Authorization and Permissions in SQL Server

03/30/20174 minutes to readContributors all

In this article

1. The Principle of Least Privilege
2. Role-Based Permissions
3. Permissions Through Procedural Code
4. Permission Statements
5. Ownership Chains
6. Procedural Code and Ownership Chaining
7. External Resources
8. See Also

When you create database objects, you must explicitly grant permissions to make them accessible to users. Every securable object has permissions that can be granted to a principal using permission statements.

The Principle of Least Privilege

Developing an application using a least-privileged user account (LUA) approach is an important part of a defensive, in-depth strategy for countering security threats. The LUA approach ensures that users follow the principle of least privilege and always log on with limited user accounts. Administrative tasks are broken out using fixed server roles, and the use of the `sysadmin` fixed server role is severely restricted.

Always follow the principle of least privilege when granting permissions to database users. Grant the minimum permissions necessary to a user or role to accomplish a given task.

> **Important**
>
> Developing and testing an application using the LUA approach adds a degree of difficulty to the development process. It is easier to create objects and write code while logged on as a system administrator or database owner than it is using a LUA account. However, developing applications using a highly privileged account can obfuscate the impact of reduced functionality when least privileged users attempt to run an application that requires elevated permissions in order to function correctly. Granting excessive permissions to users in order to reacquire lost functionality can leave your application vulnerable to attack. Designing, developing and testing your application logged on with a LUA account enforces a disciplined approach to security planning that eliminates unpleasant surprises and the temptation to

grant elevated privileges as a quick fix. You can use a SQL Server login for testing even if your application is intended to deploy using Windows authentication.

Role-Based Permissions

Granting permissions to roles rather than to users simplifies security administration. Permission sets that are assigned to roles are inherited by all members of the role. It is easier to add or remove users from a role than it is to recreate separate permission sets for individual users. Roles can be nested; however, too many levels of nesting can degrade performance. You can also add users to fixed database roles to simplify assigning permissions.

You can grant permissions at the schema level. Users automatically inherit permissions on all new objects created in the schema; you do not need to grant permissions as new objects are created.

Permissions Through Procedural Code

Encapsulating data access through modules such as stored procedures and user-defined functions provides an additional layer of protection around your application. You can prevent users from directly interacting with database objects by granting permissions only to stored procedures or functions while denying permissions to underlying objects such as tables. SQL Server achieves this by ownership chaining.

Permission Statements

The three Transact-SQL permission statements are described in the following table.

Permission Statement	Description
GRANT	Grants a permission.
REVOKE	Revokes a permission. This is the default state of a new object. A permission revoked from a user or role can still be inherited from other groups or roles to which the principal is assigned.

Permission Statement	Description
DENY	DENY revokes a permission so that it cannot be inherited. DENY takes precedence over all permissions, except DENY does not apply to object owners or members of `sysadmin`. If you DENY permissions on an object to the `public` role it is denied to all users and roles except for object owners and `sysadmin` members.

- The GRANT statement can assign permissions to a group or role that can be inherited by database users. However, the DENY statement takes precedence over all other permission statements. Therefore, a user who has been denied a permission cannot inherit it from another role.

Note

Members of the `sysadmin` fixed server role and object owners cannot be denied permissions.

Ownership Chains

SQL Server ensures that only principals that have been granted permission can access objects. When multiple database objects access each other, the sequence is known as a chain. When SQL Server is traversing the links in the chain, it evaluates permissions differently than it would if it were accessing each item separately. When an object is accessed through a chain, SQL Server first compares the object's owner to the owner of the calling object (the previous link in the chain). If both objects have the same owner, permissions on the referenced object are not checked. Whenever an object accesses another object that has a different owner, the ownership chain is broken and SQL Server must check the caller's security context.

Procedural Code and Ownership Chaining

Suppose that a user is granted execute permissions on a stored procedure that selects data from a table. If the stored procedure and the table have the same owner, the user doesn't need to be granted any permissions on the table and can even be denied permissions. However, if the stored procedure and the table have different owners, SQL Server must check the user's permissions on the table before allowing access to the data.

Note

Ownership chaining does not apply in the case of dynamic SQL statements. To call a procedure that executes an SQL statement, the caller must be granted permissions on the underlying tables, leaving your application vulnerable to SQL Injection attack. SQL Server provides new mechanisms, such as impersonation and signing modules with certificates, that do not require granting permissions on the underlying tables. These can also be used with CLR stored procedures.

External Resources

For more information, see the following resources.

Resource	Description
Permissions in SQL Server Books Online	Contains topics describing permissions hierarchy, catalog views, and permissions of fixed server and database roles.

See Also

Securing ADO.NET Applications
Application Security Scenarios in SQL Server
Authentication in SQL Server
Server and Database Roles in SQL Server
Ownership and User-Schema Separation in SQL Server
ADO.NET Managed Providers and DataSet Developer Center