

Advanced Databases, I3306

Advanced Databases, In3306: Security and Authorization

UL-Nabatieh

Octobre 30, 2018



Outline



- 1 Security and Authorization
- 2 Authorization in TSQL



Introduction



- Database security refers to protection from malicious access ;
- Some database-system users may be authorized to access only a limited portion of the database;
- Other users may be allowed to issue queries, but may be forbidden to modify the data.



Authorization



Forms of **authorization** on parts of the database:

- **Read**: allows reading, but not modification of data;
- **Insert**: allows insertion of new data, but not modification of existing data;
- **Update**: allows modification, but not deletion of data;
- **Delete**: allows deletion of data.
- **Create table**: allows table creation
- ⋮

We may assign the user all, none, or a combination of these types of authorization





Granting of Privileges

A user who has been **granted some form of authorization** may be allowed to **pass on** this authorization to other users.

Authorization graph

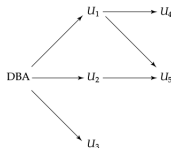
- **Nodes**: users;
- **Arcs**: $U_i \rightarrow U_j$ if user U_i grants authorization to U_j ;
- The **root** of the graph is the **database administrator**.





Authorization graph: example

- The database administrator grants update authorization on **TableA** to users U_1 , U_2 , and U_3 , who may in turn pass on this authorization to other users;
- User U_5 is granted authorization by both U_1 and U_2 ;
- User U_4 is granted authorization by only U_1 .

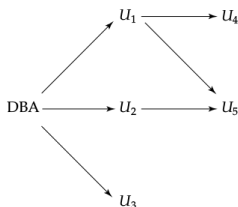


A user has an authorization if and only if there is a path from the root of the authorization graph.





Authorization graph: example

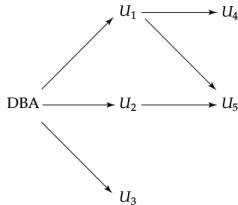


- Suppose that the database administrator decides to revoke the authorization of user U_1 ;
- Since U_4 has authorization from U_1 , that authorization should be revoked as well;
- U_5 was granted authorization by both U_1 and U_2 . Since the database administrator did not revoke update authorization on **TableA** from U_2 , U_5 retains update authorization on **TableA**.
- If U_2 eventually revokes authorization from U_5 , then U_5 loses the authorization.





Authorization graph: example



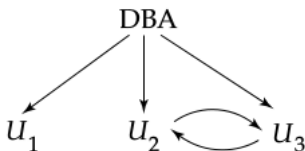
A user has an authorization if and only if there is a path from the root of the authorization graph.



Attempt to defeat authorization revocation



A pair of devious users might attempt to **defeat the rules for revocation** of authorization by granting **authorization to each other**.

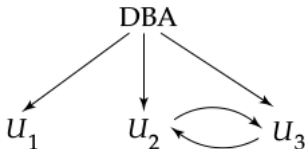




Attempt to defeat authorization revocation

If the database administrator revokes authorization:

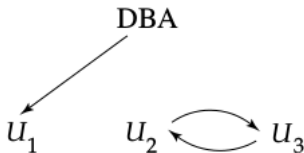
- From U_2 , U_2 retains authorization through U_3 ;
- From U_3 , U_3 retains authorization through U_2 .





Attempt to defeat authorization revocation

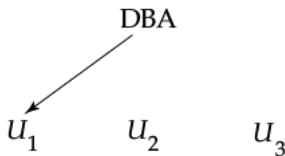
However, when the database administrator **revokes** authorization from U_2 and U_3 , the **edges from U_3 to U_2 and from U_2 to U_3 are no longer part of a path starting with the database administrator.**



Attempt to defeat authorization revocation



- We require that all edges in an authorization graph be part of some path originating with the database administrator ;
- The edges between U_2 and U_3 are deleted.



Notion of roles



Example: university

- Each **instructor** must have the same types of authorizations to the same set of relations;
- Each **teacher assistant** must have the same types of authorizations to the same set of relations;

Whenever a new instructor is appointed, he will have to be given all these authorizations individually!





Notion of roles

Example: university

- Specify the authorizations that every instructor is to be given, and to separately identify which database users are instructors;
- When a new person is hired as an instructor, a user identifier must be allocated to him, and he must be identified as an instructor;
- Individual permissions given to instructors need not be specified again.



Notion of roles



Role

- A set of roles is created in the database;
- Authorizations can be granted to roles, in exactly the same fashion as they are granted to individual users;
- Each database **user is granted a set of roles** (which may be empty) that he or she is authorized to perform.



Outline



- 1 Security and Authorization
- 2 Authorization in TSQL





Privileges in TSQL

The **grant** command is used to provide access or privileges on the database objects to the users.

```
1 grant privilege_name
2 on object_name
3 to {user_name | PUBLIC | role_name}
4 [with grant option];
```

- **privilege_name**: the access right or privilege granted to the user, i.g. SELECT, UPDATE;
- **object_name**: the name of an database object, e.g. TABLE, VIEW;
- **user_name**: the name of the user to him an access right is being granted;
- **PUBLIC**: used to grant access rights to all users;
- **role_name**: the role to it an access right is being granted;
- **WITH GRANT OPTION**: allows a user to grant access rights to other users.



Privileges in TSQL



```
1 grant privilege_name
2 on object_name
3 to {user_name | PUBLIC | role_name}
4 [with grant option];
```

- Granting a **privilege on a view does not imply granting any privileges on the underlying relations**;
- The **grantor of the privilege must already hold the privilege** on the specified item (or be the database administrator)



Privileges in TSQL



- **select**: allows read access to relation, or the ability to query using the view;
- **insert**: the ability to insert tuples;
- **update**: the ability to update using the SQL update statement;
- **delete**: the ability to delete tuples;
- **:**
- **all**: used as a short form for all the allowable privileges.





Examples

- Grant SELECT, INSERT, UPDATE, and DELETE privileges on a table called employees to the user lionel:

```
1 grant select , insert , update , delete on employees to lionel;
```

- Grant all the permissions (ie: SELECT, INSERT, UPDATE and DELETE) to a user named lionel:

```
1 grant all on employees to lionel;
```

- Grant only SELECT access on the employees table to all users:

```
1 grant select on employees to public;
```

Examples



- Grant users lionel and marc update authorization on the salary attribute of the employees relation

```
1 grant update(salary) on employees to lionel, marc
```

- Grant user marc create table authorization

```
1 grant create table to marc;
```





Revoking Authorization in SQL

- The **revoke** statement is used to revoke authorization;

```
1 revoke privilege_name  
2 on object_name  
3 from {user_name | PUBLIC | role_name}  
4 [cascade]
```

- If the same privilege was **granted twice to the same user by different grantees**, the user **may retain** the privilege after the revocation;
- All privileges that depend on the privilege being revoked are also revoked.



- La commande REVOKE retire des droits précédemment attribués à un ou plusieurs rôles. Le mot clé PUBLIC fait référence au groupe implicitement défini de tous les rôles.
- Notez qu'un rôle possède la somme des droits qui lui ont été donnés directement, des droits qui ont été donnés à un rôle dont il est membre et des droits donnés à PUBLIC. Du coup, par exemple, retirer les droits de SELECT à PUBLIC ne veut pas nécessairement dire que plus aucun rôle n'a le droit de faire de SELECT sur l'objet : ceux qui en avaient obtenu le droit directement ou via un autre rôle l'ont toujours.
- Si GRANT OPTION FOR est précisé, seul l'option de transmission de droit (grant option) est supprimée, pas le droit lui-même. Sinon, le droit et l'option de transmission de droits sont révoqués.



- Si un utilisateur détient un privilège avec le droit de le transmettre, et qu'il l'a transmis à d'autres utilisateurs, alors les droits de ceux-ci sont appelés des droits dépendants. Si les droits ou le droit de transmettre du premier utilisateur sont supprimés, et que des droits dépendants existent, alors ces droits dépendants sont aussi supprimés si l'option CASCADE est utilisée, sinon la suppression de droits est refusée.
- Cette révocation réursive n'affecte que les droits qui avaient été attribués à travers une chaîne d'utilisateurs traçable jusqu'à l'utilisateur qui subit la commande REVOKE.
- Du coup, les utilisateurs affectés peuvent finalement garder le droit s'il avait aussi été attribué via d'autres utilisateurs.





Examples

- Revoke delete privileges on a table called employees from a user named marc:

```
1 revoke delete on employees from marc
```

- Revoke all permissions (ie: SELECT, INSERT, UPDAT and DELETE) on a table for a user named marc:

```
1 revoke all on employees from marc
```



Examples



- Revoke select privileges on a table called employees from all users:

```
1 revoke select on employees from public
```

- Revoke update permission on the column salary in the relation employees from the user lionel with CASCADE:

```
1 revoke update(salary) on employees from lionel CASCADE;
```



Creating roles



■ To create a role:

```
1 create role name_of_role
2 create role instructor
3 create role teaching_assistant
```

■ Privileges can be granted to roles:

```
1 grant select on TableA to instructor
```



Creating roles



■ To create a role:

```
1 create role name_of_role
2 create role instructor
3 create role teaching_assistant
```

■ Privileges can be granted to roles:

```
1 grant select on TableA to instructor
```

■ Roles can be granted to users, as well as to other roles:

```
1 grant teaching_assistant to John
2 grant teaching_assistant to instructor
```





Creating roles

■ To create a role:

```
1 create role name_of_role
2 create role instructor
3 create role teaching_assistant
```

■ Privileges can be granted to roles:

```
1 grant select on TableA to instructor
```

■ Roles can be granted to users, as well as to other roles:

```
1 grant teaching_assistant to John
2 grant teaching_assistant to instructor
```

■ Chain of roles

```
1 create role dean
2 grant teaching_assistant to instructor
3 grant instructor to dean
```

Login



- Access to the server is granted via logins;
- There are two main categories of logins:
 - ① Windows authenticated logins
 - ② SQL Server authenticated logins



Login/User: example



```
1 create login login1 with password='1234'  
2 create login login2 with password='1234'  
3 create login login3 with password='1234'  
  
5 create user user1 for login login1  
6 create user user2 for login login2  
7 create user user3 for login login3
```



User/Role



To grant a role to a user:

```
1 create role role1
2 create role role2
3 create role role3
4 alter role role1 add member user1
5 alter role role2 add member user2
6 alter role role3 add member user3
```

To remove a role from a user:

```
1 alter role role1 drop member user1
2 alter role role2 drop member user2
3 alter role role3 drop member user3
```





Example

```
1 select CURRENT_USER
2 — grant permissions of reading/inserting/updating course table
3 grant select, insert, update, delete on Course to role1 with grant option;
4 — user1 context
5 execute as user = 'user1';
6 select CURRENT_USER
7 — grant to users mapped to role2, permissions from user1 permissions
8 grant select, insert on Course to role2 with grant option;
9 — end user1 context
10 revert;
11 select CURRENT_USER
12 — user2 context
13 execute as user = 'user2';
14 select CURRENT_USER
15 — grant to users mapped to role3, select permissions from user2 permissions
16 grant select on Course to role3 with grant option;
17 select * from INFORMATION_SCHEMA.TABLE_PRIVILEGES
```

Remarks



Revert: switches the execution context back to the caller of the last EXECUTE AS statement.



Some fixed-database roles



- **db_datareader**: members can **read all data** from all user tables
- **db_datawriter**: members can **add, delete, or change** data in all user tables
- **db_denydatareader**: members **cannot read any data** in the user tables within a database
- **db_denydatawriter**: members **cannot add, modify, or delete any data** in the user tables within a database.

Remark: the permissions assigned to the fixed-database roles cannot be changed.



Some indications



Error

Error message: (provider: Shared Memory Provider, error: 0 - No process is on the other end of the pipe.)

Solution

You should enable the Server authentication mode to mixed mode as following:
In SQL Studio, select YourServer → Property → Security → Select SqlServer and Window Authentication mode. Restart the server



Some indications



To change security authentication mode:

- 1 In SQL Server Management Studio Object Explorer, right-click the server, and then click Properties;
- 2 On the Security page, under Server authentication, select the new server authentication mode, and then click OK;
- 3 In the SQL Server Management Studio dialog box, click OK to acknowledge the requirement to restart SQL Server;
- 4 In Object Explorer, right-click your server, and then click Restart. If SQL Server Agent is running, it must also be restarted.

