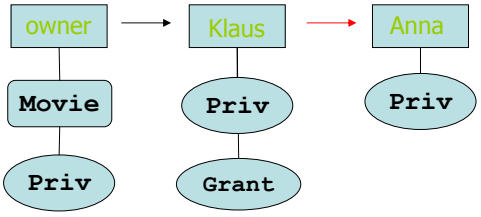




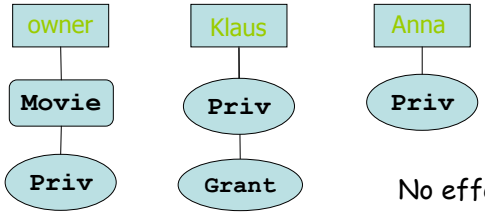

# User Access Control: Examples

Owner: `GRANT Update ON Movie TO Klaus WITH GRANT OPTION;`

Klaus: `GRANT Update ON Movie TO Anna;`




Owner: `REVOKE Update ON Movie FROM Klaus RESTRICT;`

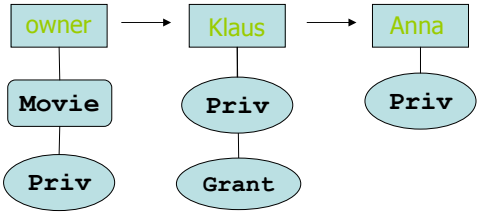



No effect

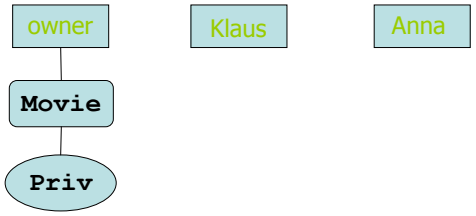
# User Access Control: Examples

Owner: `GRANT Update ON Movie TO Klaus WITH GRANT OPTION;`

Klaus: `GRANT Update ON Movie TO Anna;`




Owner: `REVOKE Update ON Movie FROM Klaus CASCADE;`

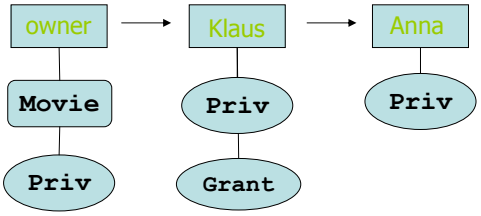




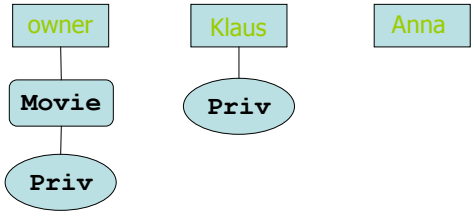

# User Access Control: Examples

Owner: `GRANT Update ON Movie TO Klaus WITH GRANT OPTION;`

Klaus: `GRANT Update ON Movie TO Anna;`




Owner: `REVOKE GRANT OPTION FOR Update ON Movie FROM Klaus CASCADE;`

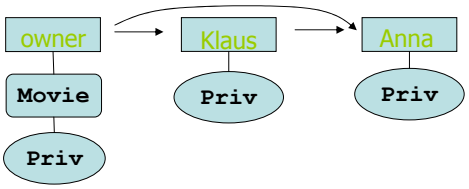


# User Access Control: Examples

Owner: `GRANT Update ON Movie TO Klaus WITH GRANT OPTION;`

Owner: `GRANT Update ON Movie TO Anna;`

Klaus: `GRANT Update ON Movie TO Anna;`




Owner: `REVOKE UPDATE ON Movie FROM Klaus CASCADE;`




# Access restrictions

- Views compared with explicit GRANT

  Advantage:

  access restriction on columns *and* rows specified by predicate

- Important aspect not discussed:

  how to make applications secure?

  e.g. Customer may see her own orders but not those of other customers

  Note: Customer is not an object of the DB!

  see e.g. "Private Virtual Databases (PVD)" in Oracle

## Summary

- Security of DB and DB applications extremely important
- Granting privileges already in SQL / SEQUEL
- Roles make privileges with many users managable
- Implicit privileges by role lattices (part. order) not used in SQL
- Views play an important role, but updatability?
- Assignment of privileges / roles to applications makes web based DB applications somehow secure
- Security in DB related applications: independent "reinvention" of security subsystems
- Fine granular access restriction on objects very important in DB context.