



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Korszerű adatbázisok előadás 04



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Miről lesz szó?

Programozás T-SQL-ben

- Tranzakciók
- Triggerek
- Nézetek
- Tárolt eljárások
- Függvények

A tranzakció DML-utasítások olyan sorozata, amelyet egyetlen logikai egységként kezelhetünk

A tranzakció végén vagy minden változást érvényesítünk (COMMIT), vagy minden egyes lépést visszavonunk (ROLLBACK)



Tranzakció* tulajdonságok - ACID



- **Atomicity** – nem valósulhat meg részlegesen
- **Consistency** – végrehajtása után az állapot konzisztens marad (pl. kényszerek teljesülnek)
- **Isolation** – a párhuzamosan futó tranzakciók nem zavarhatják egymást
- **Durability** – sikeres lefutás után a változás tartósan megmarad

Zárolás (lock) fogalma

A zárolás olyan eszköz, amely segítségével az adatbáziskezelő rendszer megakadályozza, hogy egyidőben több tranzakció is módosítsa ugyanazokat az adatokat.

- A zárolásnak fontos szerepe van a tranzakciók izolálásában
- Amikor egy tranzakció elkezdi az adatok módosítását, akkor az érintett adatok zárolódnak, így a többi tranzakció nem tudja módosítani őket
- A zárolás megvalósulhat több szinten (sor, lap, tábla) és többféle módon (pl: kizárólagos, megosztott)

Egyidejű (konkurens) tranzakciók kezelése

A konkurens hozzáférések a következő problémákat vetik fel:

Módosítások elvesztése (**lost updates**)

- Amennyiben egy sor módosítását egyszerre végzi két tranzakció, akkor amelyik később menti el a módosítást, az felülírja az előzőleg módosított adatokat.

„Piszkos” adatok olvasása (**dirty reads**)

- Egy nem véglegesített tranzakció adatait olvassuk. Az adat azonban még változhat a tranzakció végrehajtása során.

Egyidejű (konkurens) tranzakciók kezelése (folyt.)

„Nem megismételhető” olvasás (**non-repetable reads**)

- Ugyanazt az adatot többször olvassuk, és mindig más eredményt kapunk, mert egy másik tranzakció közben változtatja az adatot.

Fantom adatok olvasása (**phantom reads**)

- Többször megismételt olvasás közben a korábban meglévő sorok elvesznek, vagy újak kerülne be az eredménybe, mivel egy közben egy másik tranzakció „INSERT” vagy „DELETE” műveletet hajtott végre

Elkülönítési (Izolációs) szintek

Az izolációs szintek azt szabályozzák, hogy milyen módon kezeljük a konkurencia-problémákat.

Az izolációs szintek szigorúság* szerint növekvő sorrendben

- ☐ **Read uncommitted:** minden adat olvasható (a nem véglegesítettek is)
- ☐ **Read committed:** csak a véglegesített (COMMITTED) adatok olvashatók (alapértelmezett szint)
- ☐ **Repeatable read:** az olvasott adatot nem módosíthatja más tranzakció
- ☐ **Serializable:** az olvasott adathalmazra nem engedélyezett az új adat beszúrása sem

*A szigorúbb izolációs szint csökkenti a konkurenciából adódó problémák valószínűségét, viszont növeli a zárolások miatti várakozási időt. A szigorúbb szint mindig tartalmazza a felette lévő (kevésbé szigorú szintek) korlátozásait is.

Konkurencia problémák és izolációs szintek - táblázat

Levels/ Solved problems	Lost updates	Dirty reads	Nonrepeatable reads	Phantom reads
Read uncommitted	+	-	-	-
Read committed	+	+	-	-
Repeatable Read	+	+	+	-
Serializable	+	+	+	+

Olvasási konzisztencia – Read committed

Az olvasási konzisztencia célja, hogy biztosítsa, hogy minden felhasználó úgy lássa az adatokat, ahogy az utolsó commit pillanatában léteztek

- Aki csupán olvas (SELECT), az nem vár mások folyamatban lévő tranzakcióira
- Aki módosít, nem vár azokra, akik csak olvasnak
- Aki módosít, vár azokra, akik szintén módosítják ugyanazokat az adatokat



Fontosabb tranzakció módok

☐ Autocommit tranzakciók:

Minden utasítás egy külön tranzakció (alapértelmezett), láthatatlan BEGIN TRANSACTION utasítással (ld. később)

☐ Explicit tranzakciók:

- ☐ Mi magunk definiáljuk a BEGIN TRANSACTION utasítással (ld. később).
- ☐ Az explicit tranzakciók egymásba is ágyazhatók. Ilyenkor a @@TRANCOUNT változó mondja meg, hogy hányadik szinten vagyunk*
- ☐ Kezdetben, illetve ROLLBACK után a @@TRANCOUNT értéke 0
- ☐ Minden BEGIN TRANSACTION 1-gyel növeli, minden COMMIT 1-gyel csökkenti a @@TRANCOUNT értékét

* A @@TRANCOUNT jelentése nem beágyazott tranzakció esetén: adott session-ban futó, nyitott tranzakciók száma. A nyitott tranzakciók megtekinthetők pl: a DBCC OPENTRAN parancs segítségével

Fontosabb tranzakció módok

❑ Implicit tranzakciók:

- ❑ Ha @@TRANCOUNT = 0, akkor a legelső tranzakciót kiváltó utasítás hatására (ld. Köv. dia) elindul egy új tranzakció, így a @@TRANCOUNT értéke 1 lesz
- ❑ Ha @@TRANCOUNT > 0, akkor már nem indul el láthatatlan BEGIN TRANSACTION
- ❑ Az implicit tranzakció befejeződik, ha @@TRANCOUNT 0 lesz (pl. COMMIT vagy ROLLBACK hatására – ezt nekünk kell kiadni)
- ❑ Az implicit tranzakciós mód az SQL server-en a SET IMPLICIT TRANSACTIONS ON utasítással aktiválható

Tranzakciót kiváltó SQL-utasítások

CREATE

INSERT

UPDATE

SELECT
(ha táblát is érint)

ALTER TABLE

TRUNCATE
TABLE

DELETE

GRANT

REVOKE

FETCH

Explicit tranzakciók megvalósítása SQL-ben

```
BEGIN TRANSACTION [Tran1]
  BEGIN TRY
    -- SQL utasítások

  COMMIT TRANSACTION [Tran1]

  END TRY

  BEGIN CATCH
    -- Hibakezelés
    ROLLBACK TRANSACTION [Tran1]
  END CATCH
```

Explicit tranzakciók megvalósítása SQL-ben

- Véglegesítés: **COMMIT;**
- Visszaállítás a tranzakció előtti állapotra: **ROLLBACK;**
- Mentési pont elhelyezése a tranzakcióban:
SAVEPOINT név; (nem minden rendszerben létezik)
- Visszaállítás a tranzakció közepében lévő mentési pontra:
ROLLBACK TO SAVEPOINT név;

Az adatok értéke a **COMMIT** vagy **ROLLBACK** előtt

- ❑ Az adatok tranzakció előtti állapotát még mindig vissza lehet állítani (ROLLBACK)
- ❑ Az a munkamenet, amelyben folyamatban van a tranzakció, az SELECT utasításaiban látja a tranzakció során addig végrehajtott összes változtatást (DML)
- ❑ A többi munkamenet ekkor még nem látja a folyamatban lévő tranzakció által végzett módosításokat
- ❑ A folyamatban lévő tranzakció által módosított sorok le vannak zárva (foglalva). Más tranzakciók nem módosíthatják ezeket a sorokat mindaddig, amíg a folyamatban lévő tranzakció be nem fejeződik.

Az adatok értéke a **COMMIT** után

- ☐ Az adatok módosítása véglegessé válik az adatbázisban.
- ☐ A tranzakciós módosítások előtti értékek már nem visszaállíthatóak.
- ☐ Minden munkamenet látja az új értékeket.
- ☐ A zárok (lefoglalások) felszabadulnak.
- ☐ A tranzakció során esetleg elhelyezett mentési pontok (SAVEPOINT) elvesznek.

Az adatok értéke a **ROLLBACK** után

- ☐ A tranzakció által elvégzett minden módosítás elveszik
- ☐ A régi értékek visszaállnak
- ☐ A zárok (lefoglalások) felszabadulnak

(DML) Triggerek*

Olyan speciális eljárások, amelyek DML utasítás előtt, után vagy helyett futnak le.

- A triggerek az adott esemény (pl: INSERT) bekövetkezésekor automatikusan futnak le
- A triggerek segítségével olyan kényszerek is teljesíthetők, amelyeket nem tudunk megadni az adat definíció során (pl: egy tanulónak max. 50 jegye lehet)

OSZTÁLYZAT		
Tkód	Tankód	Jegy
T01	Tan01	5
T01	Tan02	3

* Az egyéb trigger típusokkal (pl: LOGON triggerek) nem foglalkozunk

Triggerek - SSMS

The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the active file is 'SQLQuery9.sql - gyak01.database.windows.net.dvd (gmolnar285 (65))* - Microsoft SQL Server Management Studio'. The menu bar includes File, Edit, View, Query, Project, Tools, Window, and Help. The toolbar contains various icons for file operations, query execution, and development tools. The Object Explorer on the left shows the database structure for 'gyak01.database.windows.net (SQL Server 12.0.2000.8 - g...)'.

In the Object Explorer, the 'Triggers' folder under the 'dbo' schema is highlighted with a red rectangle, and the trigger 'tg_orak_szama_max_10' is selected. The main query window, titled 'SQLQuery10.sql - g...d (gmolnar285 (67))*', displays the following SQL code:

```
CREATE TRIGGER [dbo].[tg_orak_szama_max_10]
ON [dbo].[Orak]
INSTEAD OF INSERT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @oraszam AS int
    DECLARE @tanar AS int

    BEGIN TRAN
        INSERT INTO dbo.Orak (
            ora_id, tanar, nap, Sav, targy, terem
        )
        SELECT i.ora_id, i.tanar, i.nap, i.Sav, i.targy, i.terem
        FROM inserted i

    SELECT @tanar = tanar
    FROM inserted
```

Nézetek (View-k)

A nézet egy elmentett, névvel ellátott lekérdezés.

- A nézetekből ugyanúgy lehet lekérdezni, mint táblákból
- A nézetek segítségével meghatározhatjuk a megjelenítendő adatok körét
- A nézetekhez adhatunk jogosultságokat az alaptáblákhoz való jogosultságok nélkül is
- A DML-műveletek nem mindig megengedettek nézeteken keresztül

A Nézetek előnyei

Korlátozható az
adatok elérése

A bonyolultabb
lekérdezések
egyszerűbb
formára hozhatók

Az adatokat
többféle
nézőpontból
szemlélhetjük

Az
adatfüggetlenség
biztosítása

A Nézetek két fő típusa

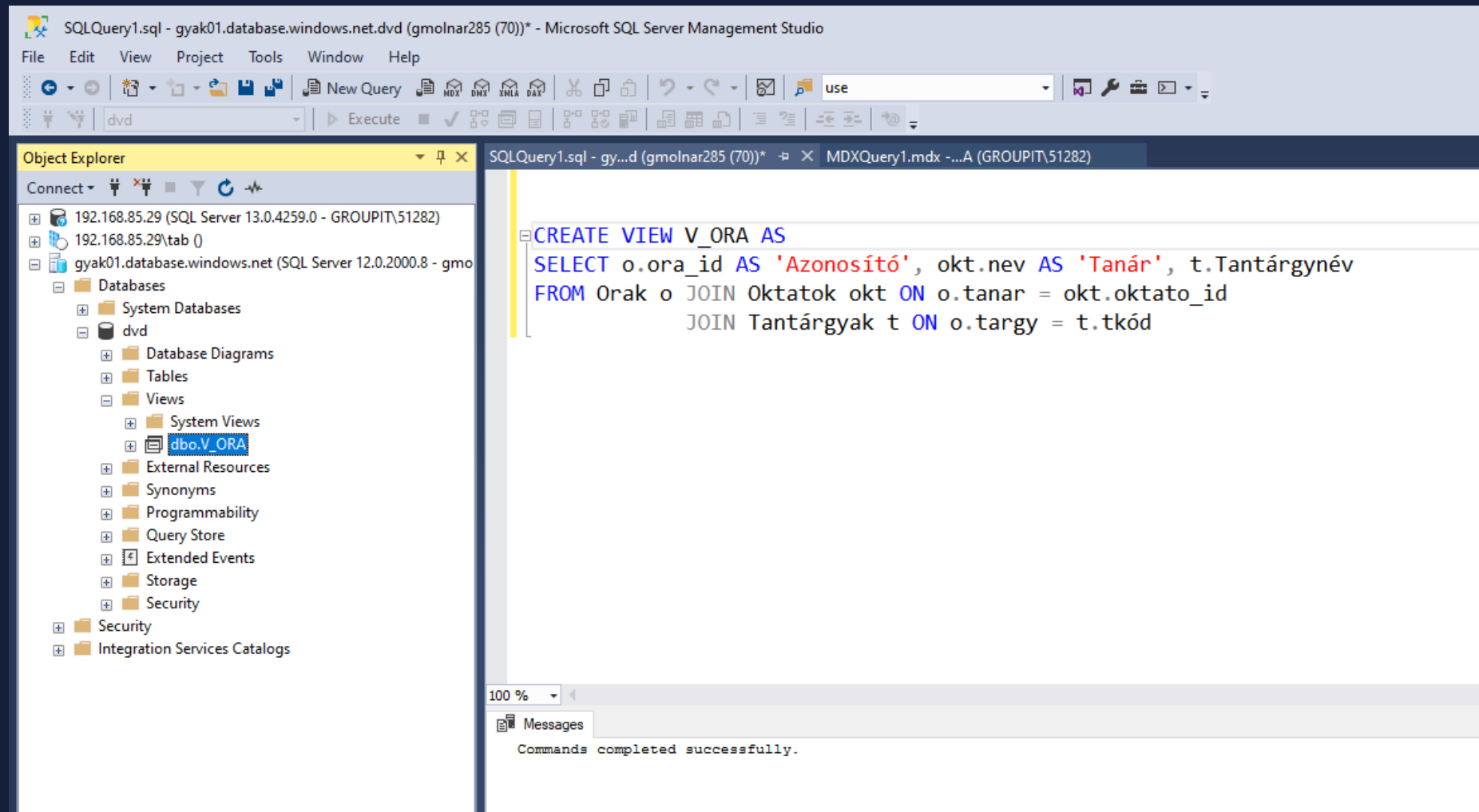
Virtuális

- Csak a lekérdezés tárolódik

Materializált

- Az adatok is tárolásra kerülnek

Nézetek – SQL Server Management Studio



Nézetek létrehozása SQL-ben

```
CREATE VIEW view_név  
[(oszlopnevek listája)]  
[WITH view_attribútumok]  
AS SELECT_utasítás  
[WITH CHECK OPTION]*
```

- ☐ A szögletes zárójelbe tett részek opcionálisak
- ☐ A view_attribútumokkal nem foglalkozunk
- ☐ Az SQL Server 2016-tól a CREATE OR ALTER VIEW forma is használható

Példa

```
CREATE VIEW v_klimas_szobak  
(Azonosító, Szobaszám, [Férőhelyek száma])  
AS  
SELECT szoba_id, szoba_szama,      ferohely  
FROM szoba  
WHERE KLIMAS='i'
```

A Nézet használata

```
SELECT * FROM v_klimas_szobak
```

*A WITH CHECK OPTION záradék az adatintegritást segíti. Bekapcsolásával csak olyan adatmódosítást végezhetünk a view-n keresztül, amely megfelel a SELECT-ben lévő feltételeknek

Nézetek – Példa

TANULÓ		
Tkod	Tnev	Tszulido
T01	Kiss Béla	1999.01.01
T02	Nagy Ilona	2003.02.12

OSZTÁLYZAT		
Tkod	Tankód	Jegy
T01	Tan01	5
T01	Tan02	3
T02	Tan01	4

TANTARGY	
Tankod	Tannév
Tan01	Algebra
Tan02	Analízis
Tan03	Programozás

V_OSZTALYZAT		
Tnév	Tannév	Jegy
Kiss Béla	Algebra	5
Kiss Béla	Analízis	3
Nagy Ilona	Algebra	4

```

CREATE VIEW V_OSZTALYZAT AS
SELECT t.tnev AS 'TNév',
       tt.tannev AS 'Tannév',
       o.jegy
FROM Osztalyzat o
     JOIN Tanulo t ON o.tkod = t.tkod
     JOIN Tantargy tt ON o.tankod =
tt.tankod
  
```

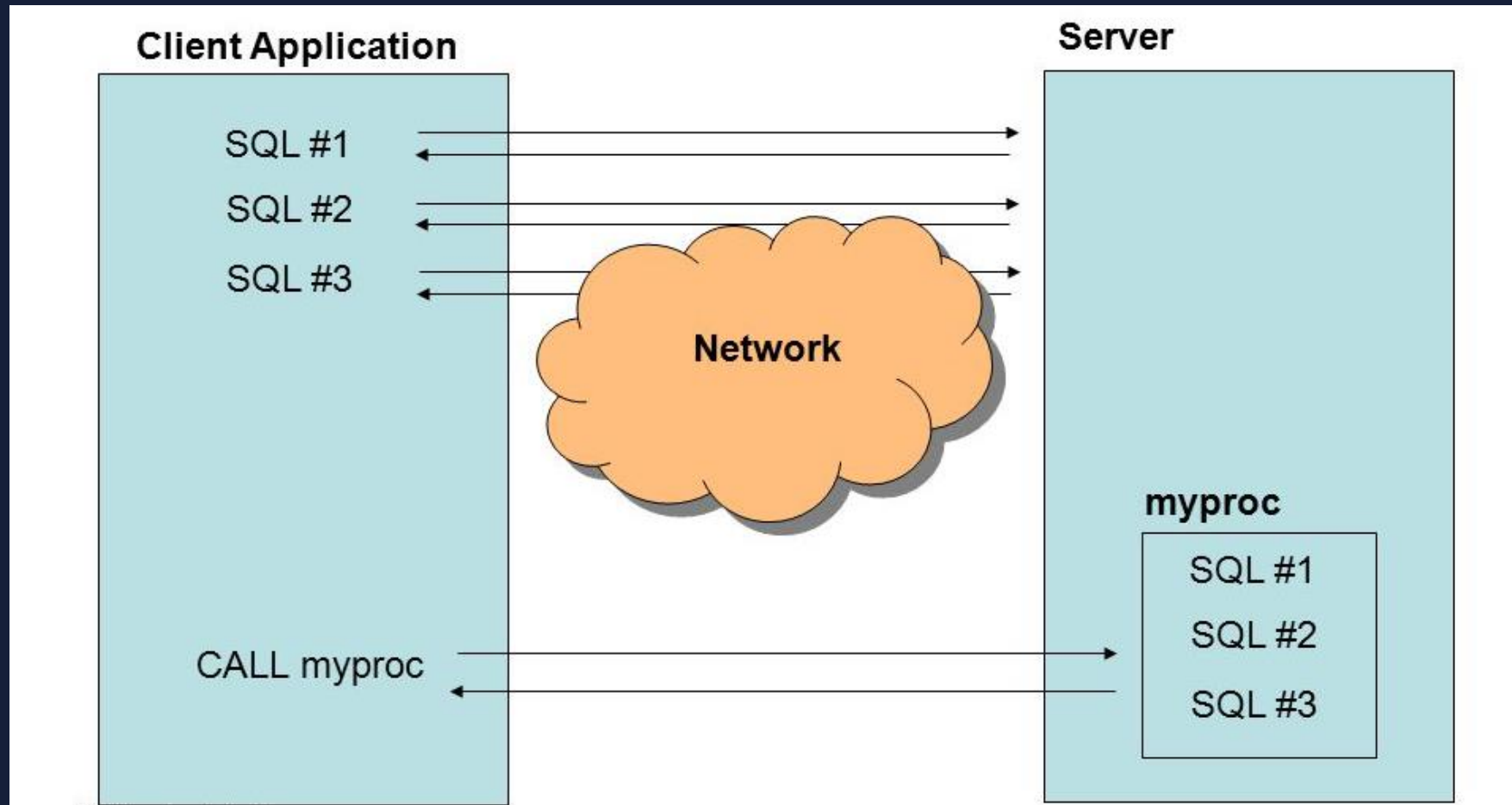
Tárolt eljárás (Stored procedure)

A tárolt eljárás olyan adatbázis objektumként tárolt program, amely SQL-utasításokat is tartalmazhat.

A tárolt eljárások főbb jellemzői

- Input és output paramétereket, valamint különböző algoritmikus szerkezeteket is tartalmazhatnak (elágazás, ciklus)
- Az adatbázis szerveren tárolódnak
- Futtatásuk jogosultságokhoz köthető

A tárolt eljárások működése



A tárolt eljárások előnyei

Hatékonyság

- Egyszerre több alkalmazás is használhatja őket
- Csökken a szerver-kliens üzenetek száma

Fenntarthatóság

- A kódok egy központi helyen találhatók
- A módosítás, tesztelés elkülönülhet a tárolt eljárást hívó alkalmazástól

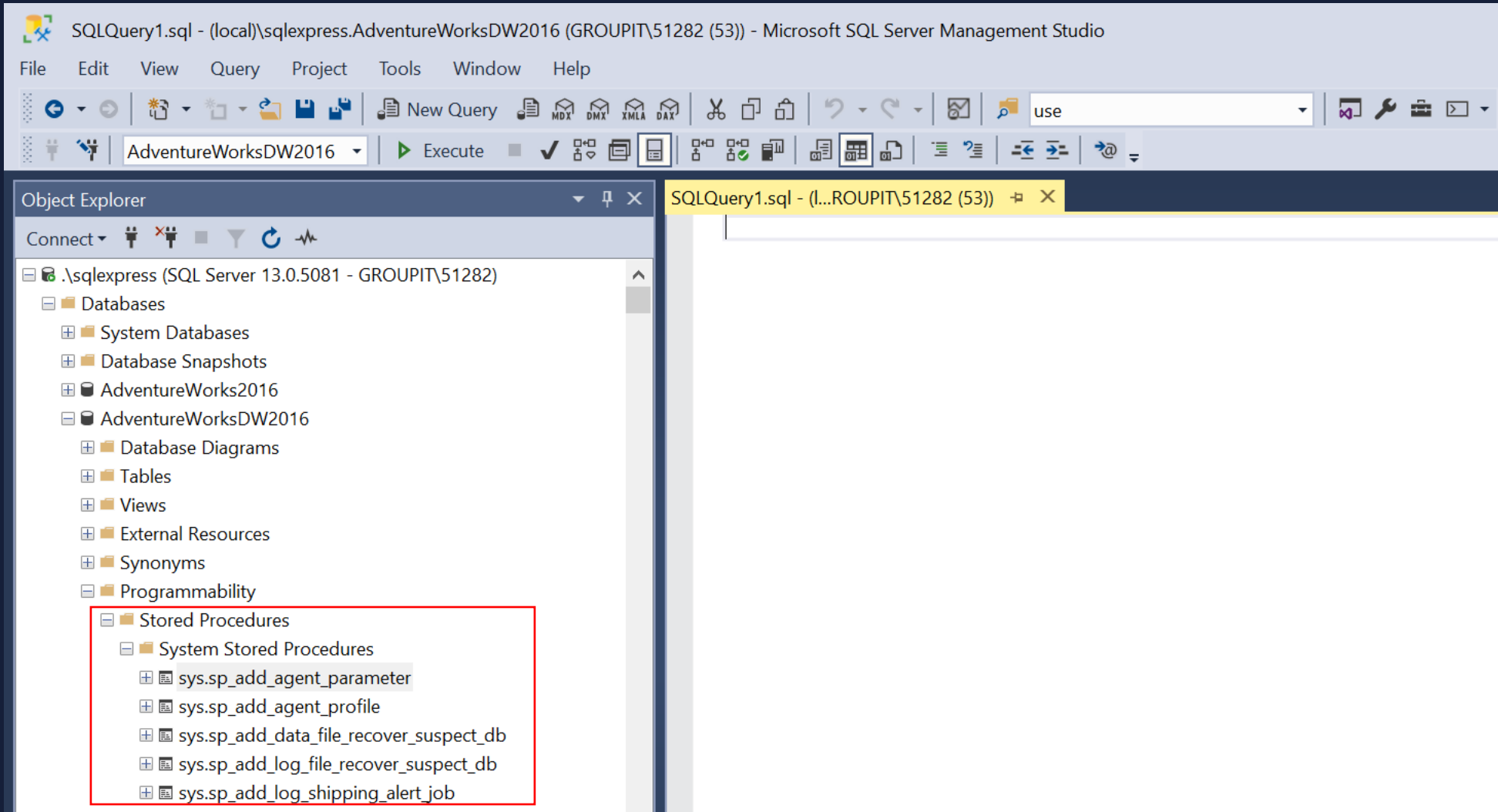
Biztonság

- Használatukkal korlátozható a táblákhoz való hozzáférés
- A hozzáférés biztosítása így nem a tárolt eljárást hívó alkalmazás feladata

Üzleti logika elkülönítése

- Az üzleti logika elkülönül a tárolt eljárást hívó alkalmazástól
- Csökkenhet a kliens programok miatti adathibák száma

Tárolt eljárások az MS SQL-ben



Tárolt eljárások létrehozása SQL-ben

CREATE PROCEDURE eljárás_név

[paraméterek listája]

[WITH eljárás_opciók]

AS

[BEGIN]

Utasítások

[END]

Példa

```
CREATE PROCEDURE vevo_foglalasok  
@vevoid nvarchar(20)
```

```
AS
```

```
BEGIN
```

```
    SELECT *
```

```
    FROM Foglalas
```

```
    WHERE UGYFEL_FK = @vevoid
```

```
END
```

Futtatás

```
EXEC vevo_foglalasok 'lászlo2'
```

- ☐ A szögletes zárójelbe tett részek opcionálisak
- ☐ Az eljárás opciókkal nem foglalkozunk
- ☐ Az SQL Server 2016-tól a CREATE OR ALTER PROCEDURE forma is használható

Függvény (UDF-User defined function)

A (felhasználó által definiált) függvény olyan adatbázis objektum, amely végrehajt egy tevékenységet, majd annak eredményét visszaadja egy érték vagy egy tábla formájában

A függvények főbb jellemzői

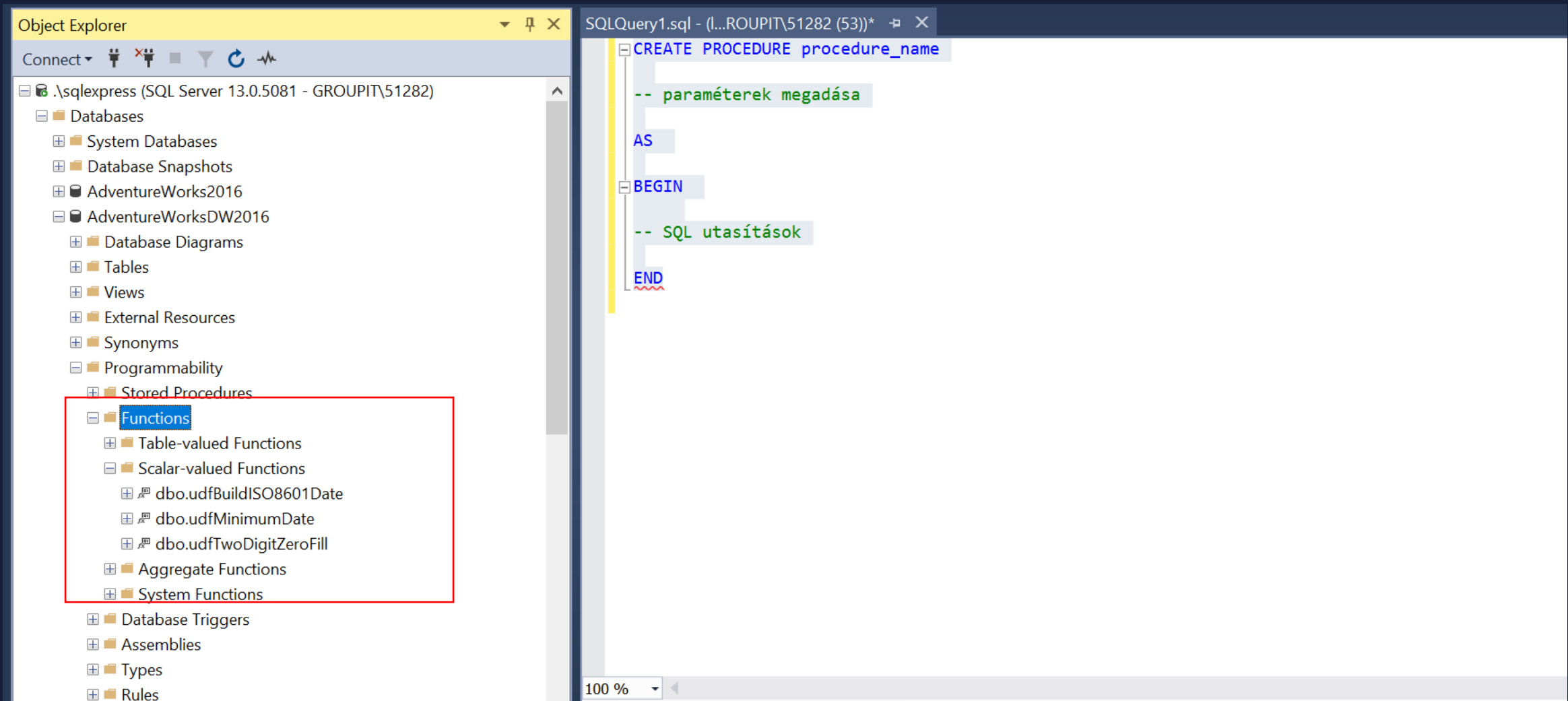
- Input paramétereket, SQL-utasításokat, valamint különböző algoritmikus szerkezeteket is tartalmazhatnak (elágazás, ciklus)
- Az adatbázis szerveren tárolódnak
- Futtatásuk jogosultságokhoz köthető
- Felhasználhatók SQL-utasításokban, pl: SELECT utasításban

Függvények vs. Tárolt eljárások

A függvények sok tekintetben a tárolt eljárásokhoz hasonló tulajdonságokkal rendelkeznek, de van közöttük néhány fontos különbség

Függvények	Tárolt eljárások
Csak input paraméterek	Input és output paraméterek
Tranzakciók nem használhatók	Tranzakciók is használhatók
A SELECT utasításban használhatók	A SELECT utasításban nem használhatók
Kivételkezelés nem használható	Kivételkezelés használható
Nem hívhat meg tárolt eljárást	Függvényhívás lehetséges
Mindig egy értéket ad vissza	Visszaadhat nulla, egy vagy több értéket

Függvények az MS SQL-ben



The screenshot displays the Microsoft SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the hierarchy of the server instance. The 'Functions' folder under 'Programmability' is highlighted with a red rectangle. The right pane shows a SQL query editor with the following code:

```
CREATE PROCEDURE procedure_name
-- paraméterek megadása
AS
BEGIN
-- SQL utasítások
END
```

The query editor shows the code with syntax highlighting and a vertical line indicating the current position. The status bar at the bottom indicates 100% zoom.

Függvények létrehozása SQL-ben

```
CREATE FUNCTION fv_név
([paraméterek listája])
RETURNS adattípus_név
[WITH fv_opciók]
[AS]
BEGIN
    Utasítások
RETURN skalar_kifejezés
END
```

Skalár-
értékű
függvény

```
CREATE FUNCTION fv_név
([paraméterek listája])
RETURNS TABLE
[WITH fv_opciók]
[AS]
RETURN select_kifejezés
```

Tábla-értékű
függvény

Példa

```
CREATE FUNCTION csillagszam
(@szallas_id INT)
RETURNS INT
AS
BEGIN
    DECLARE @db INT
    SELECT @db=CSILLAGOK_SZAMA
    FROM Szallashely
    WHERE SZALLAS_ID=@szallas_id
    RETURN @db
END
```

Függvény alkalmazása

```
SELECT dbo.csillagszam(5)
```

- ☐ A szögletes zárójelbe tett részek opcionálisak, A fv. opciókkal nem foglalkozunk
- ☐ Az SQL Server 2016-tól a CREATE OR ALTER FUNCTION forma is használható
- ☐ A CLR-függvényekkel nem foglalkozunk

Megjegyzések

- ❑ A nézetek definíciójában nem használható az ORDER BY záradék (kivéve, ha TOP záradék is definiálva van)
- ❑ A nézetekben nem hivatkozhatunk ideiglenes táblákra
- ❑ Tárolt eljárásoknál az output paramétereket az OUT kulcsszóval jelölhetjük, pl: @i INT OUT
- ❑ A függvényekben sem paraméterként, sem visszaadott értéként nem szerepelhet a Timestamp típus
- ❑ A függvények egymásba is ágyazhatók
- ❑ Tárolt eljárásoknál és fv-eknél
 - ❑ Elágazások megvalósítása: IF feltétel utasítás[blokk] [ELSE utasítás[blokk]]
 - ❑ Ciklusok megvalósítása: WHILE feltétel utasítás[blokk]