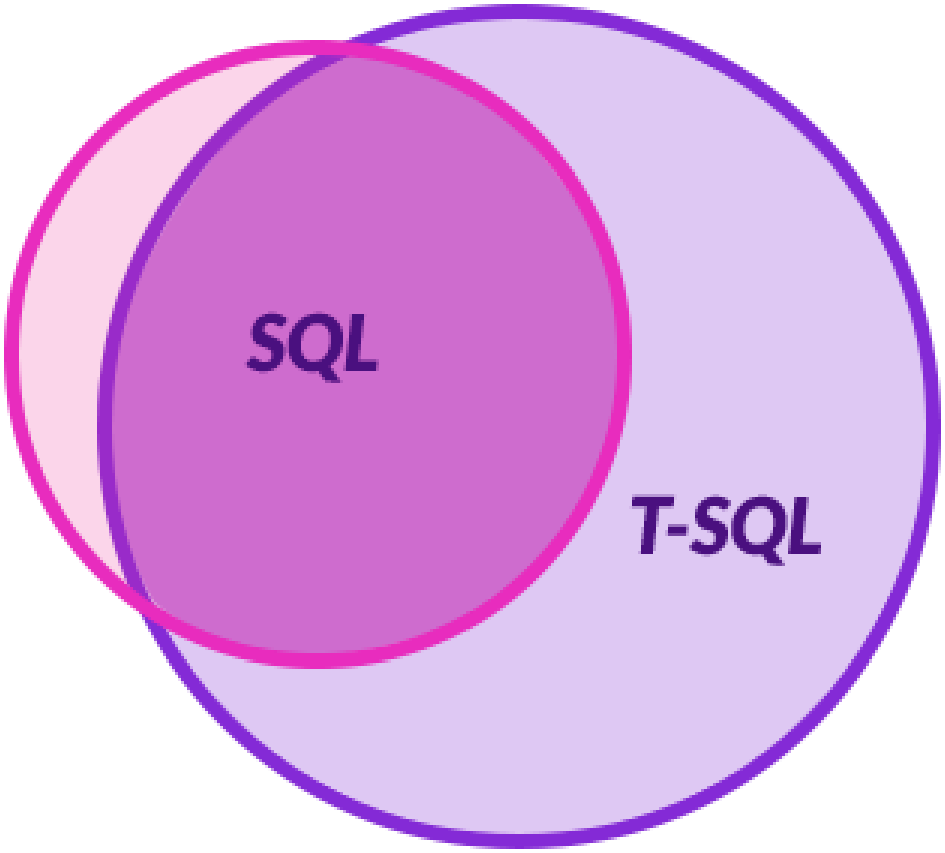
TÁROLT ELJÁRÁSOK ÉS FÜGGVÉNYEK

ADATBÁZISOK

Dóka - Molnár Andrea
andrea.molnar@ubbcluj.ro

BABEŞ-BOLYAI TUDOMÁNYEGYETEM
Matematika és Informatika Kar

Visszatekintés: Transact-SQL



SQL - ANSI/ISO standard programozási nyelv

Kiterjesztések relációs ABKR-k esetén:

MS SQL – Transact SQL

Oracle – PL/SQL

1. rész
Tárolt eljárások

Feladat:

https://goo.gl/cUjNdW

Tárolt eljárások

- Futtatható programegység.
- Adatbázisban eltárolható, mint objektum (Object Explorer → <megfelelő adatbázis> →Programmability).
- *Prekompilált program:* ABKR lefordítja, meghatározza a végrehajtási tervet és optimalizálja.

  Pl. SQL parancs karaktersorként történő megadása:

  ABKR minden alkalommal: le kell fordítsa, a végrehajtási tervet ki kell dolgozza és optimalizálja.

  → Tárolt, lefordított eljárás meghívása hatékonyabb
- Tárolt eljárások egymásba is ágyazhatók (max. 32 szint).
- Meghívhatók kliens alkalmazásokból is.

Tárolt eljárások osztályozása

- rendszerszintű tárolt eljárások (system stored procedure)
  - Rendszeradminisztrációra használt.
  - SQL Server-be „beépítettek”.
  - A master adatbázisban találhatók (lsd. sysobjects tábla).
- felhasználói tárolt eljárások (user-defined stored procedure)
  - Felhasználó által megírt, létrehozott.
  - Felhasználó adatbázisában tárolt.
- felhasználói rendszerszintű tárolt eljárás (user-defined system stored procedure)
  - Felhasználó által megírt, létrehozott - master adatbázisban tárolt. → Minden adatbázisból hozzáférhetők.

Felhasználói tárolt eljárások létrehozása

```
CREATE [OR ALTER] {PROC | PROCEDURE}
 [schema_name.] procedure_name [; number]
 [{@parameter [type_schema_name.] data_type}
 [VARYING] [= default] [OUT | OUTPUT | [READONLY]]
[ ,...n ]
[WITH <procedure_option> [ ,...n ] ]
 AS
{[BEGIN]
  sql_statement[;]
[ ...n ]
[END]}
[;]

<procedure_option> ::=
 [ENCRYPTION]
 [RECOMPILE]
 [EXECUTE AS Clause]
```

Megj.
Tárolt eljárás nevének egyedinek kell lenni egy sémán belül.
Eljárás neve maximálisan 128 karakter hosszúságú lehet.

Tárolt eljárások paraméterei

- Paraméterek (tárolt eljárásban)
  - Név – Típus – Mód hármas

    `<paraméternév> <értéktípus> [OUT]`
  - Típus: az összes Transact-SQL adattípus lehet.
  - Mód lehet:
    - OUT – kimeneti paraméterek esetén
    - Bemeneti paraméterek esetén: nem kell megadni semmit.
    - Cursor adattípus - csak kimeneti paraméter lehet (VARYING kulcsszóval)
- Paraméterek száma – max 1200.
- Default: alapértelmezett értéket adhatunk meg a paraméternek.
- Lok. változók deklarálása – az eljárás törzsében bárhol; ajánlás: legelején.

Példa tárolt eljárásra

```
CREATE PROCEDURE spMindenAlkalmazott
AS
BEGIN
   SELECT SzemSzam, Nev, Cim
   FROM Alkalmazottak

   SELECT COUNT(*)
   FROM Alkalmazottak
END
GO
```

- Meghívás:

```
EXEC spMindenAlkalmazott
```

Alkalmazottak(SzemSzám, Név, Fizetés, Cím, *RészlegID*);

Példa – tárolt eljárás bemeneti paraméterrel

Példa: *Tárolt eljárás új szállító beszúrására.*

```
CREATE PROCEDURE spUjSzallito
                (@pSzNev VARCHAR(30),
                @pSzallCim VARCHAR(30))
AS BEGIN
    SET NOCOUNT ON
    INSERT INTO Szallitok VALUES
              (@pSzNev, @pSzallCim)
   END
```

Szallitok (SzallKod, Nev, Cim), SzallKod-identity típusú

Tárolt eljárás meghívása, futtatása

```
CREATE PROCEDURE spUjSzallito
    (@pSzNev VARCHAR(30),@pSzCim VARCHAR(30))
AS BEGIN
   SET NOCOUNT ON
   INSERT INTO Szallitok VALUES(@pSzNev,@pSzCim)
 END
```

Meghívás:

- Konkrét értékek megadásával:

```
EXEC spUjSzallito 'NoName Kft.',
                  'Semmi kozepe ut 102.'
```

Tárolt eljárás meghívása, futtatása

```
CREATE PROCEDURE spUjSzallito
    (@pSzNev VARCHAR(30),@pSzCim VARCHAR(30))
AS BEGIN
   SET NOCOUNT ON
   INSERT INTO Szallitok VALUES(@pSzNev,@pSzCim)
 END
```

Meghívás:

- Konkrét értékek megadása paraméterekkel:

```
EXEC spUjSzallito @pSzNev='NoName Kft.',
                  @pSzCim = 'Semmi kozepe ut 102.'
```

- sorrend nem számít, de csak az eljárásbeli paraméternevek használhatóak:

```
EXEC spUjSzallito @pSzCim = 'Semmi kozepe ut
                  102.', @pSzNev='NoName Kft.'
```

Példa – tárolt eljárás kimeneti paraméterrel

Példa: *Adott nevű részleg dolgozóinak fizetését növeljük 10%-kal!*

```
CREATE PROCEDURE spReszlAlkUpdate
    (@pRNev VARCHAR(30), @pOut INT OUT)
AS BEGIN
 SET NOCOUNT ON
 UPDATE Alkalmazottak
  SET Fizetes*=1.1
  WHERE ReszlegID = (SELECT ReszlegID
                     FROM Reszlegek
                     WHERE ReszlegNev=@pRNev)
 SET @pOut=@@ROWCOUNT
END
```

Példa – tárolt eljárás kimeneti paraméterrel

Példa: *Adott nevű részleg dolgozóinak fizetését növeljük 10%-kal!*

```
CREATE PROCEDURE spReszlAlkUpdate
    (@pRNev VARCHAR(30), @pOut INT OUT)
AS BEGIN
 SET NOCOUNT ON
 UPDATE Alkalmazottak
  SET Fizetes*=1.1
  WHERE ReszlegID = (SELECT ReszlegID
                     FROM Reszlegek
                     WHERE ReszlegNev=@pRNev)
 SET @pOut=@@ROWCOUNT
END
```

Módosított sorok száma.

Példa – tárolt eljárás kimeneti paraméterrel

```
CREATE PROCEDURE spReszlAlkUpdate
    (@pRNev VARCHAR(30), @pOut INT OUT)
AS BEGIN
 SET NOCOUNT ON
 UPDATE Alkalmazottak
 SET Fizetes*=1.1
 WHERE ReszlegID = (SELECT ReszlegID FROM Reszlegek
                    WHERE ReszlegNev=@pRNev)
 SET @pOut=@@ROWCOUNT
END
```

- Meghívás:

```
DECLARE @out int
EXEC spReszlAlkUpdate 'Tervezes', @out OUT
SELECT @out [AS] ModositottSorokSzama
```

- **RETURN tárolt eljáráson belül?**

RETURN tárolt eljáráson belül

- Visszatérési értékkel rendelkező utasítás:
  - Alakja: `RETURN [egész_érték]`
  - Meghívás:

```
DECLARE @return_status int
EXECUTE @return_status = <procedure_name>
SELECT @return_status
```

  - Használatával azonnal kiléphetünk egy tárolt eljárásból, utasítás-sorozatból, batch-ből. (Az utána következő utasítások nem hajtódnak végre.)
  - `[egész_érték]`
    - Nem lehet NULL (máskülönben figyelmeztető üzenetet kapunk).
    - Minden rendszerszintű eljárás 0-val tér vissza sikeres végrehajtás esetén.

Példa - RETURN tárolt eljáráson belül

Példa: *Adott nevű részleg alkalmazottainak számát térítsük vissza (tárolt eljárás segítségével)! Ha nem dolgozik senki ott, -1-t térítsünk vissza!*

```
CREATE PROCEDURE spRAlkSzam_r
                (@pRNev VARCHAR(30))
AS BEGIN
 SET NOCOUNT ON
 DECLARE @AlkSzam INT
 SELECT @AlkSzam=count(*) FROM Alkalmazottak
 WHERE ReszlegID =
    (SELECT ReszlegID FROM Reszlegek
     WHERE ReszlegNev=@pRNev))
 IF @AlkSzam=0 RETURN -1
    ELSE RETURN @AlkSzam
END
```

Példa - RETURN tárolt eljáráson belül

```
CREATE PROCEDURE spRAlkSzam_r(@pRNev VARCHAR(30))
AS BEGIN
 SET NOCOUNT ON
 DECLARE @AlkSzam INT
 SELECT @AlkSzam=count(*) FROM Alkalmazottak
 WHERE ReszlegID =
    (SELECT ReszlegID FROM Reszlegek
     WHERE ReszlegNev=@pRNev))
 IF @AlkSzam=0 RETURN -1
    ELSE RETURN @AlkSzam
END
```

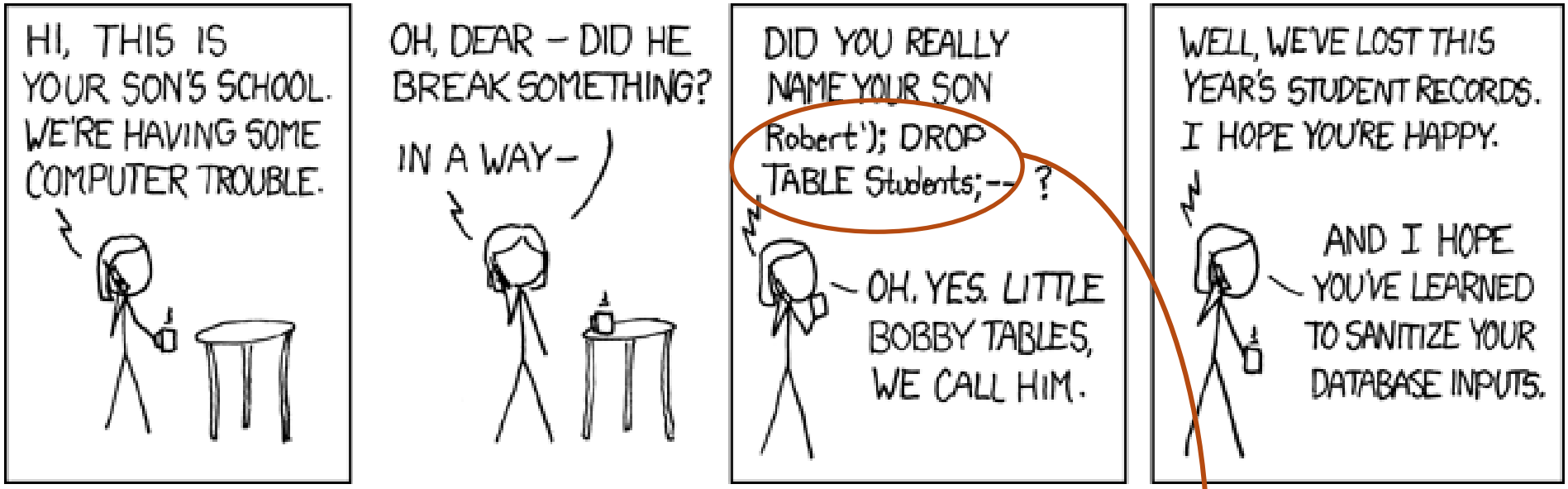
- Meghívás:

```
DECLARE @return_st int
EXEC @return_st = spRAlkSzam_r 'Tervezes'
SELECT @return_st [AS] ReturnValue
```

A tárolt eljárások előnyei

- Átláthatóság: Egy helyen koncentrálódik minden adatbázis lekérdezés - esetleges hibák megtalálása egyszerűbb.
- Jobb biztonság:
  - Jogok adása a meghívásra, de NEM az adatokon való direkt változtatásra.
  - Nem szükséges minden táblára külön adni a jogot.
  - Kód enkriptálása (*lsd. később* - WITH ENCRYPTION).
  - SQL Injection kivédésének egy lehetséges módja.

SQL Injection



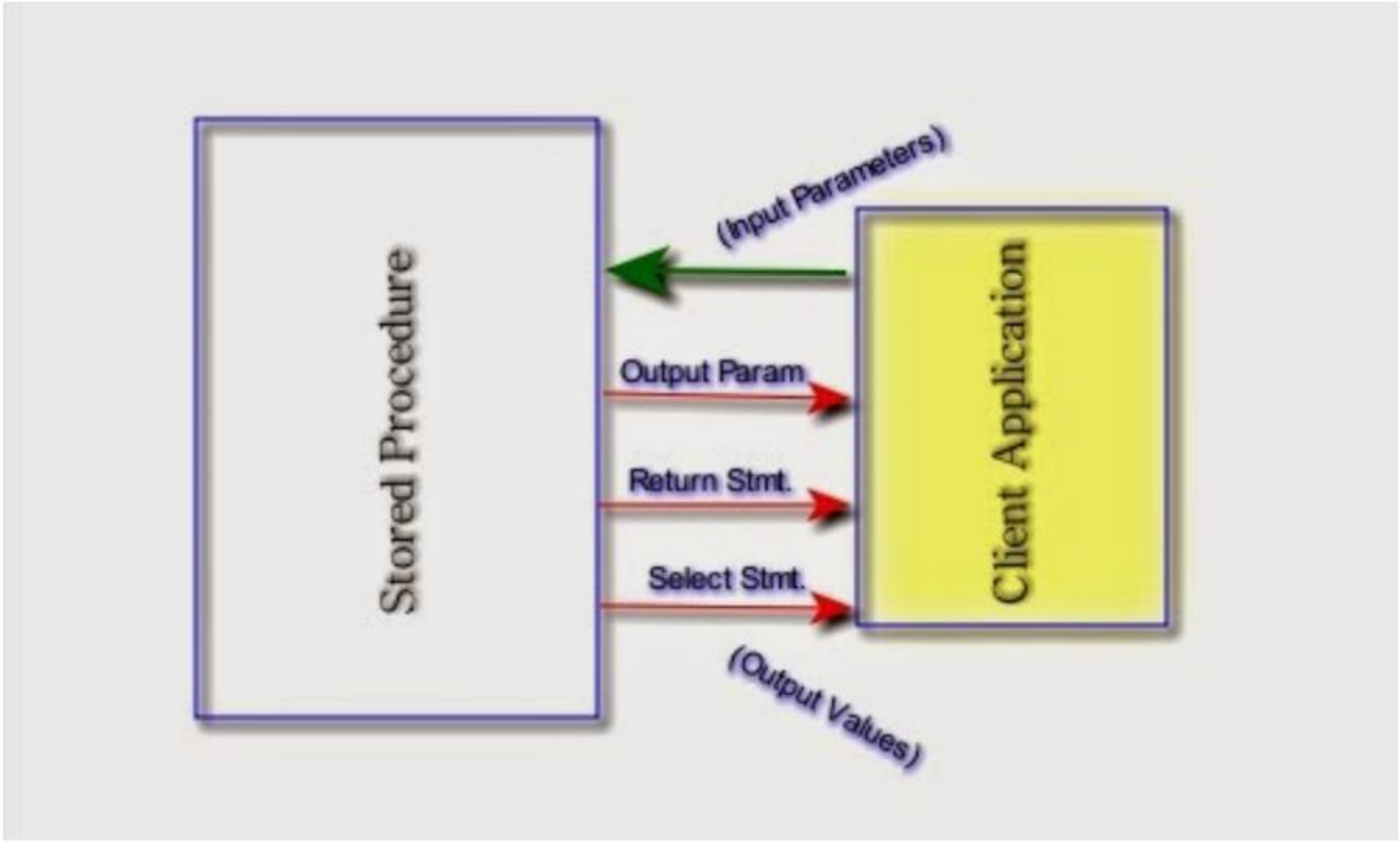
Forrás: https://xkcd.com/327/

```
insert into students ('Robert'); DROP TABLE
Students;--');
```

A tárolt eljárások előnyei

- Az optimalizálás végrehajtható az eljárások megírásakor, nem csak futtatáskor.
- Nagyobb adatfüggetlenség biztosítása: a rendszer, illetve adatbázis-specifikus részletek elrejtésének lehetősége.
  - Ha változik a logika, elég a szerver oldalon változtatni.
- Meghívhatók kliens alkalmazásokból is.
- Teljesítmény: kevesebb információt kell átküldeni a hálózaton.

Tárolt eljárás és kliens alkalmazás közti kommunikáció



A tárolt eljárások hátrányai

- Tárolt eljárás vs. utasítás-sorozat

  Hatékonyság $\rightarrow$ tárolt eljárás

  Hordozhatóság $\not\rightarrow$ tárolt eljárás

  - ABKR kicserélésekor: a tárolt eljárásokat át kell írni.

- Nincs szabvány, minden rendszer sajátosan oldja meg:
  - Oracle - PL/SQL nyelv (SQL procedurális kiterjesztése)
  - MS SQL Server - TRANSACT-SQL (lsd. MS SQL Server dokumentáció).

2. rész
Függvények

Felhasználó által definiált függvények

- Szintaxis:

```
CREATE FUNCTION <függvény-név>
    [<paraméter-lista>]
RETURNS <típus> AS
BEGIN
<függvény törzse (TRANSACT-SQL utasítások)>
END;
```

Egyetlen értéket visszaadó függvények

Példa (*lsd. utolsó példa tárolt eljárásoknál*):

```
CREATE FUNCTION fnRAlkSzam
       (@pRNev VARCHAR(30))
RETURNS INT
BEGIN
  DECLARE @alkSzam INT
  SET @alkSzam =
      (SELECT COUNT(*) FROM Alkalmazottak
       WHERE ReszlegID =
            (SELECT ReszlegID
             FROM Reszlegek
             WHERE ReszlegNev=@pRNev))
  IF @AlkSzam=0 RETURN -1
    ELSE RETURN @AlkSzam
END
```

Egyetlen értéket visszaadó függvények

- Példa:

```
CREATE FUNCTION fnRAlkSzam (@pRNev VARCHAR(30))
RETURNS INT
BEGIN
   DECLARE @alkSzam INT
   SET @alkSzam =
      (SELECT COUNT(*) FROM Alkalmazottak
       WHERE ReszlegID =
            (SELECT ReszlegID
             FROM Reszlegek
             WHERE ReszlegNev=@pRNev))
  IF @AlkSzam=0 RETURN -1
     ELSE RETURN @AlkSzam
END
```

- Meghívás:

```
DECLARE @asz INT
EXEC @asz = fnRAlkSzam 'Tervezes'
SELECT @asz
```

Inline Table-Valued Functions (ITVF)

- „Nézetek”, melyek elfogadnak paramétereket + megfelelően térítenek vissza adatokat.
- Nincs BEGIN/END törzs.
- Egyetlen SELECT utasítást tartalmazhat – virtuális táblát térít vissza. ⟷ Az oszlopok nevei különbözőek kell legyenek.
- Szintaxis:

```
CREATE FUNCTION fgvnév
     ( [{@paraméternév adattípus}[,…n]] )
RETURNS TABLE [AS]
  RETURN [() SELECT_utasítás ()]
```

- Meghívás:

```
SELECT * FROM fgvnév(<paraméterek>)
```

Multi-statement Table-Valued Functions (MSTVF)

- Nagyon hasonló az ITVF-hoz; DE:
  - tábla típusú változót térít vissza;
  - BEGIN/END blokk megléte szükséges – belül: tábla típusú változó feltöltése sorokkal + más műveletek, melyek nem írnak ki eredményt.
  - van RETURN.
- Szintaxis:

```
CREATE FUNCTION fgvnév
   ( [{@paraméternév adattípus}[,…n]] )
RETURNS @TáblaNév TABLE (<attribútumok>)
[AS] BEGIN
        <műveletek>
        INSERT INTO @TáblaNév
           SELECT ...
        RETURN
     END
```

- Meghívás (egyezik az ITVF meghívásával):

```
SELECT * FROM fgvnév(<paraméterek>)
```

Példa (ITVF)

```
CREATE FUNCTION fnJolFizetettek_ITVF
                        (@fizetes INT)
RETURNS TABLE
AS
 RETURN
    SELECT SzemSzam, Nev,
           CAST(FelvetelDatuma AS DATE) AS FD
    FROM Alkalmazottak
    WHERE Fizetes>@fizetes
```

Meghívás: `SELECT * FROM fnJolFizetettek_ITVF(500)`

- **Megj.:** Táblaszerkezet megadása hiányzik.

`Alkalmazottak(SzemSzam, Nev, Fizetes, ReszlegID, FelvetelDatuma)` [FelvetelDatuma - datetime típusú]

Példa (ITVF)

```
CREATE FUNCTION fnJolFizetettek_ITVF
                        (@fizetes INT)
RETURNS TABLE
AS
 RETURN
   SELECT SzemSzam, Nev,
          CAST(FelvetelDatuma AS DATE) AS FD
   FROM Alkalmazottak
   WHERE Fizetes>@fizetes
```

Meghívás: `SELECT * FROM fnJolFizetettek_ITVF(500)`

- **Megj.:** Táblaszerkezet megadása hiányzik.

- **CAST** vagy **CONVERT**? – lsd. https://teamsql.io/blog/?p=1455

Példa (MSTVF)

```
CREATE FUNCTION fnJolFizetettek_MSTVF
                    (@fizetes INT)
RETURNS @JolFizAlkTabla
    TABLE (SzemSzam  VARCHAR(10),
           Nev VARCHAR(30), FD DATE)
AS BEGIN
    INSERT INTO @JolFizAlkTabla
      SELECT SzemSzam, Nev,
             CAST(FelvetelDatuma AS DATE)
      FROM Alkalmazottak
      WHERE Fizetes>@fizetes
    RETURN
  END
```

Alkalmazottak(SzemSzam, Nev, Fizetes, *ReszlegID*, FelvetelDatuma)

Meghívás: `SELECT * FROM fnJolFizetettek_MSTVF(500)`

Példa

Alkalmazottak tábla

	SzemSzam	Nev	Fizetes	ReszlegID	FelvetelDatuma
1	111111	Nagy Éva	300	2	2000-01-15 11:05:30.340
2	123444	Vincze Ildikó	800	1	2018-03-12 08:12:22.043
3	222222	Kiss Csaba	400	9	2015-11-24 12:02:46.060
4	234555	Szilágyi Pál	700	2	2018-03-12 08:12:21.033
5	333333	Kovács István	500	2	2018-10-22 09:52:02.003
6	456777	Szabó János	900	9	2015-12-24 15:32:26.060

Függvényhívás(ok) eredménye:

	SzemSzam	Nev	FD
1	123444	Vincze Ildikó	2018-03-12
2	234555	Szilágyi Pál	2018-03-12
3	456777	Szabó János	2015-12-24

ITVF vagy MSTVF?

- ITVF segítségével – módosítás is lehetséges (MTVF esetén nem)
- Példa:

```
UPDATE fnJolFizetettek_ITVF(0)
SET Nev= 'Kis Éva'
WHERE SzemSzam = '111111'
```

$\Longrightarrow$ Nagy Éva $\longrightarrow$ Kis Éva

```
UPDATE fnJolFizetettek_MSTVF(0)
SET Nev= 'Kis Éva'
WHERE SzemSzam = '111111'
```

$\Longrightarrow$ Hiba: `'Object 'fnJolFizetettek_MSTVF' cannot be modified.'`

ITVF vagy MSTVF?

- ITVF segítségével – módosítás is lehetséges; *fontos: nem csak alaptáblára!*
- Példa:

```
CREATE FUNCTION fnJolFizetettekUj_ITVF(@fizetes INT)
RETURNS TABLE
AS RETURN
   SELECT SzemSzam, Nev,
          CAST(FelvetelDatuma AS DATE) AS FD, RNev
  FROM Alkalmazottak JOIN Reszlegek
    ON RID=ReszlegID
  WHERE Fizetes>@fizetes
```

```
Reszlegek(RID, RNev)
Alkalmazottak(SzemSzam, Nev,
              Fizetes,ReszlegID,
              FelvetelDatuma)
```

```
UPDATE fnJolFizetettekUj_ITVF(0)
SET RNev= 'UjReszleg'
WHERE SzemSzam = '111111'
```

→ Részlegek táblában módosít.

ITVF vagy MSTVF?

- Hatékonyság – ITVFk hatékonyabbak, mint a MSTVFk; oka: SQL szerver különbözőképp kezeli őket:
  - ITVF ⟷ nézet
  - MSTVF ⟷ tárolt eljárás

Előnyök

- Moduláris programozás lehetősége.
- Hatékonyság.
- Hálózati adaforgalom csökkentése.

Megszorítások függvények esetén

- Nem használhatunk temporális táblákat.
- SELECT utasítás használata - eredmény „kimentése” egy változóba (kivéve – ITVF-k esetén)
- PRINT + RAISERROR – nem használható.
- Utolsó utasítás: RETURN
- Tárolt eljárás hívása függvényblokkon belül – nem lehetséges.
  - Fordított eset lehetséges.

Tárolt eljárás és függvény módosítása és törlése

- **Módosítás:**

CREATE HELYETT: ALTER

Példa:

```
[CREATE OR ]ALTER PROCEDURE spReszlAlkUpdate
    (@pRNev VARCHAR(30), @pOut INT OUT)
 AS
 BEGIN
 …
 END
```

Tárolt eljárás és függvény módosítása és törlése

- Törlés: DROP

Ellenőrzéssel:

```
IF EXISTS (SELECT * FROM sys.objects
 WHERE object_id = OBJECT_ID(N'<nev>') AND
type IN (N'P', N'PC',N'FN', N'IF', N'TF') )
 DROP PROCEDURE/FUNCTION <nev>
```

SQL 2016-tól: IF EXISTS argumentum használatával

```
DROP {PROC|PROCEDURE|FUNCTION} [IF EXISTS]
 {[schema_name.]<nev>} [ ,...n ]
```

Példa:
```
DROP FUNCTION IF EXISTS
                fnJolFizetettek_ITVF
```