Database Administration and Security

A. Roles of Data Administrator and Database Administrator

- **Data Administrator (DA) –** a management-oriented role that concerns controlling the overall corporate data resources, both computerized and manual.
- **Database Administrator (DBA) –** a person who maintains a successful database environment by directing or performing all related activities to keep the data secure.
- It is the responsibility of the **Data Administrator (DA)** to determine the contents and logical boundaries of each database**.** **Database Administrators (DBA)** are responsible for the design, implementation, maintenance, and security of physical structures (databases). DA first builds a logical model of the database, which is later implemented by DBA.

Contrasting DA and DBA Activities and Tasks	
Data Administrator (DA)	**Database Administrator (DBA)**
Builds logical design	Facilitates the development and use of the database
Has a managerial orientation	Has a technical orientation
Analyze and perform business data requirements	Analyze data volumes and space requirements in DBMS
Define policies and standards (definition, naming, abbreviation)	Enforces policies and programming standards

- DA and DBA functions to organizations may tend to overlap. However, when the organization does not include a DA position, the DBA executes some of the DA's functions. In this combined role, the DBA must have a diverse mix of technical and managerial skills.
  - DBA's managerial services include supporting end-users, defining and enforcing policies and programming standards for the database, providing data backup and recovery services, and monitoring distribution and use of the data in the database.

Database Security

- **Database security** refers to DBMS features and other related measures that comply with the organization's security requirements. From the DBA's point of view, security measures should be implemented to protect the DBMS against service degradation and to protect the database against loss, corruption, or mishandling.
- It is essential to implement security within the organization to make sure that the right people have access to the right data. Without these security measures in place, you might find someone destroying your valuable data, or selling your company's secrets to your competitors, or someone invading the privacy of others.
- The **SQL security model** provides a basic syntax used to specify security restrictions. The DBMS will then implement the security system and enforce the required restrictions.

There are four (4) common concepts of SQL security as follows:
- **Users** – This represents people or programs performing actions on objects in the database. The DBMS grants users an ID for authentication and privileges to perform specific actions on specific tables/rows.
- **Objects** – These are the things defined by SQL standards in the database that users can manipulate. This includes rows, columns, tables, indexes, and views.
- **Privileges** – This refers to the rights of users to manipulate objects. These privileges start with SELECT, INSERT, DELETE, and UPDATE, ALTER, INDEX, AND REFERENCES for database objects.
- **Roles** – is a named collection of database access privileges that authorize a user to connect to the database and use its system resources.

B. Users

- Setting up security begins with **Authentication and Authorization**. Individual users, groups, or processes granted access to the SQL server instance either at server level or database level. **Server-level** includes logins and server roles. **Database-level** include users and database roles.

- **Login** allows you to connect to the SQL Server service (also called an instance), and permissions inside the database are granted to the database **users**.
- Logins must be mapped to a database user to connect to a database. If your login is not mapped to any database user, you can still connect to SQL Server instance using SQL Server Management Studio (SSMS), but you are not allowed to access any objects in the database. To access any objects in the database, you must have a login that is mapped to a user in the database, and that the user must be granted appropriate rights in the database.

Here is the syntax to **create** a new login in T-SQL:

```
CREATE LOGIN login_name_test WITH PASSWORD = 'Mypassword';
```

S: To check the newly created login or any changes in login, open the Object Explorer window under the SQL server, expand the security folder, and expand the login folder.

Here is the syntax for changing any credentials of the login
For the login name:

```
ALTER LOGIN login_name_test WITH NAME = newlogin_name;
```

For the password:

```
ALTER LOGIN newlogin_name WITH PASSWORD = 'Newpassword';
```

Assume that we have a database named *myDB* and a table named *Names* having the following values:

ID	LastName	FirstName
1	Escalona	John Smith
2	Cequena	Abby

Now we will create a new user under the *myDB* database and the login account that we have created before.
Here is the syntax:

```
USE myDB
GO
CREATE USER user1 FOR LOGIN newlogin_name
```

C. Privileges

- When multiple users can access database objects, authorization can be controlled to these objects with privileges. Every object has an owner. Privileges control if a user can modify an object owned by another user. Access privileges in relational databases are assigned through SQL GRANT and REVOKE commands.
- **GRANT** – is a command used to provide access or privileges on the database objects to the users.
- **REVOKE** – is a command used to remove privileges from a specific user or role or from all users to perform actions on database objects.

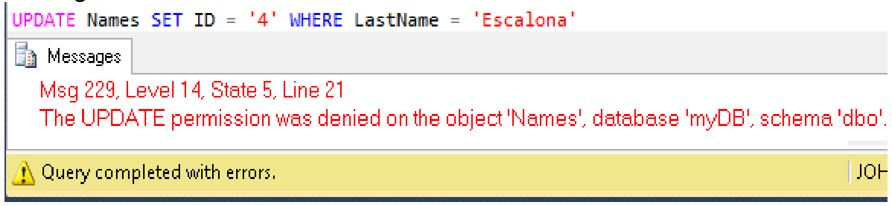
The user named *user1* that we have created before does not have any access to myDB. Using the **GRANT** command, we want *user1* to be able to view and modify the data in the myDB database. Here is the syntax:

```
USE myDB
GO
GRANT SELECT, UPDATE ON [Names] TO user1
```

Now, we want to remove the modifying privilege of *user1*. Here is the syntax:

```
REVOKE UPDATE ON [Names] TO user1
```

If we try to execute the following statement using the login account *newlogin_name* that we have created before, it will show an error like this.



```
UPDATE Names SET ID = '4' WHERE LastName = 'Escalona'
```

```
Msg 229, Level 14, State 5, Line 21
The UPDATE permission was denied on the object 'Names', database 'myDB', schema 'dbo'.
```

D. Roles

- A role is a collection of privileges that can be granted to one or more users or other roles. Roles help you grant and manage sets of privileges for various categories of users, rather than grant those privileges to each user individually.
- The main benefit of roles is efficient management. Imagine a group of 1,000 users suddenly needing to view or modify new data. Instead of modifying 1,000 user accounts, you can simply select a database-level role and assign it to a user. There are two types of

database-level roles**: fixed-database roles** that are predefined in the database and **user-defined database roles** that you can create on your own based on your preferences.

The following table shows the fixed-database roles and their capabilities.

Fixed-Database role name	Description
db_owner	Members of the db_owner fixed database role can perform all configuration and maintenance activities on the database and can also drop the database in SQL Server.
db_securityadmin	Members of the db_securityadmin fixed database role can modify role membership for custom roles only and manage permissions. Members of this role can potentially elevate their privileges.
db_accessadmin	Members of the db_accessadmin fixed database role can add or remove access to the database for Windows logins, Windows groups, and SQL Server logins.
db_backupoperator	Members of the db_backupoperator fixed database role can back up the database.
db_ddladmin	Members of the db_ddladmin fixed database role can run any Data Definition Language (DDL) command in a database.
db_datawriter	Members of the db_datawriter fixed database role can add, delete, or change data in all user tables.
db_datareader	Members of the db_datareader fixed database role can read all data from all user tables.
db_denydatawriter	Members of the db_denydatawriter fixed database role cannot add, modify, or delete any data in the user tables within a database.
db_denydatareader	Members of the db_denydatareader fixed database role cannot read any data in the user tables within a database.

We can assign a *db_owner* role to the user that we previously created using this syntax:

```
USE myDB
GO
ALTER ROLE db_owner ADD MEMBER user1
```

We can also create a user-defined database role having the privilege of viewing and updating the data in table *Names*. Here is the syntax:

```
USE myDB
GO
CREATE ROLE [Admin1]
GO
GRANT SELECT, UPDATE ON [Names]
TO [Admin1]
GO
```

REFERENCES

Coronel, C. and Morris, S. (2018). *Database systems design, implementation, & management* (13th ed.). Cengage Learning.

Elmasri, R. & Navathe, S. (2016). *Fundamentals of database systems* (7th ed.). Pearson Higher Education.

Kroenke, D. & Auer, D. *Database processing: Fundamentals, design, and implementation* (12th ed.). Pearson Higher Education.

Silberschatz A., Korth H.F., & Sudarshan, S. (2019). *Database system concepts* (7th ed.). McGraw-Hill Education.