DATABASE MANAGEMENT SYSTEMS (IT 2040)

LECTURE 08 – DATABASE SECURITY

LEARNING OUTCOMES

- At the end of the lecture, the students should be able to:
  - Explain the three important aspects of security
  - Identify different roles and permissions in SQL server at server and database level
  - Implement a security policy of a given database using access control mechanisms in SQL server

LECTURE CONTENT

- Aspects of database security
- Access control mechanisms
- Roles, Users and Permissions in SQL server

DATABASE SECURITY

- The data stored in a DBMS is often vital to the business interests of the organization and is regarded as a corporate asset.
- In addition to protecting the intrinsic value of the data, corporations must consider ways to ensure privacy and to control access to data that must not be revealed to certain groups of users for various reason.

ASPECTS OF DATABASE SECURITY

- There are three main objectives to consider while designing a secure database application:
  - **Secrecy**: Information should not be disclosed to unauthorized users.
    - For example, a student should not be allowed to examine other students' grades.
  - **Integrity**: Only authorized users should be allowed to modify data.
    - For example, students may be allowed to see their grades, yet not allowed to modify them.
  - **Availability**: Authorized users should not be denied access.
    - For example, an instructor who wishes to change a grade should be allowed to do so.

SECURITY POLICY & ACCESS CONTROLS

- To achieve the objectives in the previous slide, a clear and consistent security policy should be developed
- A security policy specifies who is authorized to do what.
- Most users need to access only a small part of the database to carry out their tasks.
- Allowing users unrestricted access to all the data can be undesirable.
- Two main mechanisms at the DBMS level to control access to data
  - Discretionary access control
  - Mandatory access control

DISCRETIONARY ACCESS CONTROL

- Discretionary access control is based on the concept of access rights, or privileges, and mechanisms for giving users such privileges.
- A privilege allows a user to access some data object in a certain manner (e.g., to read or to modify).
- A user who creates a database object such as a table or a view automatically gets all applicable privileges on that object.
- The DBMS subsequently keeps track of how these privileges are granted to other users and ensures that at all times only users with the necessary privileges can access an object.

SECURITY IN SQL SERVER

- A SQL Server instance contains a hierarchical collection of entities, starting with the server.
- Each server contains multiple databases, and each database contains a collection of securable objects.
- The SQL Server security framework manages access to securable entities through authentication and authorization
  - Authentication : verifying the identities of users trying to connect to a SQL Server instance
  - Authorization : determines which data resources that authorized users can access and what actions they can take.

SECURITY IN SQL SERVER (CONTD.)

- Authentication and authorization are achieved in SQL Server through a combination of security principals, securable, and permissions.
- Together, these three component types provide a structure for authenticating and authorizing SQL Server users.
- You must grant each principal the appropriate permissions it needs on specific securables to enable users to access SQL Server resources.

PRINCIPALS, SECURABLE AND PERMISSIONS

- **Principals :** Individuals, groups, or processes granted access to the SQL Server instance, either at the server level or database level.
  - Server-level principals include logins and server roles.
  - Database-level principals include users and database roles.
- **Securable:** Objects that make up the server and database environment. The objects can be broken into three hierarchical levels:
  - Server-level securables include such objects as databases and availability groups.
  - Database-level securables include such objects as schemas and full-text catalogs.
  - Schema-level securables include such objects as tables, views, functions, and stored procedures.
- **Permissions:** The types of access permitted to principals on specific securables. You can grant or deny permissions to securables at the server, database, or schema level.

STEPS TO PROVIDE USERS WITH ACCESS TO SQL SERVER RESOURCES

1. At the server level, create a login for each user that should be able to log into SQL Server.
   - create Windows authentication logins that are associated with Windows user or group accounts,
   - create SQL Server authentication logins that are specific to that instance of SQL Server.
2. Create user-defined server roles if the fixed server roles do not meet your configuration requirements.
3. Assign logins to the appropriate server roles (either fixed or user-defined).
4. For each applicable server-level securable, grant or deny permissions to the logins and server roles.

STEPS TO PROVIDE USERS WITH ACCESS TO SQL SERVER RESOURCES (CONTD.)

5. At the database level, create a database user for each login.
   - A database user can be associated with only one server login.
6. Create user-defined database roles if the fixed database roles do not meet your configuration requirements.
7. Assign users to the appropriate database roles (either fixed or user-defined).
8. For each applicable database-level or schema-level securable, grant or deny permissions to the database users and roles.

SCENARIO

- A software company is assigned to create a software system for a super market.
- The project manager responsible to develop the system assigns the task of creating the databases related to the projects (inventory database, payroll database) to a senior DBA (Kamal)
- The senior DBA creates the databases and all logins for the users
- The senior dba then assigns a junior DBA (Namali & Janaka) to look after each database.
- The junior DBA designs the database and assigns the task of creating tables to a database developer (Sahan)
- The junior DBA also assigns the task of entering data to some data entry operators (Amali and Mihiran)

CREATING A LOGIN USING WINDOWS AUTHENTICATION

- A login is an individual user account for logging into the SQL Server instance.
- Syntax

```
CREATE LOGIN [domain_name\login_name]
FROM WINDOWS
[ WITH DEFAULT_DATABASE = database_name
| DEFAULT_LANGUAGE = language_name ];
```

- Example

```
CREATE LOGIN [win10b\winuser01] FROM WINDOWS
WITH DEFAULT_DATABASE = master, DEFAULT_LANGUAGE = us_english;
```

CREATING A LOGIN USING SQL SERVER

- Syntax

```
CREATE LOGIN [Login Name] -- This is the User that you use for login
WITH PASSWORD = 'provide_password' MUST_CHANGE,
CHECK_EXPIRATION = ON,  CHECK_POLICY = ON,  DEFAULT_DATABASE = [Database Name],
DEFAULT_LANGUAGE = [Language Name];-- This is Optional
```

- Example

```
CREATE LOGIN sqluser01
WITH PASSWORD = 'tempPW@56789'
MUST_CHANGE, CHECK_EXPIRATION = ON,
DEFAULT_DATABASE = master, DEFAULT_LANGUAGE = us_english;
```

CREATING AND ASSIGNING SERVER ROLES

- With sever roles logins could be grouped together in order to more easily manage server-level permissions.
- SQL Server supports fixed server roles and user-defined server roles.
- With fixed server roles logins could be assigned, but permissions cannot be changed.
- For user-defined roles logins can be assigned and permission can be changed

ADDING LOGINS TO FIXED SEVER ROLES

- Assigning logins to sever roles
  - Syntax : ALTER SERVER ROLE server_role_name ADD MEMBER login
  - Server role names :

Role	Description
Sysadmin	Members of the **sysadmin** fixed server role can perform any activity in the server.
dbcreator	Members of the **dbcreator** fixed server role can create, alter, drop, and restore any database.
securityadmin	Members of the securityadmin fixed server role manage logins and their properties. They can GRANT, DENY, and REVOKE server-level permissions.

  - Ex : ALTER SERVER ROLE diskadmin ADD MEMBER Ted

CREATING USER-DEFINED SERVER ROLES AND PERMISSIONS

- Creating a user defined sever role and assigning permissions
  - Syntax for creating a server role: CREATE SERVER ROLE role_name
    - Ex: CREATE SERVER ROLE devops;
- Syntax for granting permissions : GRANT permission TO grantee_principal [WITH GRANT OPTION]
  - Permission Examples :
    - CREATE ANY DATABASE — Create a database on the server.
    - ALTER ANY DATABASE — Create, alter, or drop any login in the instance.
    - ALTER ANY LOGIN — Modify any login.
    - SHUTDOWN — Shut down the server.
  - Ex: GRANT ALTER ANY DATABASE TO devops;

REVISIT TO SCENARIO

- Senior DBA, Junior DBA, Database Developer and Data entry operator should have logins.
- According to the scenario, Senior dba also should be able to create databases and logins.
- Therefore two methods could be used to create logins
  - All logins could be created by the project manager
  - Project manager could create the login of Senior DBA and Senior DBA could create the other logins. (we will use this approach)

REVISIT TO SCENARIO (CONTD.)

- First create the login for the senior DBA using windows authentication of SQL server authentication as follows:
- Windows authentication

```
CREATE LOGIN PC2\kamal
FROM WINDOWS WITH DEFAULT_DATABASE = Master
```

- SQL server authentication

```
CREATE LOGIN Kamal
WITH PASSWORD = 'kamal@123', DEFAULT_DATABASE = Master
```

REVISIT TO SCENARIO (CONTD.)

- Now let's give permission to the senior DBA to create databases and logins.
- This could be done using pre-defined sever roles or user defined server roles as follows
- Using pre-defined server roles

  alter server role dbcreator add member kamal

  alter server role securityadmin add member kamal
- Using user-defined server role

  create server role seniordba

  grant create any database,alter any login to seniordba

  alter server role seniordba add member kamal

DATABASE USERS

- Logins are created at the server level, while users are created at the database level.
- In other words, a login allows to connect to the SQL Server service and permissions inside the database are granted to the database users, not the logins.
- The logins will be assigned to server roles (for example, *serveradmin*)
- The database users will be assigned to roles within that database
- Syntax : CREATE USER user_name FOR LOGIN login_name
  - user_name : The name of the database user that you wish to create.
  - login_name : The Login used to connect to the SQL Server instance

REVISIT TO SCENARIO

- Senior DBA can now create the databases and create a logins for the junior DBAs who are in-charge of the databases
- Create the login first

  CREATE LOGIN namali

  WITH PASSWORD = 'namali@123', DEFAULT_DATABASE = Inventory
- The senior DBA can also create all the other logins for the other roles here
- Now let's create the user for the login so that the user can be given permission at the database level

  Create user namali for login namali

CREATING DATABASE ROLES

- A database role is a group of users that share a common set of database-level permissions.
- As with server roles, SQL Server supports both fixed and user-defined database roles.
- To set up a user-defined database role, you must create the role, grant permissions to the role, and add members to the role (or add members and then grant permissions).

ASSIGNING USERS TO FIXED DATABASE ROLES

- Some examples of database roles are as follows:

Role	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.
db_accessadmin	Members of the **db_accessadmin** fixed database role can add or remove access to the database
db_ddladmin	Members of the **db_ddladmin** fixed database role can run any Data Definition Language (DDL) command in a database.

- Syntax for windows authentication :ALTER ROLE db_datawriter ADD MEMBER <domain\username>
- Syntax for SQL authentication :ALTER ROLE db_datawriter ADD MEMBER <username>;

REVISIT TO SCENARIO

- The junior DBAs are expected to manage the databases.
- Therefore they could be assigned with the db_owner role by the senior DBA as follows :

```
use inventoryDB

alter role db_owner add member namali

alter role db_securityadmin add member namali

alter role db_accessadmin add member namali
```

ASSIGNING USERS TO USER DEFINED DATABASE ROLES

- Creating a user defined role
  - Syntax : CREATE ROLE <role_name>
  - Example : CREATE ROLE student
- Assigning login to the role
  - Syntax : ALTER ROLE <role_name> ADD MEMBER <username>
  - Example : ALTER ROLE student ADD MEMBER Brady

PERMISSIONS

- Every SQL Server securable has associated permissions that can be granted to user.
- At the database level they are assigned to database users and database roles.
- Some permissions are as follows :

Role	Description
ALTER	Grants or denies the ability to alter the existing database.
CREATE TABLE	Grants or denies the ability to create a table.
INSERT	Grants or denies the ability to issue the INSERT command against all applicable objects within the database.
EXECUTE	Grants or denies the ability to issue the EXECUTE command against all applicable objects within the database.
REFERENCES	The REFERENCES permission on a table is needed to create a FOREIGN KEY constraint

CREATING AND REMOVING PERMISSIONS

- Creating and removing permissions could be done using GRANT, DENY and REVOKE
  - **GRANT** – Grants permissions on a securable to a principal.
  - **DENY** – Denies a permission to a principal.
  - **REVOKE** – Removes a previously granted or denied permission.

GRANTING PERMISSIONS

- Granting permissions to a role/user
  - Ex: GRANT SELECT TO GrantSelectRole
  - Ex: GRANT SELECT, INSERT, UPDATE, DELETE, REFERENCES, EXECUTE ON Northwind TO sqluser01;
- Denying permissions to a role/user
  - Ex: DENY SELECT TO DenyTest
- Revoking permissions assigned to a user
  - Ex: REVOKE SELECT TO GrantSelectRole;

REVISIT TO SCENARIO

- The database developers need to create tables which would probably have foreign keys
- Lets create a role names database developer for this

```
create role databaseDev

grant create table,  alter, references to databaseDev on inventoryDB

alter role databaseDev add member Sahan
```

- Similarly create another role with select and insert for the data Entry operators

VIEWS AND SECURITY

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
- 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

Thank you!