Laboratorium 5 – MS SQL Server 2019

Temat: Bezpieczeństwo w bazie danych
Opracowanie: A.Dydejczyk,

Dwa poziomy zabezpieczenia dostępu do serwera SQL.
Pierwszy :
Login – UserID, password
Drugi:
Permission Validation – User, Group, Role

Rys. 1 Poziomy zabezpieczeń do serwera bazy danych i do baz danych

Plan zajęć

1. Uwierzytelnianie w systemie MS SQL Sever 2019 : Windows i mixed mode
    a. Tworzenie kont w systemie Windows i użycie Windows Authentication
    b. Tworzenie kont (login) w MS SQL Server 2019 i wykorzystanie mixed mode
    c. Tworzenie grup kont w systemie Windows i przypisanie login'ów SQL
    d. Tworzenie ról w MS SQL Server 2019

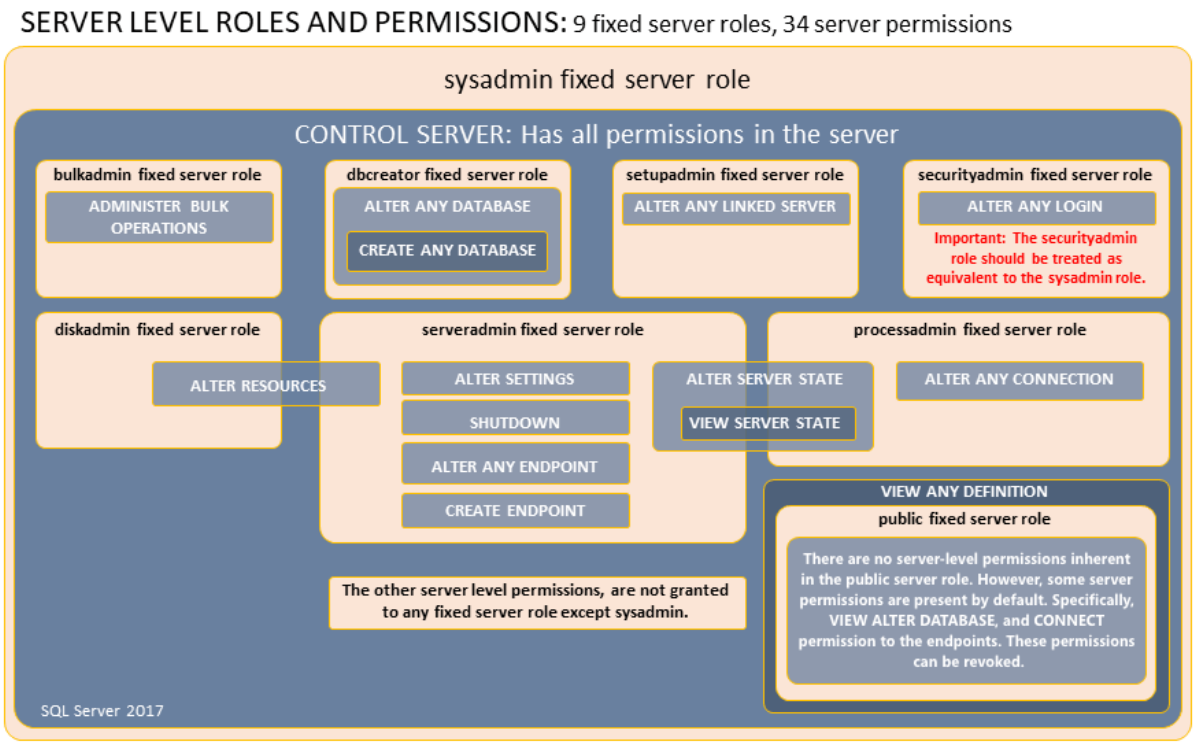
2. Nadawanie uprawnień do baz danych
    a. Wbudowane role dla baz danych
    b. Konto aplikacji
    c. Polecenia GRANT REVOKE DENY
    d. Typy uprawnień do obiektów (tabel etc.)
    e. Tworzenie ról (GUI i kod T-SQL)
    f. Nadawanie i zmiana uprawnień do obiektów baz – GUI i kod T-SQL
    g. Perspektywy – użycie w sensie bezpieczeństwa

3. Schema – rola w organizacji obiektów baz danych
    a. Tworzenie schema
    b. Nadawanie uprawnień do schema

1. Uprawnienia do serwera MS SQL Server i do baz danych

W ramach SQL Server'a został zrealizowany system bezpieczeństwa oparty o role i uprawnienia do serwera i do poszczególnych baz danych uruchomionych w ramach serwera. Na rys. 2 przedstawiono role i ich uprawnienia na poziomie serwera MS SQL Server. Pełny opis uprawnień i ról do serwera dostępny jest pod poniższym adresem:

https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/server-level-roles?view=sql-server-ver16



Rys 2. Zabezpieczenia na poziomie serwera

Kolejnym etapem zabezpieczenia w ramach serwera są uprawnienia z poziomu serwera w dostępie do baz danych poprzez przyznanie im określonych ról systemowych. W ramach bazy danych można przydzielać uprawnienia użytkownikom i rolom poprzez odpowiednie polecenia języka SQL.

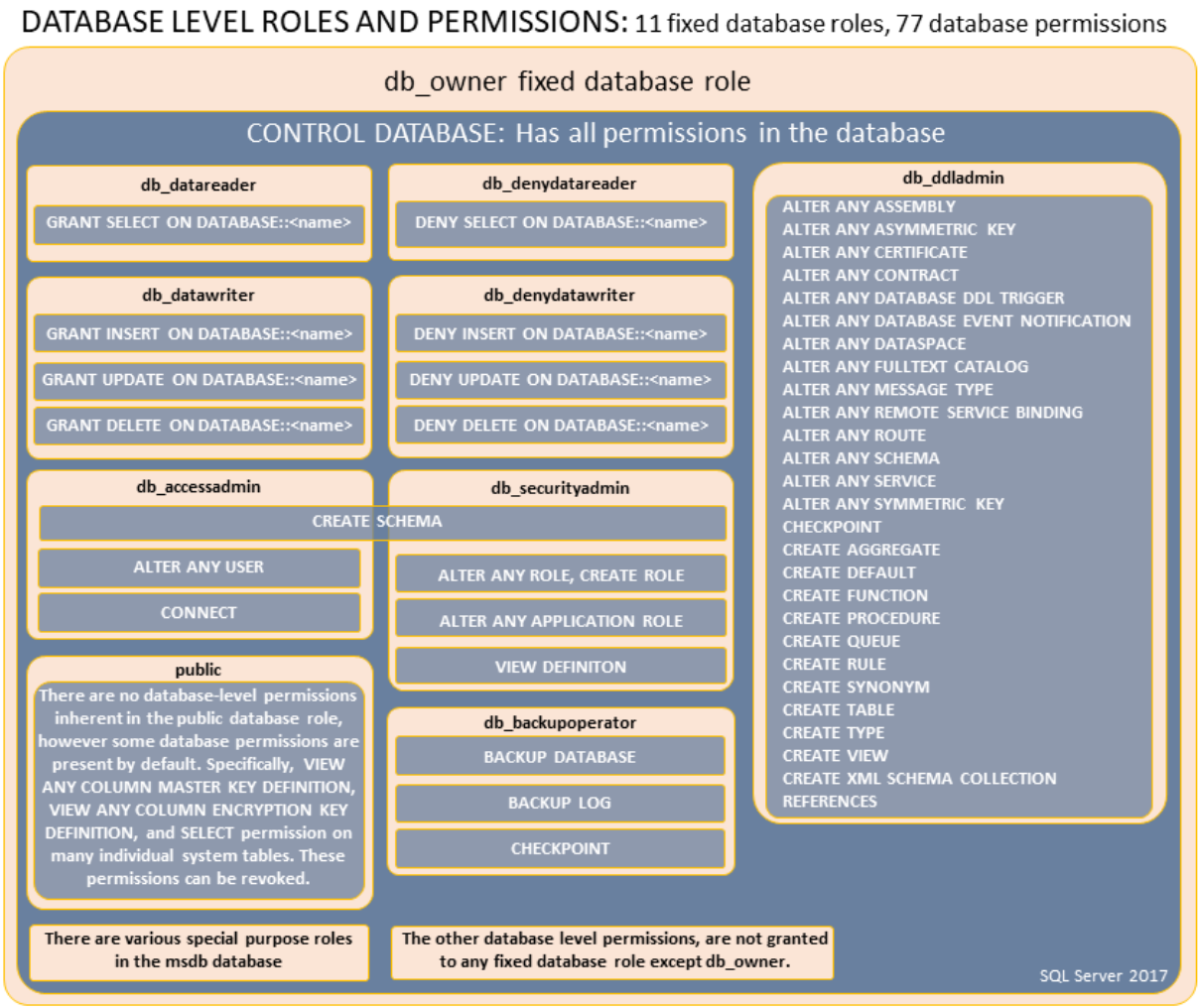
Użytkownik musi mieć nadane prawo do bazy, nie wystarczy login do serwera (rys.1). W SQL Server został zdefiniowany zakres ról, które pozwalają na przydzielenie właściwych uprawnień użytkowników (principals) do obiektów (securables). Istnieje także możliwość zdefiniowania własnych ról.

Pod pojęciami principals i securables w ramach bazy danych rozumiemy:

- **Securables** – obiekty podlegające zabezpieczeniom:
  - server
  - database
  - schema
  - tabele, widoki i inne

- **Principals** – obiekty (user i role) którym nadajemy uprawnienia (GRANT) do securables.

Uprawnienia w dostępie do baz danych w ramach SQL Server'a przedstawiono na rys.3 a pełny opis uprawnień i ról dostępny jest pod poniższym adresem:
https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/database-level-roles?view=sql-server-ver16



Rys.3 Role i uprawnienia do baz danych w MS SQL Server

Predefiniowane role to:

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. (In SQL Database and Azure Synapse, some maintenance activities require server-level permissions and cannot be performed by **db_owners**.)
db_securityadmin	Members of the **db_securityadmin** fixed database role can modify role membership for custom roles only and manage permissions. Members of this role can potentially elevate their privileges and their actions should be monitored.
db_accessadmin	Members of the **db_accessadmin** fixed database role can add or remove access to the database for Windows logins, Windows groups, and SQL Server logins.
db_backupoperator	Members of the **db_backupoperator** fixed database role can back up the database.
db_ddladmin	Members of the **db_ddladmin** fixed database role can run any Data Definition Language (DDL) command in a database. Members of this role can potentially elevate their privileges by manipulating code that may get executed under high privileges and their actions should be monitored.
db_datawriter	Members of the **db_datawriter** fixed database role can add, delete, or change data in all user tables. In most use cases this role will be combined with **db_datareader** membership to allow reading the data that is to be modified.
db_datareader	Members of the **db_datareader** fixed database role can read all data from all user tables and views. User objects can exist in any schema except *sys* and *INFORMATION_SCHEMA*.
db_denydatawriter	Members of the **db_denydatawriter** fixed database role cannot add, modify, or delete any data in the user tables within a database.
db_denydatareader	Members of the **db_denydatareader** fixed database role cannot read any data from the user tables and views within a database.

Opis uprawnień udostępnianych w ramach MS SQL Server'a można również prześledzić w ramach opracowania dostępnego na poniższej stronie.

https://learn.microsoft.com/en-us/sql/relational-databases/security/permissions-database-engine?source=recommendations&view=sql-server-ver16

Diagram przedstawiający pełny zakres oddziaływania uprawnień na wszystkie obiekty w bazie danych można prześledzić na diagramie, który dostępny jest pod adresem:

https://aka.ms/sql-permissions-poster

2. Uwierzytelnianie i autoryzacja w systemie MS SQL Server 2019

Dostęp do serwera (logowanie) jest realizowany w ramach SQL Server na dwa sposoby:

- windows authentication
  (np. dla użytkowników domeny windows)
  Windows Authentication – logowanie z konta użytkownika Windows, tożsamość potwierdzona przez Windows. W tym trybie SQL Server nie sprawdza tożsamości. Jest to bezpieczniejszy sposób logowania, wykorzystuje protokół Kerberos.

- mixed mode
  (np. dla aplikacji z logowaniem się umieszczonym w kodzie, z dostępem przez witryny web użytkowników nie będących uczestnikami domeny)
  Mixed Mode – możliwe logowanie z konta użytkownika Windows albo poprzez autentykację SQL Servera. W tym drugim przypadku trzeba stworzyć nazwę użytkownika i hasło, dane te są przechowywane w SQL Serverze. Użytkownicy łączący się z serwerem w ten sposób za każdym razem muszą podać nazwę i hasło. Jest to sposób mniej bezpieczny – nie używany protokołu Kerberos, ale za to możliwa współpraca z aplikacjami sieciowymi, w których użytkownicy tworzą własne loginy.

SQL Server jako usługa pracuje na koncie Windows:

- **lokalnym – local acco**unt (to konto ma więcej uprawnień niż to jest wymagane, jest to konto administratora)
- **domain user account** – zalecane dla instalacji domenowych

Poprzez logowanie do bazy danych użytkownik jest weryfikowany dwufazowo:

- I etap - uwierzytelnienie: sprawdzamy login i hasło użytkownika;
- II etap - autoryzacja: sprawdzamy uprawnienia do bazy danych (User, Group, Role).

Użytkownicy przypisani są do ról. Każdy obiekt typu securables posiada właściciela, co także ma wpływ na uprawnienia do obiektów securables.

Obiekty principals mają do dyspozycji polecenia SQL dwóch typów:

- **DDL:**
  CREATE, DROP, ALTER, CONTROL, TAKE OWNERSHIP, and GRANT
  Tworzenie, usuwanie, nadawanie uprawnień do obiektów baz danych i samej bazy danych

- **DML**
  SELECT, INSERT, UPDATE, DELETE, EXECUTE and EXECUTE AS

Prezentacja:

Sprawdzamy sposób uwierzytelnienia (autentyfikacji):

EXEC xp_loginconfig 'login mode'

Sprawdzamy "login'y" systemowe

SELECT * FROM sys.server_principals

Sprawdzamy niepoprawne wpisy (orphaned logins), czy użytkownicy usunięci z systemu operacyjnego zostali również usunięci z serwera MS SQL Server.

EXEC sp_validatelogins

Sposób autentykacji można sprawdzić na trzy sposoby:

- Wykorzystując procedurę rozszerzoną xp_loginconfig -
  EXEC xp_loginconfig 'login mode'

- W SSMS, prawy przycisk myszki na nazwie serwera a następnie
  => Properties => Security => Server authentication

- Sprawdzając wpis w rejestrze:
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MicrosoftSQLServer\<nazwa_instancji>\MSSQLServer\LoginMode '0' oznacza Windows authentication, a '1' Mixed mode

Zmiana sposobu autentykacji jest możliwa poprzez jeden z dwóch ostatnich sposobów.

3. Uprawnienia do baz danych

Specjalną funkcję pełni predefiniowana rola **public** – wszyscy użytkownicy bazy danych otrzymują uprawnienia tej roli, zwykle do przeglądania tabel systemowych i obiektów bazy master oraz wykonywania poleceń, które nie wymagają uprawnień (np. PRINT). Do tej roli nie można dodawać użytkowników ani grup, nie można jej też usunąć.

Fixed Database Role	Rights Granted	Permits
db_owner	CONTROL	Any operation in the database, including deleting it
db_securityadmin	ALTER ANY APPLICATION ROLE ALTER ANY ROLE CREATE SCHEMA VIEW DEFINITION	Security management
db_accessadmin	ALTER ANY USER CONNECT (With GRANT) CREATE SCHEMA	DB Login management

db_backupoperator	BACKUP DATABASE BACKUP LOG CHECKPOINT	Backup management
db_datareader	SELECT	Select data from any table
db_datawriter	DELETE INSERT UPDATE	Add or edit data in any table
db_ddladmin	ALTER ANY ASSEMBLY ALTER ANY CERTIFICATE ALTER ANY CONTRACT ALTER ANY EVENT NOTIFICATION ALTER ANY DATASPACE ALTER ANY FULLTEXT CATALOG ALTER ANY MESSAGE TYPE ALTER ANY REMOTE SERVICE BINDING ALTER ANY ROUTE ALTER ANY SCHEMA ALTER ANY SERVICE ALTER ANY SYMMETRIC KEY ALTER ANY TRIGGER ALTER ANY XML SCHEMA COLLECTION CHECKPOINT CREATE AGGREGATE CREATE ASSEMBLY CREATE CONTRACT CREATE DEFAULT CREATE FUNCTION CREATE MESSAGE TYPE CREATE PROCEDURE CREATE QUEUE CREATE REMOTE SERVICE BINDING CREATE ROUTE CREATE RULE CREATE SCHEMA CREATE SERVICE CREATE SYMMETRIC KEY CREATE SYNONYM CREATE TABLE CREATE TYPE CREATE VIEW CREATE XML SCHEMA COLLECTION REFERENCES	All DDL operations
db_denydatareader	Denied: SELECT	Prevents the principal from reading data in any table
db_denydatawriter	Denied: DELETE Denied: INSERT Denied: UPDATE	Prevents the principal from writing data in any table

Testowanie uprawnień do bazy danych MS SQL Server

Zadanie 1. Uwierzytelnienie poprzez konta w systemie Windows

Tworzymy grupy w systemie Windows (grupa1, grupa2) oraz użytkowników (tester1, tester2 i tester 3)

Grupa	**Użytkownicy**
grupa1	tester1, tester2, tester3
grupa2	tester3, tester4, tester5

Dla jednej grupy zezwalamy na logowanie (grant) a dla drugiej (deny).
Przydzielamy im uprawnienia do domyślnej (default) bazy danych – AdvantureWorks2019.

Wynikiem ćwiczeń mają być pliki z poleceniami T-SQL (możliwie w pełnej postaci) realizujących zadania. Do realizacji zadań można wykorzystać opcję Script as
Na każdym etapie po wykonaniu weryfikujemy ustawienia (szczególnie domyślne) w właściwościach danego obiektu.

Dodatkowo należy odpowiedzieć na pytania:

Jakie jest efektywne uprawnienie dla użytkownika należącego do obu grup ?
Jakie są uprawnienia (jaka rola) do domyślnej bazy danych ?

Polecenia T-SQL przydatne do realizacji zadania.

Uwierzytelnienie: windows authentication

```
CREATE LOGIN [nazwa_domeny\tester1] FROM WINDOWS WITH
DEFAULT_DATABASE=[AdventureWorks2019], DEFAULT_LANGUAGE=[us_english]
```

lub przez wbudowaną procedurę

```
EXEC sp_grantlogin [@loginame=] 'login'
EXEC sp_grantlogin 'nazwa_domeny\tester1'
```

usunięcie login

```
EXEC sp_revokelogin 'nazwa_domeny\tester1'
```

usunięcie uprawnienia do połączenia z bazą danych

```
EXEC sp_denylogin 'nazwa_domeny\tester1'
```

Przypisanie domyślnej bazy:

```
EXEC sp_defaultdb 'kuba', ' AdventureWorks '      ( wersja niezalecana )
```

ALTER LOGIN [nazwa_domeny\tester1] WITH DEFAULT_DATABASE = AdventureWorks - mixed mode

Uwierzytenienie: Login utworzony w SQL 2019

```
CREATE LOGIN [tester1] WITH PASSWORD=N'random_password',
DEFAULT_DATABASE=[pubs], DEFAULT_LANGUAGE=[polski],
CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO

ALTER LOGIN [tester1] DISABLE
```

Formy niezalecane.

```
sp_addlogin 'login', 'password', 'defaultdatabase', 'defaultlanguage', 'sid',
'encryption_option'

EXEC sp_addlogin 'tester1', 'supertajne', 'AventureWorks'
EXEC sp_droplogin  'tester1'
EXEC sp_password 'myoldpassword', 'mynewpassword', 'tester1'
```

Komentarz:

SQL Server automatycznie tworzy użytkownika ‘BUILTINS/Administrators’, do którego przypisuje wszystkich użytkowników grup Administratorzy systemu Windows,
A dla mixed mode tworzy także użytkownika sa i przypisuje mu rolę sysadmin.

Server Roles
SQL Server posiada predefiniowane, ustalone role. Pozwalają one na wykonywanie zadań administracyjnych. Użytkownik może należeć do wielu ról.

W ramach zadania należy również opracować wyzwalacz obsługujący zdarzenie (LOGON), który jest wyzwalany kiedy ustalana jest sesja logującego użytkownika (lab. 04)

```
CREATE TRIGGER nazwa_wyzwalacza
ON ALL SERVER
[ WITH <opcje_wyzwalacza_logon > [ ,...n ] ]
{ FOR | AFTER }
LOGON
AS { wyrażenie_sql [ ; ] [ ,...n ] }
< opcje_wyzwalacza_logon > ::=
        [ ENCRYPTION ]
        [ EXECUTE AS Clause ]
```

Pomocne linki:

- https://learn.microsoft.com/en-us/sql/relational-databases/triggers/logon-triggers?view=sql-server-ver16
- https://www.mssqltips.com/sqlservertip/6103/sql-server-logon-trigger-examples/

Zadanie 2. Testowanie uprawnień na poziomie bazy danych

Tworzymy grupę w systemie Windows (grupa3) oraz użytkowników (tester6 i tester7).

Przydzielamy im uprawnienia do domyślnej (default) bazy danych (master).
Przydzielamy ich do ról: dbcreator i serveradmin.
Testujemy zakres operacji, które można wykonać logując się z tymi kontami.
Uwaga: testy w postaci skryptów T-SQL z komentarzem co wynika z wykonania tego testu.

Przydatne polecenia T-SQL do zrealizowania zadania.

Przydział określonego login'u do roli.

```
sp_addsrvrolemember
[ @loginame = ] 'login',
[ @rolename = ] 'role'

EXEC sp_addsrvrolemember 'nazwa_domeny\tester4', 'sysadmin'
EXEC sp_dropsrvrolemember 'nazwa_domeny\tester4', 'sysadmin'
```

W celu otrzymania informacji o rolach wykorzystujemy poniższe polecenia.

```
-- informacje o użytkownikach przypisanych do ról na poziomie serwera
select * from sys.server_role_members

select * from sys.server_principals

select * from sys.server_principals
        JOIN sys.server_role_members ON principal_id = member_principal_id

-- sprawdzenie czy dany (lub bieżący) login należy od roli
IS_SRVROLEMEMBER ('role' ,'login']))
```

Zadanie 3. Uprawnienia do obiektów baz danych

Efektywne uprawnienia użytkownika (user) określone są przez przynależność do roli (lub wielu ról). SQL Serwer obsługuje w pełni mechanizm przydzielania uprawnień SELECT, INSERT, EXECUTE i wszystkich pozostałych stosownych (mających sens) do poszczególnych obiektów typu tabela, widok, funkcja czy procedura przechowywana. Efektywne uprawnienia są określone przez mechanizm, w którym jest zaangażowana hierarchia.

W kolejności ważności

1. Rola sysadmin
2. Deny na obiekcie lub rola db_denydatareader lub rola db_denydatawriter
3. Grant na obiekcie lub rola własności obiektu lub rola db_datareader lub rola db_datewriter
4. Revoke na obiekcie

W ramach zadania wykorzystamy użytkowników i grupy utworzone w zadaniu 1.

Grupa	Użytkownicy
grupa1	tester1, tester2, tester3
grupa2	tester3, tester4, tester5

Przydzielamy dla grup uprawnienia roli: db_datawriter.
Użytkownik tester3 ma uprawnienia sysadmin.
Użytkownicy tester2 i tester4 posiadają tylko uprawnienia do SELECT.

Czy można skonstruować rolę standardową rozwiązującą ten problem i oczywiście dokumentacja w formie skryptów.

Polecenia T-SQL przydatne do zrealizowania zadania.

```
GRANT Permission, Permission
ON Object
TO User/role, User/role

WITH GRANT OPTION

CREATE ROLE nazwa_roli [ AUTHORIZATION posiadacz(owner) ]
DROP ROLE nazwa_roli
ALTER ROLE nazwa_roli WITH NAME =
```

Niezalecane rozwiązania:

```
EXEC sp_addrole 'Manager'
EXEC sp_addrolemember 'Manager', Ant
```

Przykłady:

```
GRANT Select, Update ON Person to Guest, Ant
GRANT Select ON Person TO Ant WITH GRANT OPTION
REVOKE All ON Person TO Public
DENY Select ON Person TO Ant CASCADE
```

Każdy obiekt (w tym tabela i widok) ma przypisanego właściciela. Kto jest właścicielem danego obiektu można sprawdzić wykorzystując polecenie T-SQL.

```
OBJECTPROPERTY(object_id,'owner_id')
```

Identyfikatory właścicieli można znaleźć w widokach sys.database_principals i sys.server_principals, a identyfikatory obiektów w sys.objects.
Własność obiektu (z pewnymi zastrzeżeniami można zmienić).

ALTER AUTHORIZATION ON OBJECT::<nazwa_obiektu> TO <nowy_właściciel>

Własność obiektów może tworzyć tzw. łańcuchy (ownership chains). Mówimy, że łańcuch własności jest nieprzerwany w następującej sytuacji: niech widok vw2 korzysta z widoku vw1, a ten z kolei pobiera dane z tabeli tab1 i właścicielem wszystkich tych obiektów jest jedna osoba (np. user1). W opisanej powyżej sytuacji jeśli innemu użytkownikowi (np. user2) zostanie przyznane prawo do czytania (SELECT) tylko widoku vw2, to rzeczywiście ten użytkownik będzie mógł obejrzeć dane (SELECT * FROM vw2). Jeśli jednak łańcuch jest przerwany (czyli np. właścicielem vw1 jest user3), nie będzie możliwości odczytania widoku.

W ramach zadanie zrealizować zadanie z lab. 04, skrypt Lab04.26 dla użytkowników utowrzonych w ramach tego laboratorium.