# Public Role in SQL Server

blog.netwrix.com/2022/11/18/public-role-sql-server

Database roles are similar to Windows groups — rather than revoking or granting access to each user separately, administrators manage access by granting or revoking permissions from roles and  by changing role membership. Using roles makes it easier to accurately grant and revoke privileges for database users. And since multiple users can be members of a SQL database role, you can easily manage rights for a whole group of users at once.

Handpicked related content:

[Free Guide] SQL Server Hardening Best Practices

In this post, we explain the **public** role in Microsoft SQL Server and some of the best practices related to it. (We are deliberately using lowercase letters here, since that is how the public role is spelled in the SQL Server world, unlike in the Oracle world.)

## Types of Roles in SQL Server

Microsoft SQL Server offer several types of pre-built roles:

- **Fixed server** roles — They are called “fixed” server roles because, except for the public role, they cannot be modified or drop
- **Fixed database** roles
- **Application** role — A database principal that can be used by an application to run with its own set of permissions (disabled by default)
- **User-defined roles** (starting with SQL Server 2012) — Only server-level permissions can be added to user-defined

## Where the Public Role Fits In

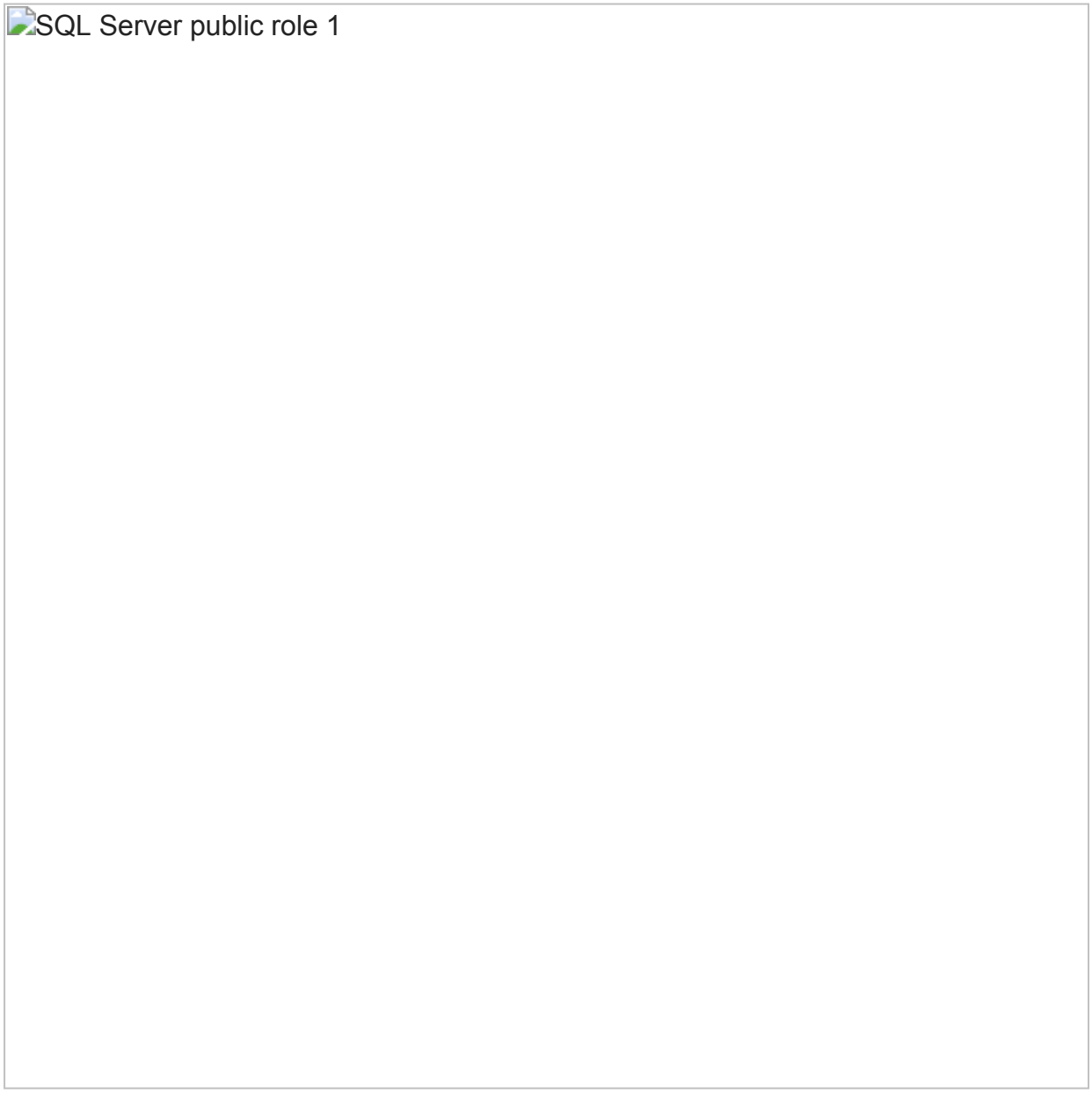
All database platforms come with a pre-defined role called public, but the implementation of this role varies by platform. In SQL Server, the public role is part of the fixed server role, and permissions can be granted to, denied to or revoked from the SQL Server public role permissions.

When a SQL Server login is created, the public role is assigned to the login and cannot be revoked. If the server principal is not granted or denied specific permissions on a securable object, the login will automatically inherit the permissions for that object that are granted to the public role.

## Permissions Assigned to the Public Role

In order to maintain SQL Server security and comply with many regulations, including PCI DSS and HIPAA, you need to know all the server and database-level roles assigned to each user. Let's review the server-level permissions assigned to the public role by using a Management Studio transact-SQL query:

```
SELECT sp.state_desc as "Permission State", sp.permission_name as "Permission",
sl.name "Principal Name",sp.class_desc AS "Object Class", ep.name "End Point"
FROM sys.server_permissions AS sp
  JOIN sys.server_principals AS sl
    ON sp.grantee_principal_id = sl.principal_id
  LEFT JOIN sys.endpoints AS ep
    ON sp.major_id = ep.endpoint_id
WHERE sl.name = 'public';
```




As the table above shows, only five server-level permissions are assigned to the public role. Note that the VIEW ANY DATABASE permission does not give users access to any database objects; it only allows them to list all the databases in a SQL Server instance.

Therefore, if you create a new login and assign no other roles or permissions, the user can only log in to the instance and is unable to do anything else.

## Permissions when a SQL Server Login Is Assigned a Default Database

Now, let's create SQL Server login that uses SQL Authentication by assigning a default database to that user.

```
USE [master]
GO
CREATE LOGIN [SQTest] WITH
PASSWORD=N'nhggLboBn6SHolSWfipjzO/7GYw8M2RMbCt1LsCTK5M=', DEFAULT_DATABASE=
[SBITS], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
```

If we log in as that user and list all the database level permissions for the user's default database SBITS, we can see what permissions they have. As shown below, the user Stealth does not have any permissions on the SBITS database even though that is the user's default database. In other words, just because a default user database is assigned to a user login, it does not mean the user will be able to see the database objects or data.

To verify this, we can use the following script:

```
EXECUTE AS LOGIN= 'SQTest';
GO
USE SBITS
GO
SELECT dp.state_desc AS "Class Description", dp.permission_name AS "Permission
Name",
SCHEMA_NAME(ao.schema_id) AS "Schema Name", ao.name AS "Object Name"
FROM sys.database_permissions dp
 LEFT JOIN sys.all_objects ao
  ON dp.major_id = ao.object_id
 JOIN sys.database_principals du
  ON dp.grantee_principal_id = du.principal_id
WHERE du.name = 'TestLoginPerms'
 AND ao.name IS NOT NULL
ORDER BY ao.name;
REVERT
```


## Permissions Inherited by the Public Role on the Master Database

Since the user login Stealth is part of the **public** role by default, let's see the permissions inherited by the **public** role on the **master** database.

```
USE master;
GO
 SELECT sp.state_desc AS "Permission State", sp.permission_name AS "Permission",
 SCHEMA_NAME(ao.schema_id) AS 'Schema', ao.name AS "Object Name"
 FROM sys.database_permissions sp
  LEFT JOIN sys.all_objects ao
     ON sp.major_id = ao.object_id
   JOIN sys.database_principals dp
     ON sp.grantee_principal_id = dp.principal_id
 WHERE dp.name = 'public'
 AND ao.name IS NOT NULL
 ORDER BY ao.name
```


In SQL Server 2016, there are 2,089 permissions stored in the **master** database granted to the public role. While it might seem daunting, they are all SELECT permissions and do not allow the user Stealth to make any changes in the master database. However, it is good practice to revoke some of the permissions based on the security policies in your organization. However, exercise caution while revoking them because some are required by users for normal operations in certain circumstances.

## Simplifying Role Review and Management

Rather than resorting to complex custom scripts to figure out the issues related to the public role in one instance at a time, consider using StealthAUDIT. This data access governance platform can enumerate all the SQL Server roles and privileges, including the SQL Server public roles and SQL Server public database roles, and produce detailed entitlement reports out of the box. It provides a single-pane-of-glass view into the public role across all the SQL Servers in your enterprise and helps you remediate any issues with a click of a button.

## Best Practices for the Public Role in SQL Server

We recommend the following best practices when it comes to public role in SQL Server:

- Do not grant any additional privileges to the public role outside of the default privileges, under any circumstance. If necessary, make use of a user-defined role.

- Do not modify the server-level permissions to the public role because doing so may prevent users from connecting to the database.
- Review the public permissions every time you upgrade your SQL Server since Microsoft often makes changes to the public role.

## FAQ

### What is public database role?

Permissions granted to the public database role are inherited by each database user.

### What permissions does the public role have in SQL Server?

Every SQL Server login belongs to the public server role. When a server principal hasn't been granted or denied specific permissions on a securable object, the user inherits the permissions granted on that object to the public role.

Joe Dibley
Security Researcher at Netwrix and member of the Netwrix Security Research Team. Joe is an expert in Active Directory, Windows, and a wide variety of enterprise software platforms and technologies, Joe researches new security risks, complex attack techniques, and associated mitigations and detections.