Discretionary Access Control

Discretionary Access Control

- Based on the concept of access rights or privileges for objects (tables and views), and mechanisms for giving users privileges (and revoking privileges).
- Creator of a table or a view automatically gets all privileges on it.
  - DMBS keeps track of who subsequently gains and loses privileges, and ensures that only requests from users who have the necessary privileges (at the time the request is issued) are allowed.

System vs. Object Privileges

- System Privileges
  - The right to access the database and its objects – create table, create view, back up any table, etc.
  - Generally granted by the DBA
- Object Privileges
  - The right to manipulate the content of the objects database – alter, delete, execute, index, insert, reference, select, update
  - Grant by the owner of the object

Control of User Access

- Oracle Server database security, you can do the following:
  - Control database access
  - Give access to specific objects in the database
  - Confirm given and received privileges within the Oracle data dictionary
  - Create synonyms for database objects

Privileges

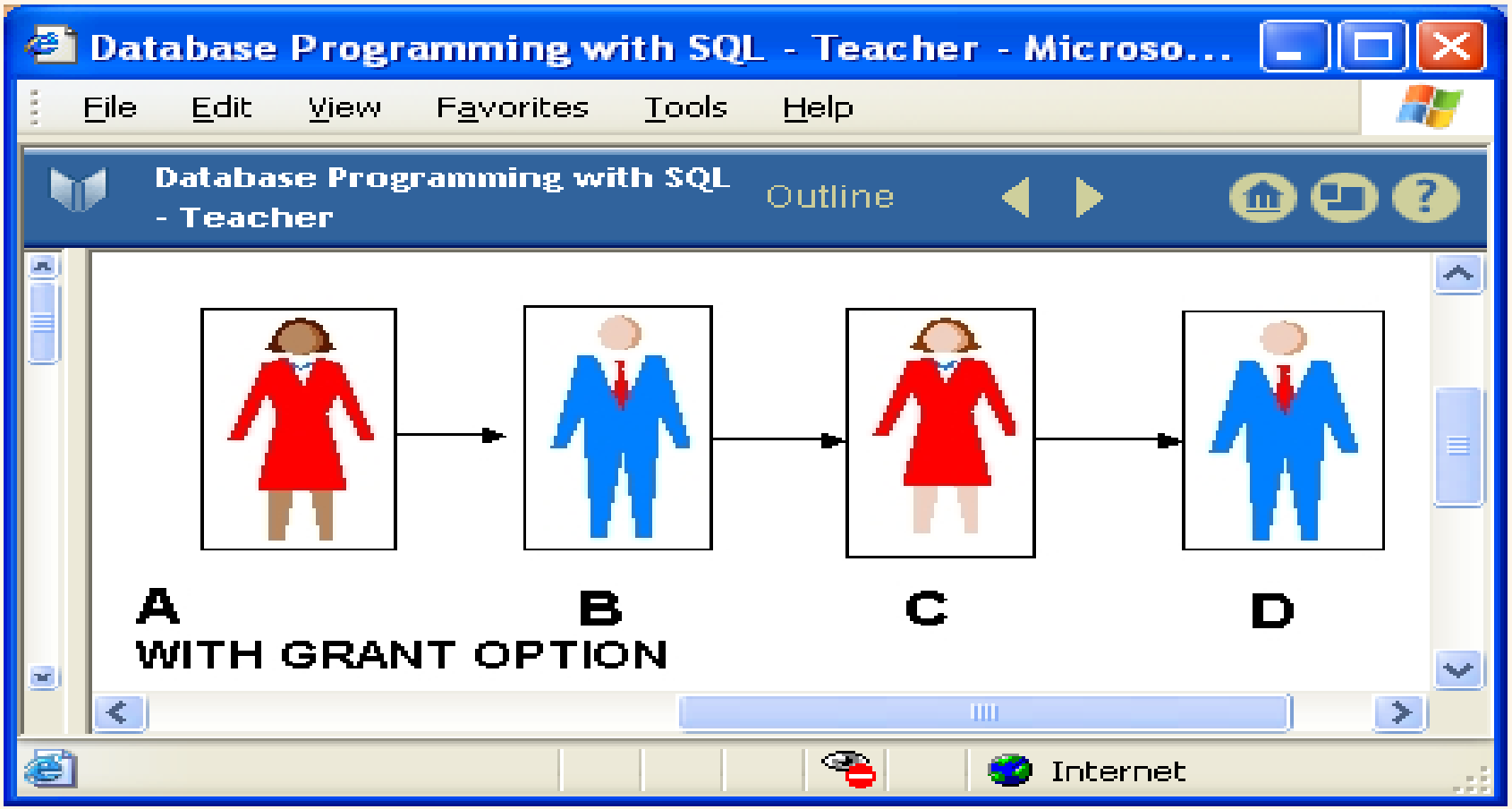
- Right to execute particular SQL statements.
- DBA – high-level user with ability to grant users access
- Users require system privileges to gain, access to databases/objects to manipulate content
- Users can be given privilege to grant additional privileges to other users/roles

Grant and Revoke: Definition

Grant:

The GRANT statement is used to give permissions to a user or role.
By using the GRANT statement, it is possible to assign permissions to both statements as well as objects.

You can use the GRANT statement with the WITH GRANT OPTION clause to permit the user or role receiving the permission to further grant/revoke access to other accounts


Database Programming with SQL - Teacher - Microso...
File Edit View Favorites Tools Help
Database Programming with SQL - Teacher
Outline
A
B
C
D
WITH GRANT OPTION
Internet

Grant and Revoke: Definition

Revoke:

The REVOKE statement is used to remove a previously granted or denied permission from a user in the current database.

You can specify the GRANT OPTION FOR clause with the REVOKE statement to remove the WITH GRANT OPTION permissions.

Specify the CASCADE clause along with the WITH GRANT OPTION clause, if the permissions being revoked were originally granted using the WITH GRANT OPTION setting.

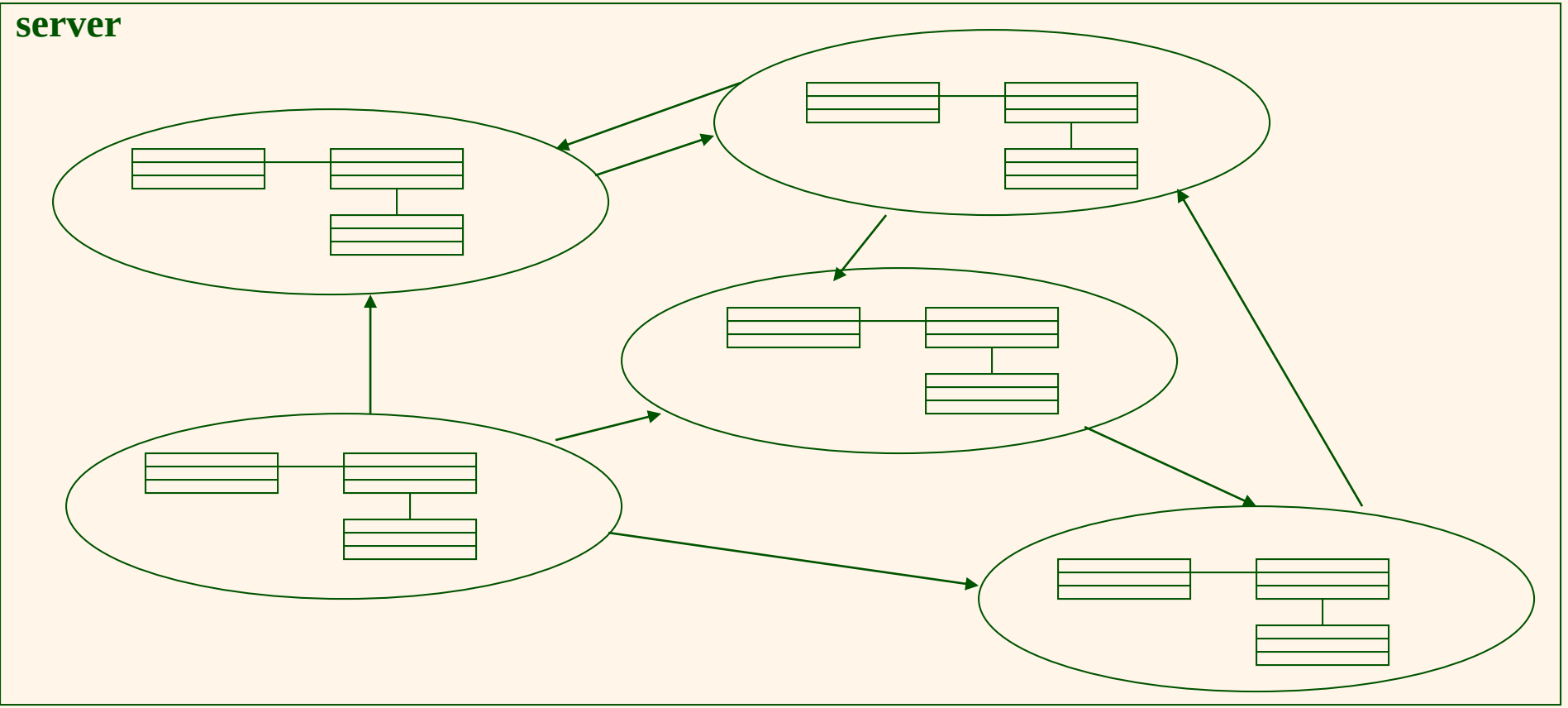
Rights Levels

- Users or Grant and Revoke rights are created at 4 levels
  - Global level
  - Database level
  - Table level
  - Column level

Global level

Global level

Global privileges apply to all databases on a given server.



Database level

Database level

Database privileges apply to all tables in a given database.

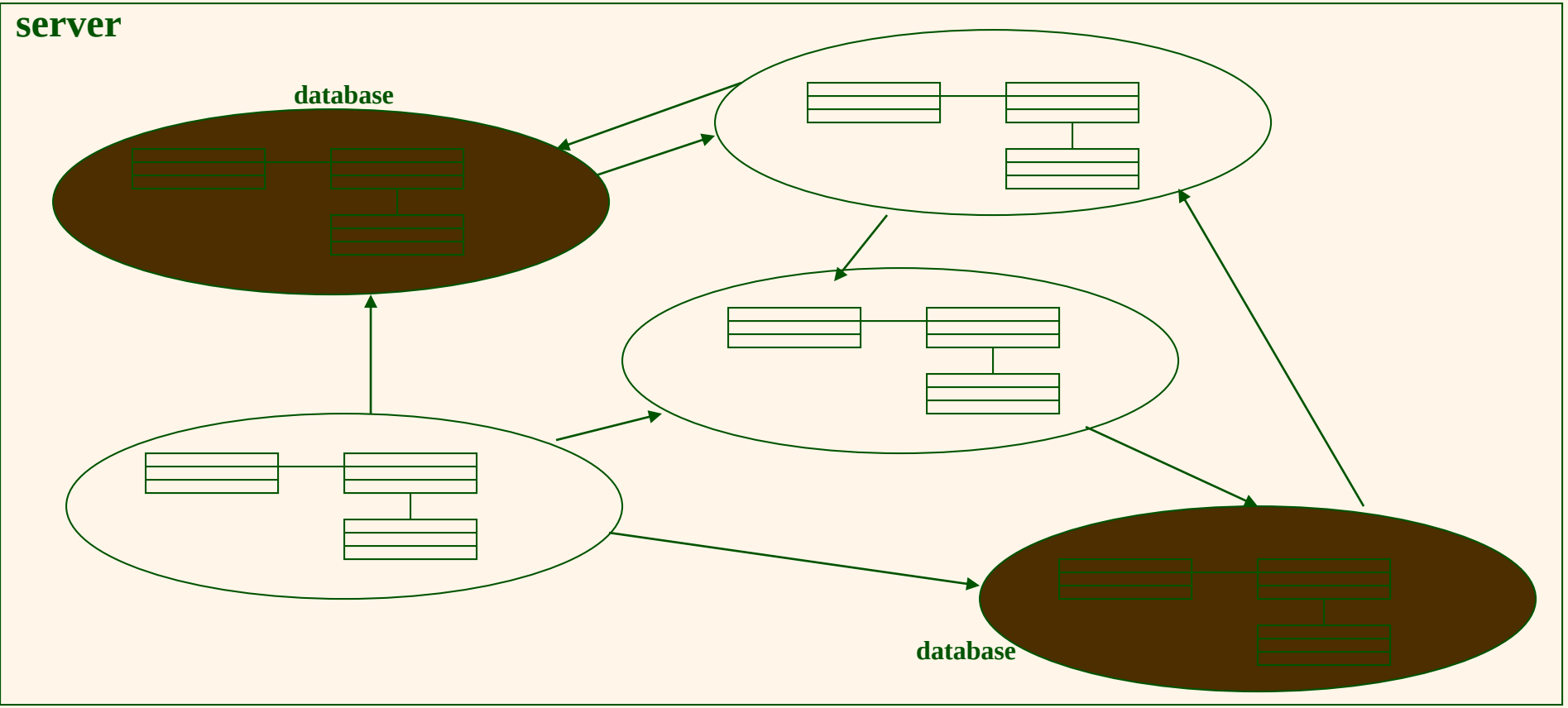
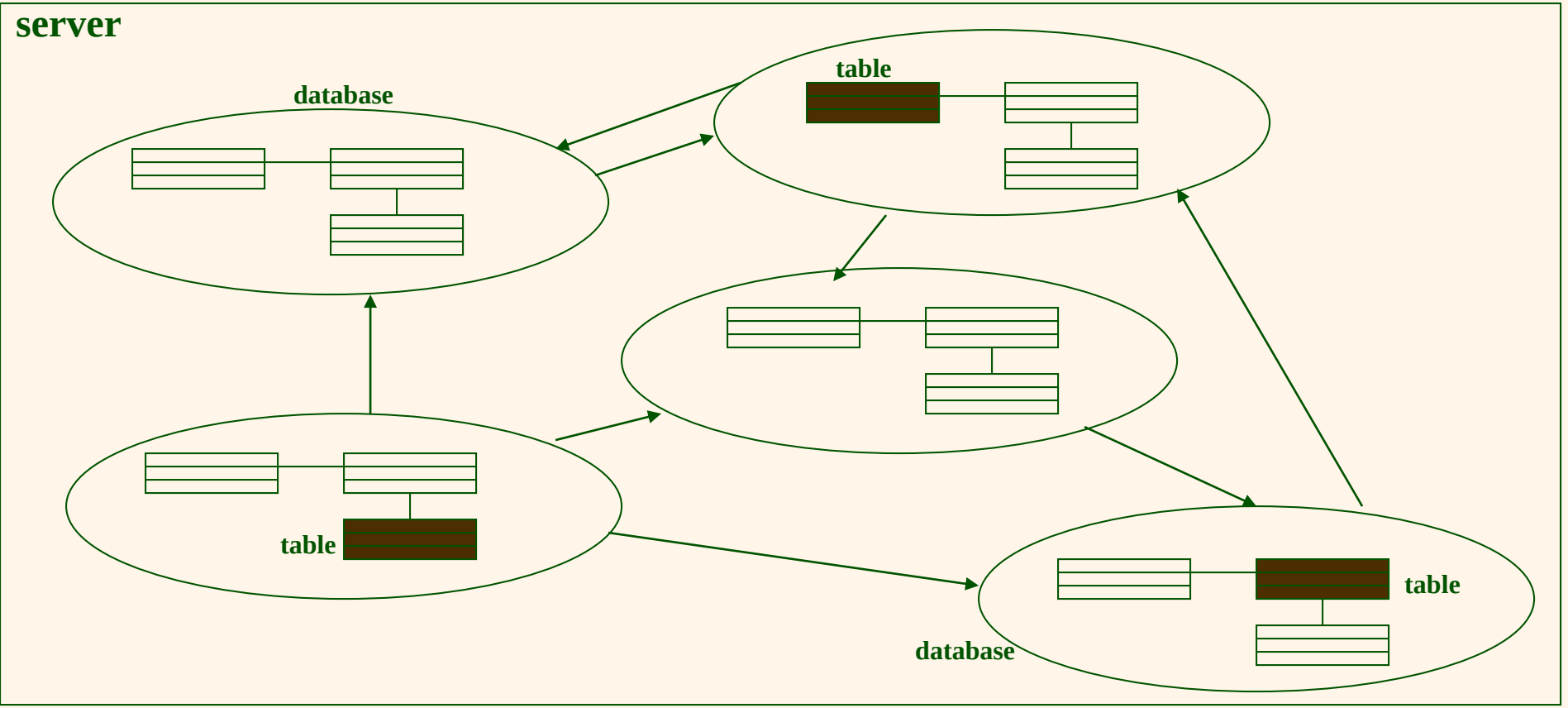


Table level

Table level

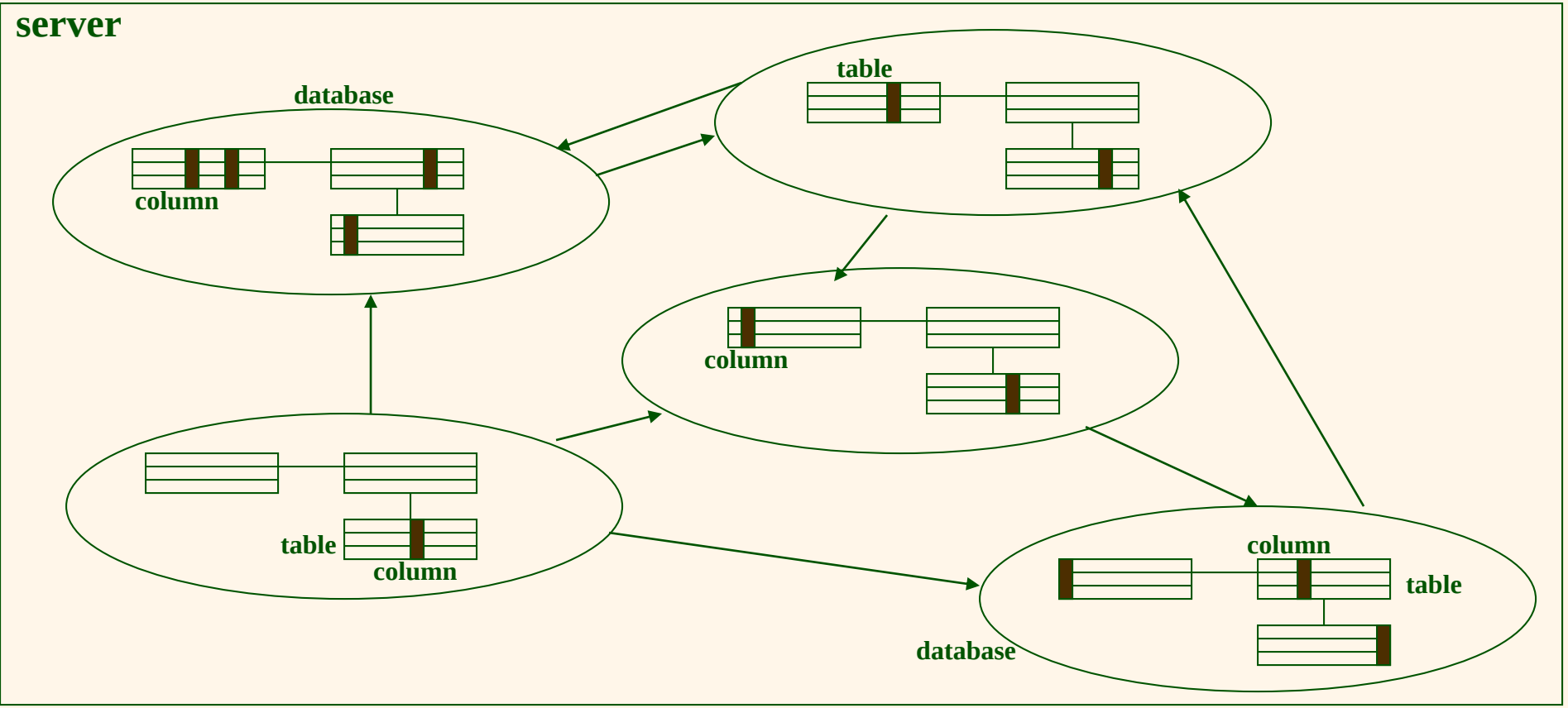
Table privileges apply to all columns in a given table.



Column level

Column level

Column privileges apply to single columns in a given table.



Syntax & Sémantic

→ MySQL

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)] ...]
    ON {tbl_name | * | *.* | db_name.*}
    TO user_name [IDENTIFIED BY 'password']
        [, user_name [IDENTIFIED BY 'password'] ...]
    [WITH GRANT OPTION]

REVOKE priv_type [(column_list)] [, priv_type [(column_list)] ...]
    ON {tbl_name | * | *.* | db_name.*}
    FROM user_name [, user_name ...]
```

Syntax & Sémantic

⟹ **Postgres**

GRANT { { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER } [,...] | ALL [PRIVILEGES] }
ON [TABLE] *tablename* [, ...]
TO { *username* | GROUP *groupname* | PUBLIC }
[, ...]

REVOKE { { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER } [,...] |
ALL [PRIVILEGES] }
ON [TABLE] *tablename* [, ...]
FROM { *username* | GROUP *groupname* | PUBLIC } [, ...]

Syntax & Sémantic

Oracle

```
GRANT SELECT, INSERT, UPDATE,
DELETE ON DEPARTMENT
TO username;
REVOKE SELECT
ON   employee
FROM   BOB;
```

Syntax & Sémantic

Microsoft SQL

GRANT SELECT, ...
ON table
TO username

REVOKE SELECT, ...
ON table
FROM username

GRANT Command

GRANT privileges ON object TO users [WITH GRANT OPTION]

- The following privileges can be specified:
  - SELECT: Can read all columns (including those added later via ALTER TABLE command).
  - INSERT(col-name): Can insert tuples with non-null or non-default values in this column.
    - INSERT means same right with respect to all columns.
  - DELETE: Can delete tuples.
  - REFERENCES (col-name): Can define foreign keys (in other tables) that refer to this column.
- If a user has a privilege with the GRANT OPTION, can pass privilege on to other users (with or without passing on the GRANT OPTION).
- Only owner can execute CREATE, ALTER, and DROP.

GRANT and REVOKE of Privileges

- GRANT INSERT, SELECT ON Sailors TO Horatio
  - Horatio can query Sailors or insert tuples into it.
- GRANT DELETE ON Sailors TO Yuppy WITH GRANT OPTION
  - Yuppy can delete tuples, and also authorize others to do so.
- GRANT UPDATE (*rating*) ON Sailors TO Dustin
  - Dustin can update (only) the *rating* field of Sailors tuples.
- GRANT SELECT ON ActiveSailors TO Guppy, Yuppy
  - This does NOT allow the 'uppies to query Sailors directly!
- REVOKE: When a privilege is revoked from X, it is also revoked from all users who got it *solely* from X.

Some examples in DB2

Authority R_1

Grant retrieve (S#, Sname, city), Delete

To John

ON Attempted Violation Reject

Or Lock_KB, etc

Authority R1

Grant insert , Delete , Retrieve (S # , SName , city)

ON S where S.city ≠ 'london'

To Jim, Fred

ON attempted violation Reject

Value dependent Authority rules

Authority R2

Grant Retrieve, Update (Sname, City)

ON S where S.city ≠ 'Paris'

To John;

Value dependent

Authority R3

Grant Delete, Retrieve (Sname, S#)

ON S

Value independent

Context dependent Authority rules

Grant Retrieve, Update (Sname, City)

ON S where Day ( ) IN ('SUN' , 'Mon' , 'Tue')

AND Time ( ) > Time '9: 00 ' AM

AND Time ( ) < Time '5: 00'PM AND Terminal ( ) = 'T4'

To Ali

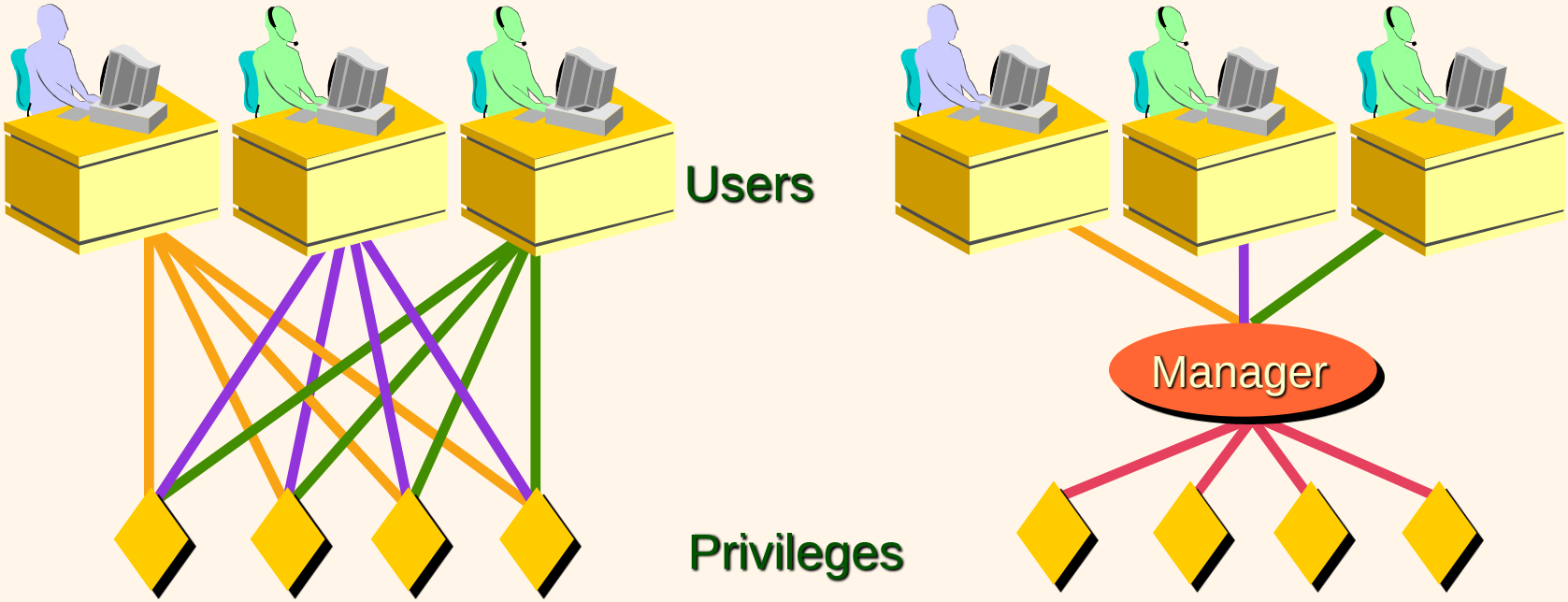
What is a Role?

- A role is a named group of related privileges that can be granted to the user.
- Makes it easier to revoke and maintain privileges.
- A user can have access to several roles, and several users can be assigned the same role.
- Roles are typically created for a database application.
- Syntax:
  - CREATE ROLE role;

Example of a Role

- CREATE ROLE manager;
  - Role created.
- GRANT create table TO manager;
  - Grant succeeded.
- GRANT manager TO User_name;
  - Grant succeeded.
- PRIVILEGES ARE GRANTED TO ROLES
- PEOPLE ARE ASSIGNED ROLES

Why Roles are easier?



Allocating privileges without a role

Allocating privileges with a role

Viewing privilege in data dictionary

- Types of privilege you can view

Data Dictionary Table	Description
ROLE_SYS_PRIVS	System privileges granted to roles
ROLE_TAB_PRIVS	Table privileges granted to roles
USER_ROLE_PRIVS	Roles accessible by the user
USER_TAB_PRIVS_MADE	Object privileges granted on the user's objects
USER_TAB_PRIVS_RECD	Object privileges granted to the user
USER_COL_PRIVS_MADE	Object privileges granted on the columns of the user's objects
USER_COL_PRIVS_RECD	Object privileges granted to the user on specific columns

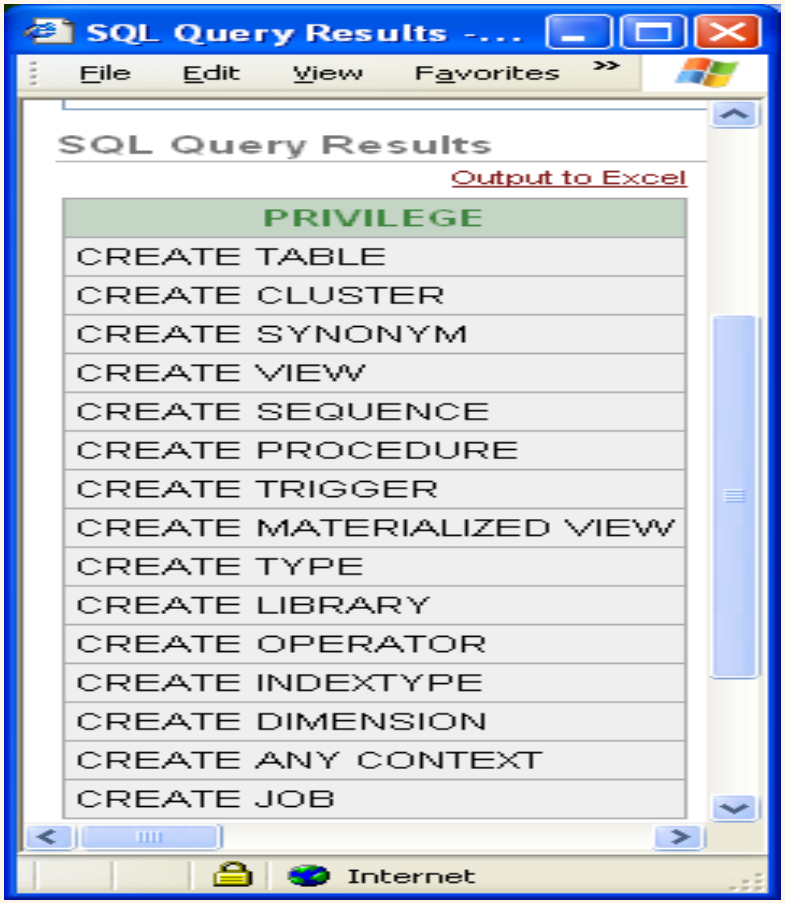
Example of privileges commands

- SELECT *
  FROM role_tab_privs
  WHERE role = 'MANAGER';
- SELECT *
  FROM user_sys_privs;
- SELECT *
  FROM user_role_privs;

Displaying your privileges

- To show what privileges a user has on the databases enter:
- SELECT * FROM SESSION_PRIVS ;
- You have a list of privileges you have displayed.
- Run the command to see what you get. See next slide.

Results of previous command



GRANT/REVOKE on Views

- If the creator of a view loses the SELECT privilege on an underlying table, the view is dropped!

- If the creator of a view loses a privilege held with the grant option on an underlying table, (s)he loses the privilege on the view as well; so do users who were granted that privilege on the view!

Views and Security

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
  - Given ActiveSailors, but not Sailors or Reserves, we can find sailors who have a reservation, but not the *bid*'s of boats that have been reserved.
- Creator of view has a privilege on the view if (s)he has the privilege on all underlying tables.
- Together with GRANT/REVOKE commands, views are a very powerful access control tool.

Example:

Consider the following view defined on table S by DBA:

```
CREATE View V as
    Select * from S where Field='Computer‘
```

And this authority rule has been defined:

```
GRANT Retrieve
ON V
TO Ali
```

Now, is Ali able to run the following query???

Ali: Select * from S where Field='Computer'

Security to the Level of a Field!

- Can create a view that only returns one field of one tuple. (How?)
- Then grant access to that view accordingly.
- Allows for *arbitrary* granularity of control, *but*:
  - Clumsy to specify, though this can be hidden under a good UI
  - Performance is unacceptable if we need to define field-granularity access frequently. (Too many view creations and look-ups.)

Checking the authority rules

1- When a user submits a query, the DBMS considers the union of all related authority rules. He will be permitted if he has given permission in at least one authority rule.

2- If the user wants to see sth further than his privilege, there are two major methods in system's reaction:

I- Request modification

II- Denial of access and error message

Checking the authority rules

Example in Quel:

Define Permit Retrieve on P To John
Where city = "London"

John: Retrieve (p.p# , p.weight)
Where p.color = "red"

Request modification

Retrieve (p.p# , p.weight)
Where city = "London" and p.color = "red"

?