Databases (Key: 1644)
What do I need to connect to a DB?

---------- Arellanes Conde Esteban ----------

Contents

1 Homework: Connecting to a Database **3**
1.1 Requirements to Connect to a Database 3
1.2 Permissions at System and Object Levels 4
1.3 Granting and Revoking Permissions 4
1.4 Difference Between Roles and Users 6

2 References **7**

1 Homework: Connecting to a Database

1.1 Requirements to Connect to a Database

To establish a connection with a database, the following elements are required:

- A database management system (DBMS) such as PostgreSQL, MySQL, or SQL Server.
- A database client tool (e.g., `psql`, MySQL Workbench, or pgAdmin).
- Network access to the database server.
- Authentication credentials (username and password).
- Appropriate permissions granted by the database administrator.

Example command to connect to a PostgreSQL database:

```
psql -U user -h 10.12.75.16 -p 5432 -d student -W
```

According to Oracle Cloud[1] these are also some prerequisites for creating an Oracle-DB connection:

- Ensure that you have write permissions to the database.
- Ensure that you have the required permissions to run stored procedures and SQL statements.
- Know the database URL, including the hostname or IP address and the port number.
- Know the database system ID and service name.
- Know the username and password for connecting to the database.

[1] https://docs.oracle.com/en/cloud/paas/integration-cloud/database-adapter/prerequisites-creating-connection.html

1.2 Permissions at System and Object Levels

Quoting IBM[2] the Permissions determine what types of actions users can perform in the ObjectServer. You assign permissions to roles by using the `GRANT` command. There are two types of permissions:

System-Level Permissions: These control actions on the database structure and control the commands that can be run in the ObjectServer DB.

- `CREATE` - Allows creating new databases, tables, and other objects.
- `DROP` - Allows deleting databases or tables.
- `ALTER` - Allows modifying existing database structures.

Object-Level Permissions: These apply to specific tables or objects, which control access to individual objects, such as tables.

- `INSERT` - Allows inserting new records.
- `SELECT` - Allows reading data from tables.
- `UPDATE` - Allows modifying existing records.
- `DELETE` - Allows removing records.

1.3 Granting and Revoking Permissions

Permissions in a database are managed using the `GRANT` and `REVOKE` SQL commands.

- **Granting permissions:**

```
GRANT SELECT, INSERT ON students TO user_name;
```

- **Revoking permissions:**

```
REVOKE DELETE ON students FROM user_name;
```

[2]https://www.ibm.com/docs/en/netcoolomnibus/8.1?topic=roles-system-object-permissions

Every SQL Server securable has associated permissions that can be granted to a principal. Permissions in the Database Engine are managed at the server level assigned to logins and server roles, and at the database level assigned to database users and database roles. Example:

```
1    GRANT SELECT ON SCHEMA::HumanResources TO role_HumanResourcesDept;
2    REVOKE SELECT ON SCHEMA::HumanResources TO role_HumanResourcesDept;
```

Listing 1: "SQL Permission Commands"

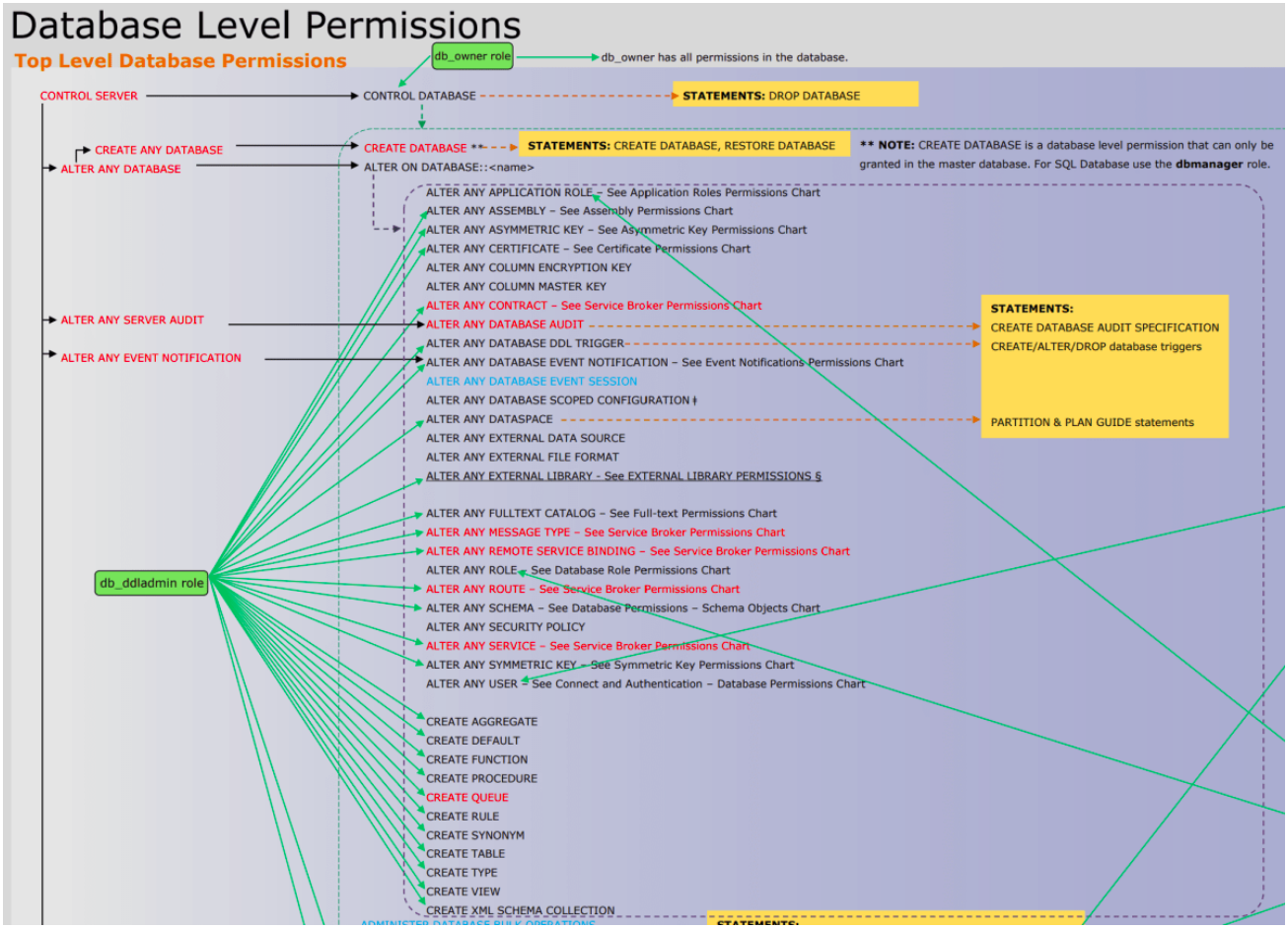

Figure 1: SQL Permissions

Microsoft Permissions Database Engine[3]

[3]https://learn.microsoft.com/en-us/sql/relational-databases/security/permissions-database-engine

1.4 Difference Between Roles and Users

Roles simplify permission management by allowing administrators to assign multiple users to a predefined role[4]. In the case of PostgreSQL database manager. It manages access permissions using the concept of roles.

A role can be thought of as either a database user, or a group of database users, depending on how the role is set up. Roles can own database objects (for example, tables) and can assign privileges on those objects to other roles to control who has access to which objects. Furthermore, it is possible to grant membership in a role to another role, thus allowing the member role use of privileges assigned to the role it is a member of.

The concept of roles subsumes the concepts of "`users`" and "`groups`". In PostgreSQL versions before 8.1, users and groups were distinct kinds of entities, but now there are only roles. Any role can act as a user, a group, or both. In less words:

- **User:** A single account that can access the database.
- **Role:** A group of permissions that can be assigned to multiple users.

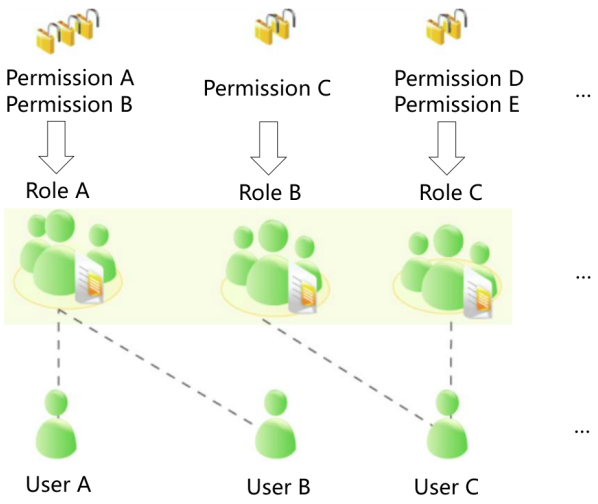

Figure 2: Users & Roles. Retrieved from docs.otc.t-systems.com

[4]https://www.postgresql.org/docs/8.1/user-manag.html

Example:

```
    CREATE ROLE name;
    DROP ROLE name;
    createuser name
    dropuser name
    SELECT rolname FROM pg_roles;
```

Listing 2: "PostgreSQL Roles Commands"

2 References

[1] Ramez Elmasri, Shamkant B. Navathe. Fundamentals of Database Systems (GlobalEdition). Pearson 2017

[2] De Miguel Martínez, Adoración, Piattini, Mario, Esperanza, Marcos Diseño de bases de datos relacionales. México, Alfaomega, 2000.

[3] Kriegel, Alex; Trukhnov. SQL Bible, second edition, Willey 2008.

[4] Alan Beaulieu, Learning SQL: Generate, Manipulate, and Retrieve Data, O'Reilly, 3rd. Edition - 2020.

[5] Joan Casteel, Oracle 12c: SQL. Cengage Learning, Sep 11, 2015.

[6] IEEE, "Guide to Software Requirements Specifications," IEEE Std 830-1984. `https://chumpolm.files.wordpress.com/2018/09/ieee-std-830-1984.pdf`