GRANT Object Permissions (Transact-SQL)

Applies To: SQL Server 2014, SQL Server 2016 Preview

Grants permissions on a table, view, table-valued function, stored procedure, extended stored procedure, scalar function, aggregate function, service queue, or synonym.

Applies to: SQL Server (SQL Server 2008 through current version), Azure SQL Database.

Transact-SQL Syntax Conventions

Syntax

```
GRANT <permission> [ ,...n ] ON
    [ OBJECT :: ][ schema_name ]. object_name [ ( column [ ,...n ] ) ]
    TO <database_principal> [ ,...n ]
    [ WITH GRANT OPTION ]
    [ AS <database_principal> ]

<permission> ::=
    ALL [ PRIVILEGES ] | permission [ ( column [ ,...n ] ) ]

<database_principal> ::= Database_user
    | Database_role
    | Application_role
    | Database_user_mapped_to_Windows_User
    | Database_user_mapped_to_Windows_Group
    | Database_user_mapped_to_certificate
    | Database_user_mapped_to_asymmetric_key
    | Database_user_with_no_login
```

Arguments

permission

Specifies a permission that can be granted on a schema-contained object. For a list of the permissions, see the Remarks section later in this topic.

ALL

Granting ALL does not grant all possible permissions. Granting ALL is equivalent to granting all ANSI-92 permissions applicable to the specified object. The meaning of ALL varies as follows:

Scalar function permissions: EXECUTE, REFERENCES.

Table-valued function permissions: DELETE, INSERT, REFERENCES, SELECT, UPDATE.

Stored procedure permissions: EXECUTE.

Table permissions: DELETE, INSERT, REFERENCES, SELECT, UPDATE.

View permissions: DELETE, INSERT, REFERENCES, SELECT, UPDATE.

PRIVILEGES

Included for ANSI-92 compliance. Does not change the behavior of ALL.

column

Specifies the name of a column in a table, view, or table-valued function on which the permission is being granted. The parentheses ( ) are required. Only SELECT, REFERENCES, and UPDATE permissions can be granted on a column. *column* can be specified in the permissions clause or after the securable name.

> **Caution**
>
> A table-level DENY does not take precedence over a column-level GRANT. This inconsistency in the permissions hierarchy has been preserved for backward compatibility.

ON [OBJECT ::] [*schema_name*] . *object_name*

Specifies the object on which the permission is being granted. The OBJECT phrase is optional if *schema_name* is specified. If the OBJECT phrase is used, the scope qualifier (::) is required. If *schema_name* is not specified, the default schema is used. If *schema_name* is specified, the schema scope qualifier (.) is required.

TO <database_principal>

Specifies the principal to which the permission is being granted.

WITH GRANT OPTION

Indicates that the principal will also be given the ability to grant the specified permission to other principals.

AS <database_principal>

Specifies a principal from which the principal executing this query derives its right to grant the permission.

Database_user

Specifies a database user.

Database_role

Specifies a database role.

Application_role

Specifies an application role.

Database_user_mapped_to_Windows_User
Specifies a database user mapped to a Windows user.

Database_user_mapped_to_Windows_Group
Specifies a database user mapped to a Windows group.

Database_user_mapped_to_certificate
Specifies a database user mapped to a certificate.

Database_user_mapped_to_asymmetric_key
Specifies a database user mapped to an asymmetric key.

Database_user_with_no_login
Specifies a database user with no corresponding server-level principal.

Remarks

Important

A combination of ALTER and REFERENCE permissions in some cases could allow the grantee to view data or execute unauthorized functions. For example: A user with ALTER permission on a table and REFERENCE permission on a function can create a computed column over a function and have it be executed. In this case the user would also need SELECT permission on the computed column.

Information about objects is visible in various catalog views. For more information, see Object Catalog Views (Transact-SQL).

An object is a schema-level securable contained by the schema that is its parent in the permissions hierarchy. The most specific and limited permissions that can be granted on an object are listed in the following table, together with the more general permissions that include them by implication.

Object permission	Implied by object permission	Implied by schema permission
ALTER	CONTROL	ALTER
CONTROL	CONTROL	CONTROL
DELETE	CONTROL	DELETE
EXECUTE	CONTROL	EXECUTE
INSERT	CONTROL	INSERT

RECEIVE	CONTROL	CONTROL
REFERENCES	CONTROL	REFERENCES
SELECT	RECEIVE	SELECT
TAKE OWNERSHIP	CONTROL	CONTROL
UPDATE	CONTROL	UPDATE
VIEW CHANGE TRACKING	CONTROL	VIEW CHANGE TRACKING
VIEW DEFINITION	CONTROL	VIEW DEFINITION

Permissions

The grantor (or the principal specified with the AS option) must have either the permission itself with GRANT OPTION, or a higher permission that implies the permission being granted.

If you are using the AS option, the following additional requirements apply.

AS	**Additional permission required**
Database user	IMPERSONATE permission on the user, membership in the db_securityadmin fixed database role, membership in the db_owner fixed database role, or membership in the sysadmin fixed server role.
Database user mapped to a Windows login	IMPERSONATE permission on the user, membership in the db_securityadmin fixed database role, membership in the db_owner fixed database role, or membership in the sysadmin fixed server role.
Database user mapped to a Windows Group	Membership in the Windows group, membership in the db_securityadmin fixed database role, membership in the db_owner fixed database role, or membership in the sysadmin fixed server role.
Database user mapped to a certificate	Membership in the db_securityadmin fixed database role, membership in the db_owner fixed database role, or membership in the sysadmin fixed server role.
Database user mapped to an asymmetric key	Membership in the db_securityadmin fixed database role, membership in the db_owner fixed database role, or membership in the sysadmin fixed server role.
Database user not	IMPERSONATE permission on the user, membership in the db_securityadmin fixed

mapped to any server principal	database role, membership in the db_owner fixed database role, or membership in the sysadmin fixed server role.
Database role	ALTER permission on the role, membership in the db_securityadmin fixed database role, membership in the db_owner fixed database role, or membership in the sysadmin fixed server role.
Application role	ALTER permission on the role, membership in the db_securityadmin fixed database role, membership in the db_owner fixed database role, or membership in the sysadmin fixed server role.

Examples

A. Granting SELECT permission on a table

The following example grants `SELECT` permission to user `RosaQdM` on table `Person.Address` in the `AdventureWorks2012` database.

```
USE AdventureWorks2012;
GRANT SELECT ON OBJECT::Person.Address TO RosaQdM;
GO
```

B. Granting EXECUTE permission on a stored procedure

The following example grants `EXECUTE` permission on stored procedure `HumanResources.uspUpdateEmployeeHireInfo` to an application role called `Recruiting11`.

```
USE AdventureWorks2012;
GRANT EXECUTE ON OBJECT::HumanResources.uspUpdateEmployeeHireInfo
    TO Recruiting11;
GO
```

C. Granting REFERENCES permission on a view with GRANT OPTION

The following example grants `REFERENCES` permission on column `BusinessEntityID` in view `HumanResources.vEmployee` to user `Wanida` with `GRANT OPTION`.

```
USE AdventureWorks2012;
GRANT REFERENCES (BusinessEntityID) ON OBJECT::HumanResources.vEmployee
    TO Wanida WITH GRANT OPTION;
GO
```

D. Granting SELECT permission on a table without using the OBJECT phrase

The following example grants `SELECT` permission to user `RosaQdM` on table `Person.Address` in the `AdventureWorks2012` database.

```
USE AdventureWorks2012;
GRANT SELECT ON Person.Address TO RosaQdM;
GO
```

E. Granting SELECT permission on a table to a domain account

The following example grants `SELECT` permission to user `AdventureWorks2012\RosaQdM` on table `Person.Address` in the `AdventureWorks2012` database.

```
USE AdventureWorks2012;
GRANT SELECT ON Person.Address TO [AdventureWorks2012\RosaQdM];
GO
```

F. Granting EXECUTE permission on a procedure to a role

The following example creates a role and then grants `EXECUTE` permission to the role on procedure `uspGetBillOfMaterials` in the `AdventureWorks2012` database.

```
USE AdventureWorks2012;
CREATE ROLE newrole ;
GRANT EXECUTE ON dbo.uspGetBillOfMaterials TO newrole ;
GO
```

See Also

- DENY Object Permissions (Transact-SQL)
- REVOKE Object Permissions (Transact-SQL)
- Object Catalog Views (Transact-SQL)
- Permissions (Database Engine)
- Principals (Database Engine)
- Securables
- sys.fn_builtin_permissions (Transact-SQL)
- HAS_PERMS_BY_NAME (Transact-SQL)
- sys.fn_my_permissions (Transact-SQL)

Community Additions